National College of Ireland

BSc (Honours) in Computing

Cybersecurity

2020-2021

Dovydas Girskis

X17454656

X17454656@student.ncirl.ie

MOTOTO

Technical Report

Contents

Executive Summary

The aim of this report is to demonstrate how the project came to be and what aspect were considered in terms of implementation of different functions as well as security features. The intention of the project is to provide an automotive platform for buying and selling cars, allowing users to refine their search to a specific car with the help of filters, in turn saving a lot of time compared to typical filters. What many competing platforms do not address is the features aspect of a car where for example a person may not necessarily mind the make or model, but they are looking for features such as parking sensors and cameras to aid and put a driver at ease.

The major points of the report include the technologies used, the design and architecture behind the project as well as the implementation of security features and code that run the project in its current state. Testing is covered in terms of security vulnerability which also demonstrates how they were mitigated, there is also frontend user testing to demonstrate the ease of use of Mototo. In general, the project is currently at an acceptable point of vulnerability, but it can also be improved in the future which I go over in further development and research. There, I also mention different ideas that can be implemented into the project which can make the project truly shine and someday be deployed online to compete with existing platforms such as DoneDeal.

1.0 Introduction

1.1. Background

The purpose of the project was to develop a website which somewhat competes with the likes of DoneDeal and others. It came about when my friend was looking to get a car for himself and struggled to find exactly what he was looking for in terms of features. I decided that if I can help eliminate that then it is worth a shot making something he could use in the future. As mentioned, DoneDeal is currently the top website for buying and selling cars, and while it has made a lot of changes throughout the years, especially with an extensive filtering system, they do not have a features system. Features typically include reversing cameras, parking sensors, Android Auto and many more. While many, but not all listings (especially from non-dealers) contain information regarding features, it is not always as extensive or easy to find. With so many variations in car makes, models and features, I decided to fill that gap in by creating my own website.

Mototo aims to provide a simple yet effective user experience where a user can find a car and view the features available, provided by the seller. The biggest contribution would be trying to define and explore different methods of security as I am doing cybersecurity as my specialisation. During my internship, I was able to grasp how important security is and how it is currently a huge concern for many online platforms. Due to the nature of the Mototo, there is not necessarily a lot of sensitive information, but where there is, I have implemented security features to prevent a malicious actors accessing that information. An example of this would be password hashing. As the passwords are stored in the database, it is important not to store them in plaintext. PHP provides a great function for this by SALTING the passwords, meaning decryption would take a very long time. Fortunately, this function does not impede with the overall user experience, nor does it slow the system down.

Mototo follows the latest OWASP Top Ten Web Application Security Risks. While not all these risks are associated with Mototo, the ones covered are authentication and access control. Each user who registers and then logins go through an authentication procedure where it compares and checks with the database before they can login. After logging in, the user access control is defined by their ID, meaning that they cannot access another account unless of course they would know the password. This also includes user sessions, meaning that the system is aware of when a user is logged in our out and will therefore display the information accordingly. If a user attempts to directly access a section of the website, they will be unable to due to the session monitoring.

1.2. Aims

The aim of this project is to create a website with security in mind. This means that user credentials are stored safely without the risk of them being access by a malicious actor. There is also a separation of access implemented in the system, meaning that you may only delete your own adverts and not others'.

With this type of project, it is important to consider implementing confidentiality, integrity and availability, also known as "CIA", for short. Confidentiality means that data can only be accessed if the user has permission to do so, it is not accessible by regular users, especially where there is sensitive information. Integrity (which corelates to what I outlined earlier regarding deleting adverts) means that data cannot be changed by unauthorized users. It ensures the user is seeing true, unaltered information. Availability means that data can be accessed quickly, and the user can only see what is presented to them.

The main concern is adding too much unnecessary security which may change the behaviour of the website. This can include the introduction of new bugs as well as increased loading times which goes against availability as mentioned earlier. Due to the nature of the project, the best approach to security implementation is minimizing the attack surface area. Keep planned features to a minimum and minimize vulnerabilities as much as possible without overdoing it as it can introduce more problems down the line.

1.3. Technologies

XAMPP

I used this to develop the website offline on my local machine. It provides support for MySQL, PHP and others from the get-go. I was able to configure the database through phpMyAdmin which provides many administration tools in a simple-to-use GUI.

MySQL Database

As mentioned, this is a feature of XAMPP, more specifically phpMyAdmin which can be used as a standalone. This has allowed me to create a database to store user credentials, car makes/models, car features as well as ads posted on the website.

PHP

The whole project is created using PHP. The reason I used PHP is because I'm familiar with it and it can be embedded in HTML. This means that traditional .html file extensions are changed to .php. It also means that when viewing the source code, all PHP functions are hidden by default which is great because it can prevent malicious actors taking advantage of the code.

HTML

As it is a website, HTML provides a lot of the visual qualities. It is used in conjunction with Bootstrap to make the website work on all types of devices. Alongside this, there is also the use of custom CSS which allowed me to portray the website in a more modern, appealing manner.

JavaScript

There are a few JavaScript files adapted to the website which are used for small functions such as returning to the top of the page as well as clicking through images. These small scripts contribute to the small things in terms of functionality to the overall feel of the website

1.4. Structure

The purpose of this report is to give insight into all the different functionalities available, how everything works and what problems have been addressed. I will be going into the different requirements for the project, overall design and architecture of the system. After that I will be showing the fundamental code behind the functionalities and how everything came together behind the GUI. The plan is to then provide some results from testing the system and shining some light on the security aspect of the project. Finally, I will present my evaluation of the results and end with my conclusion of the project's future and development.

2.0 System

2.1. Requirements

In an ideal use case including the results from frontend user testing, it proved that users can easily navigate the system without any issues. While there are no necessary user requirements including some sort of IT technicality, the general flow follows many other websites users avail of daily.

2.1.1. Functional Requirements

Login

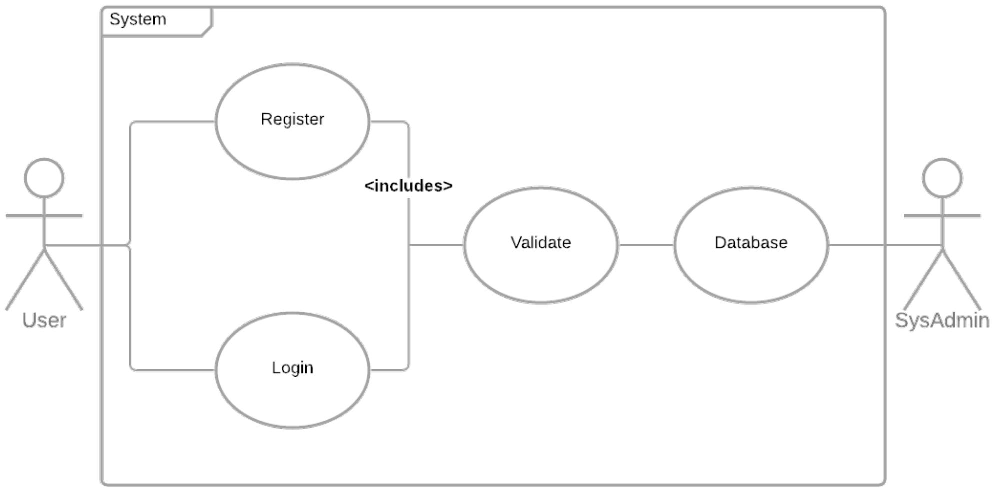
Scope

The scope of this use case is to show a user logging in.

Description

This use case describes the User logging in and how the System validates entries.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet. The user needs to have an account in the system so they can login using an email and password. If the user does not have an account, they can create one.

Activation

This use case starts when the User clicks on “Sign in” on the website.

Main flow

1. The User clicks on “Sign in” on the home page of the website.
2. The System displays a login page for the User with a username and password. (A1) (E1)
3. The User enters their username and password.
4. The System validates the username and password against the Database.
5. The System redirects the User to the homepage (index.html)

Alternate flow

A1: *Correct email and password*

1. The User Enters an invalid email or password.
2. The System tells the user that the email or password is incorrect and to try again.
3. The User then enters the correct email and password.
4. The Use Case continues.

Exceptional flow

E1: *Incorrect email or password*

1. The User enters their email and password.
2. The System detects an incorrect email or password.
3. The System does not allow the User to login successfully.
4. The User continues entering an incorrect email or password.
5. The User is unable to login.

Termination

The System will disable a user account if multiple attempts are made, and logs will show more information for the Sysadmin to view.

Post condition

The System allows the User to access homepage functions of the website.

Data requirements

Passwords must contain unique characters and must be over six characters long.

User requirements

The User needs to have a unique username which has not been taken.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to enter their credentials correctly or as required/indicated.

Register

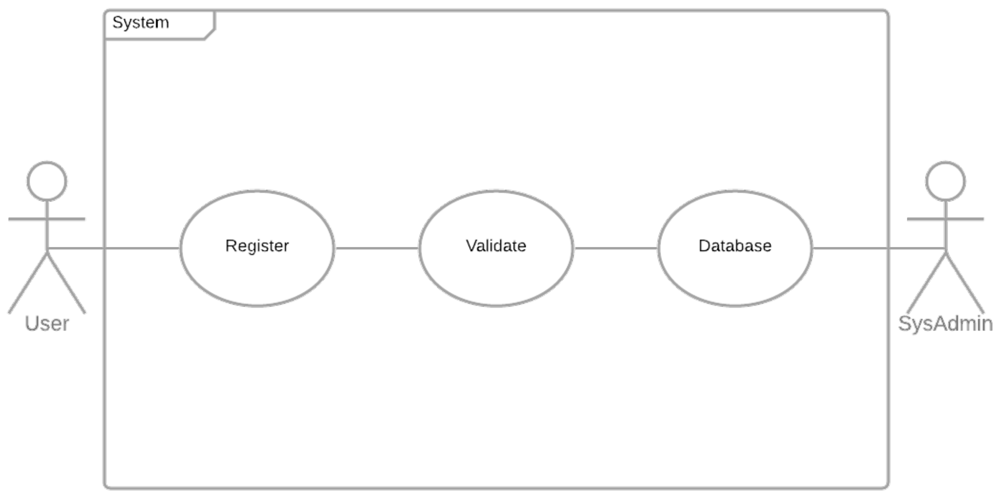
Scope

The scope of this use case is to show a user registering.

Description

This use case describes the User registering and how the System validates entries.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet.

Activation

This use case starts when the User clicks on “Sign in” on the website.

Main flow

1. The User clicks on “Register” on the Login page of the website.
2. The System displays a registration page for the User to enter a name, email and password. (A1) (E1)
3. The User enters their name, email and password.
4. The System validates the email and password against the Database.
5. The User logs in successfully.
6. The System redirects the User to the homepage (index.php)

Alternate flow

A1: *Correct email and password*

5. The User Enters an invalid username or password.
6. The System tells the user that the username or password is incorrect and to try again.
7. The User then enters the correct username and password.
8. The Use Case continues.

Exceptional flow

E1: *Incorrect email or password*

1. The User enters their email and password.
2. The System detects and displays that the email exists.
3. The System does not allow the User to register successfully.

Termination

The System will not allow a User to register if the email exists.

Post condition

The System allows the User to register successfully.

Data requirements

Email must not already exist in the Database.

User requirements

The User needs to have a unique email which has not been taken.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to enter their credentials correctly or as required/indicated.

Post Advert

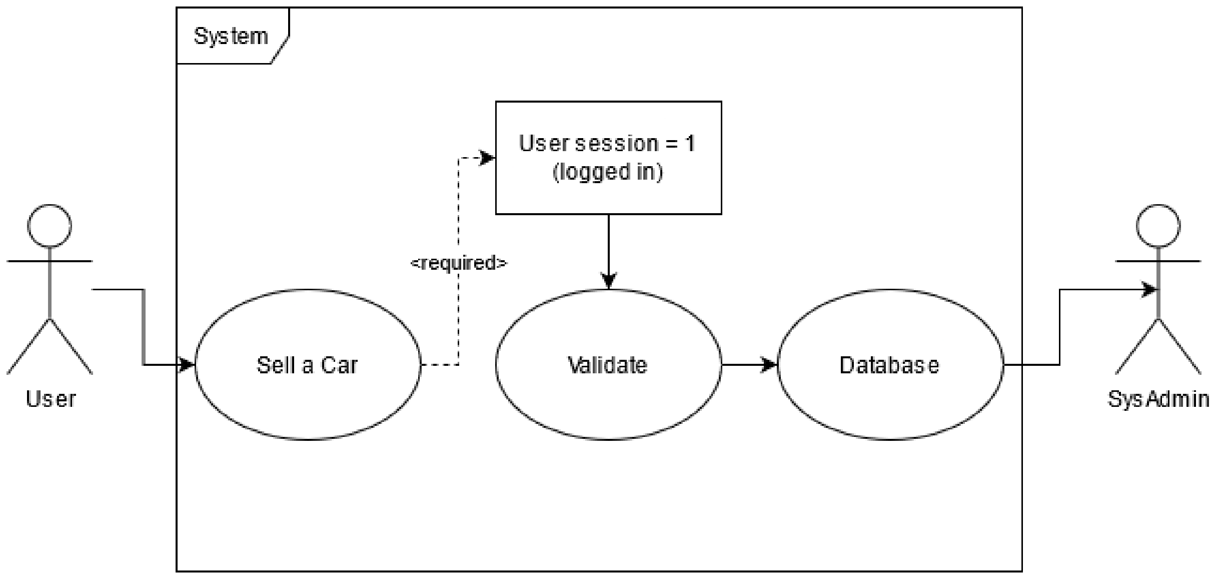
Scope

The scope of this use case is to show a User selling a car.

Description

This use case describes the User selling a car and how the System inserts entries.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet and must be logged in.

Activation

This use case starts when the User clicks on “Sell a Car” via the dropdown menu.

Main flow

1. The User clicks on “Sell a Car” on the dropdown menu of the website.
2. The System displays a form for the User to enter/select the required details. (A1) (E1)
3. The User enters the required details.
4. The System inserts the information into the Database.
5. The User posts an advert successfully.
6. The System redirects the User to the homepage (index.php)

Alternate flow

A1: *All fields are filled*

1. The User does not fill out required field.
2. The System tells the User that the field must be filled.
3. The User fills the required field correctly.
4. The Use Case continues.

Exceptional flow

E1: *Invalid entry in field*

1. The User fills out the required fields.
2. The System detects an invalid/unselected field.
3. The System does not allow the User to post an advert.

Termination

The System will not allow a User to post an advert if fields are not entered/selected correctly.

Post condition

The System allows the User to post an advert successfully.

Data requirements

Make/model is retrieved from the Database.

User requirements

The User needs to follow the instructions and fill out the fields correctly and must be logged in.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to enter their details correctly or as required/indicated.

Change Password

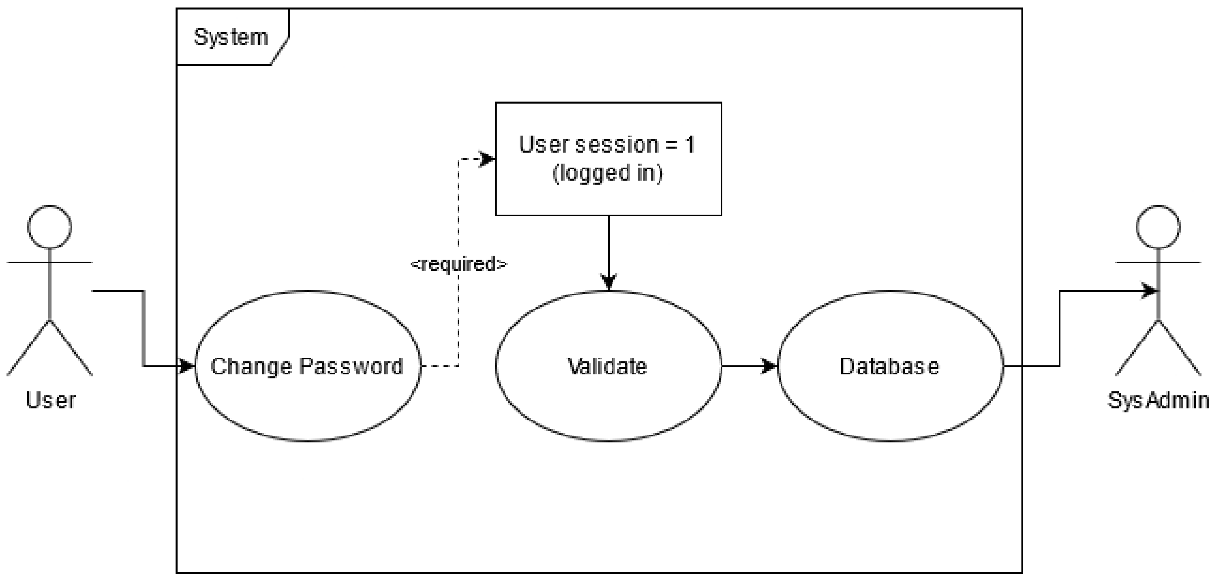
Scope

The scope of this use case is to show a User changing their password.

Description

This use case describes the User changing their password and how the System updates entries.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet and must be logged in.

Activation

This use case starts when the User clicks on “Settings” via the dropdown menu.

Main flow

1. The User clicks on “Settings” on the dropdown menu of the website.
2. The System displays a form for the User to change their password. (A1) (E1)
3. The User enters the required details.
4. The System inserts the details into the Database.
5. The User changes their password successfully.
6. The System redirects the User to the same form (account-settings.php)

Alternate flow

A1: *All fields are filled*

1. The User enters the wrong password.
2. The System tells the User that the password entered is incorrect.
3. The User enters the password correctly.
4. The Use Case continues.

Exceptional flow

E1: *Invalid password in field*

1. The User fills out the required fields.
2. The System detects an invalid password.
3. The System does not allow the User to change their password.

Termination

The System will not allow a User to change their password if they have entered the incorrect password.

Post condition

The System allows the User to change their password successfully.

Data requirements

Password is temporarily compared to Database.

User requirements

The User needs to follow the instructions and fill out the fields correctly and must be logged in.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to enter their details correctly or as required/indicated.

Log Out

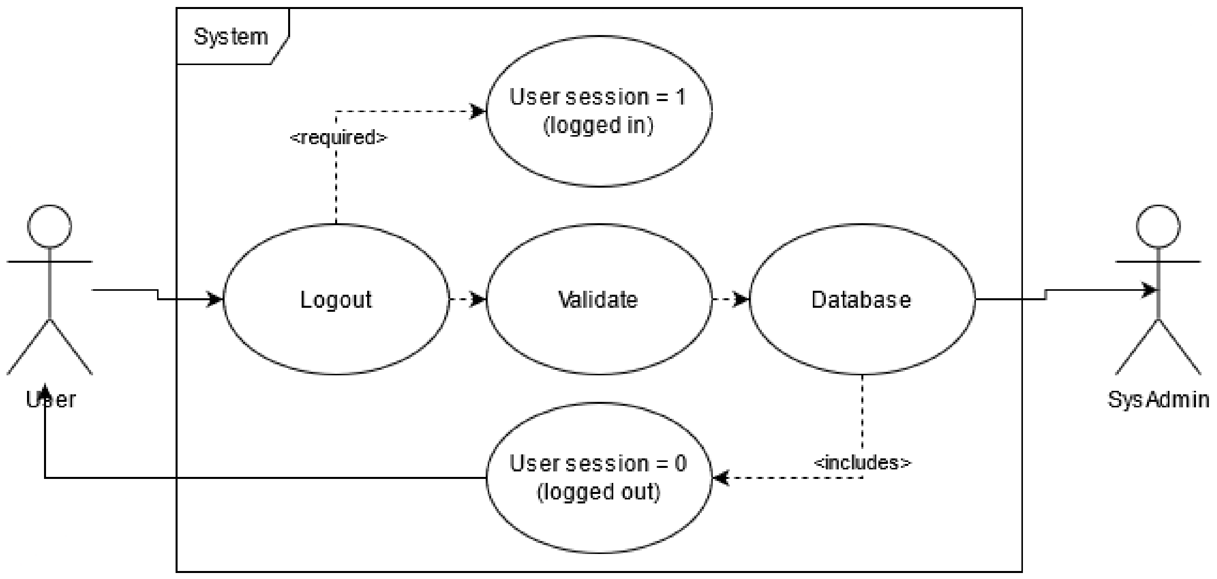
Scope

The scope of this use case is to show a User logging out and the System deleting a session.

Description

This use case describes the User logging out and how the System updates the session.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet and must be logged in.

Activation

This use case starts when the User clicks on “Logout” via the navbar.

Main flow

1. The User clicks on “Logout” on the navbar of the website.
2. The System executes logout.php (A1) (E1)
3. The System redirects the User to the same form (account-settings.php)

Alternate flow

A1: *User is logged in*

1. The User attempts to logout.
2. The System tells the User that there are unsaved changes.
3. The User decides if they want to logout or continue.
4. The Use Case continues.

Exceptional flow

E1: *Invalid logout*

1. The User attempts to logout by accessing logout.php
2. The System compares user session to Database.
3. The System does not allow the User to access the script unless they are logged in under User session = 1

Termination

The System will not allow a User to log out if they are already logged out as logout.php is hidden under User session = 0

Post condition

The System allows the User to logout successfully.

Data requirements

User session = 0/1 is compared to Database.

User requirements

The User needs to click "Logout" and they must be logged in.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to logout or continue to unsaved changes.

Filter

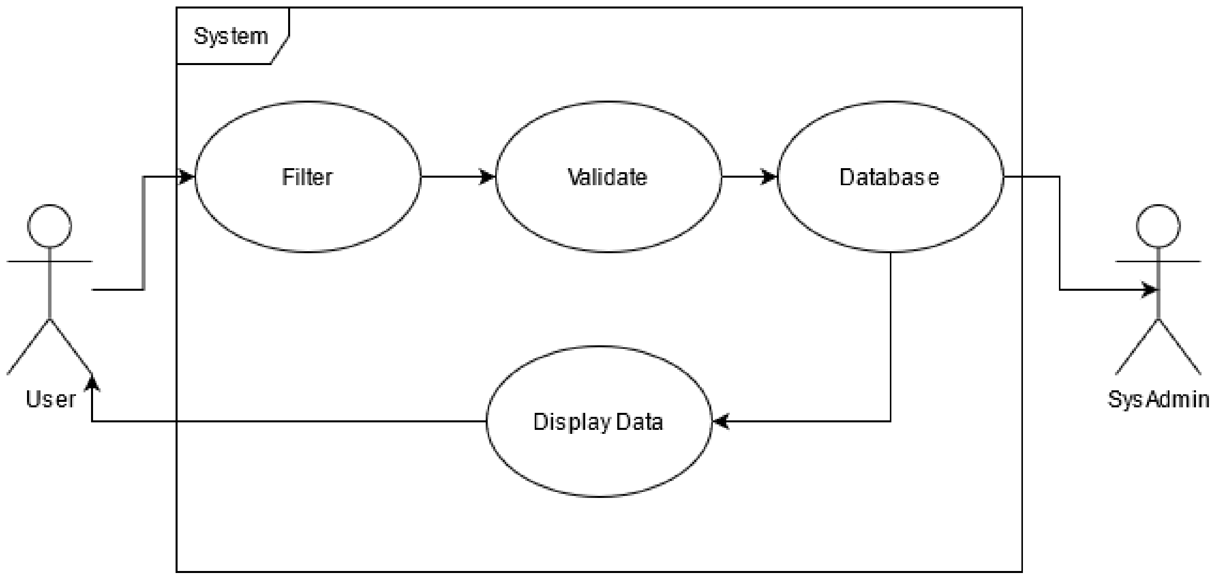
Scope

The scope of this use case is to show a User using the filter function.

Description

This use case describes the User using the filter function and how information from Database is retrieved to view.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet and must be logged in.

Activation

This use case starts when the User changes entries on filter list.

Main flow

1. The User clicks makes change to filter list.
2. The System displays information relevant to the filter list. (A1) (E1)
3. The User selects the required filters.
4. The System fetches the details from the Database.
5. The User can view the details.

Alternate flow

A1: *Filters are selected*

1. The User clicks makes change to filter list.
2. The System displays information relevant to the filter list. (A1) (E1)
3. The User selects the required filters.
4. The System fetches the details from the Database.
5. The User can view the details.

Exceptional flow

E1: *Filters display no results*

1. The User selects a field.
2. The System detects and displays that that no matches have been found.
3. The System does not allow the User to view any available cars under the filter.

Termination

The System will not allow a User to view cars if the filter is not selected correctly.

Post condition

The System allows the User to view the available cars.

Data requirements

Filter field must match table in Database.

User requirements

The User needs to click "Filter" to display results.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to select the filter fields accordingly.

Delete

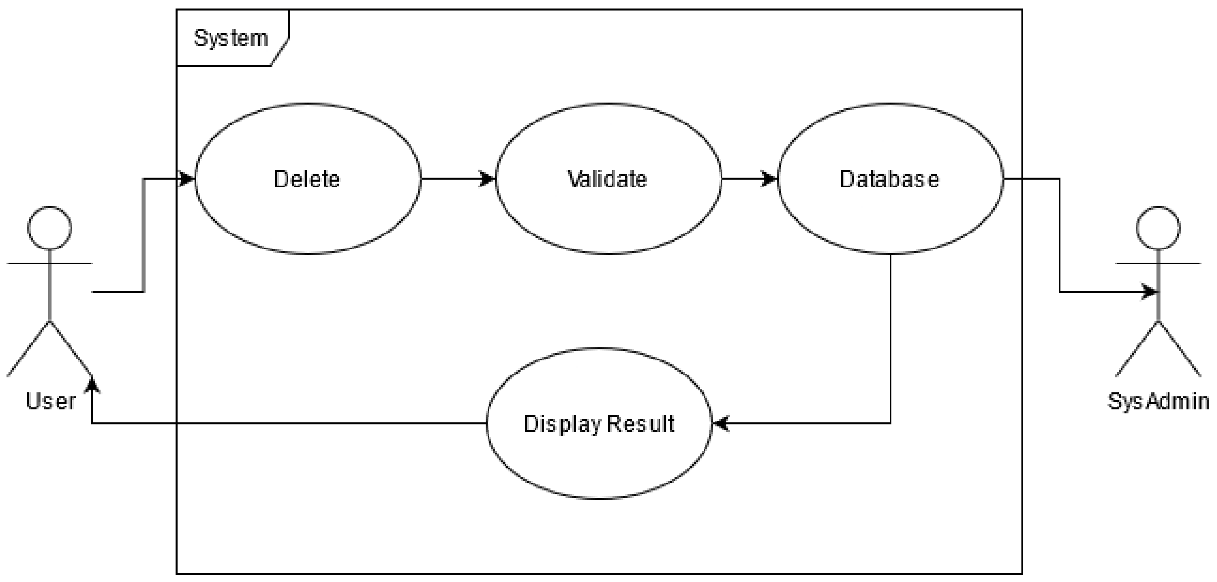
Scope

The scope of this use case is to show a User deleting their advert.

Description

This use case describes the User deleting adverts how the System validates and presents these changes.

Use Case Diagram



Flow Description

Precondition

The user needs to have access to a web browser and internet. The User must be logged in. The User must have an advert posted.

Activation

This use case starts when the User clicks on “Delete” in the “My Cars” section.

Main flow

1. The User clicks on “Delete” in the “My Cars” section.
2. The System executes a query and deletes the advert associated with the user account, specifically their ID.
3. The System redirects the User to the “My Cars” section (my-vehicles.html)
4. The User will no longer see their advert, meaning the query has been successful.

Alternate flow

1. The User attempts to delete an advert which does not exist.
2. The User must create an advert to access the delete function.
3. The User can delete the advert after creating one.

Exceptional flow

1. The User is not logged in and attempts to access my-vehicles.php
2. The System will check if the user session = 1
3. The System detects that the user session is invalid.
4. The User cannot access my-vehicle.php unless they are logged in.

Termination

The System will not display a delete function is an advert does not exist.

Post condition

The System allows the User to delete their advert successfully.

Data requirements

The advert must exist in the database where it is fetched and removed upon the executed function.

User requirements

The User must have an advert posted.

Environmental requirements

The User must have access to the internet and a web browser.

Usability requirements

The User must be able to post an advert correctly.

2.2. Design & Architecture

Website Design

I think that the overall look and design of a website (Figure 1) is important to the user and therefore a lot of time was put in to achieve just that. Throughout many projects, design was very important to me as I wanted to make sure everything looks presentable to the user. When designing the website, I decided to go for a modern, minimal look using Bootstrap for responsiveness and Font Awesome for the icons scattered across the website. The overall colour scheme I was going for was purple, grey and white. A white canvas where information, pictures and other things are visible and easy to read. To add some flare, I added a touch of purple to “highlight” and make details pop. To add some sort of distinguishable separation of information, I used blocks of grey. I took inspiration of the colour scheme from websites such as Monster IE as well as Twitch as I thought the purple looked appealing.

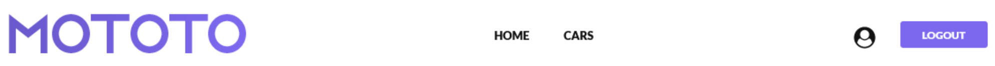


Figure 1

In terms of the logo (Figure 2), I went with the same colour scheme to match the rest of the website. Again, I went for something minimal when designing the logo. The font is called “Azonix” which I feel gave a nice modern look. The logo was designed on Adobe Photoshop. While it may not be very easily distinguished but there is a gradient which start from a slightly darker purple to a lighter one. This is also applied to the buttons on the website, once hovered on them, the slightly turn darker purple to provide some visual feedback.

Figure 2

Database Design

Here is the database for the website (Figure 3). Due to the nature of the website, it is very simple and straightforward. I will expand on each of the tables and their purpose.

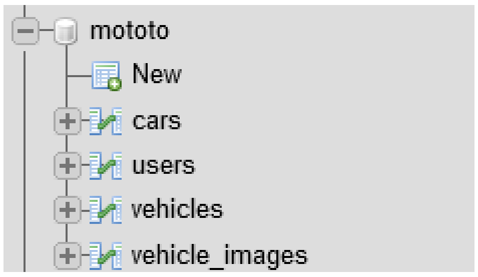
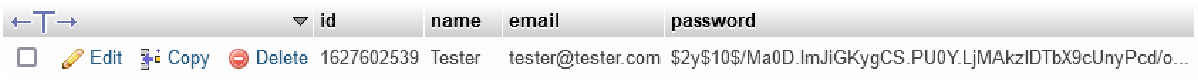


Figure 3

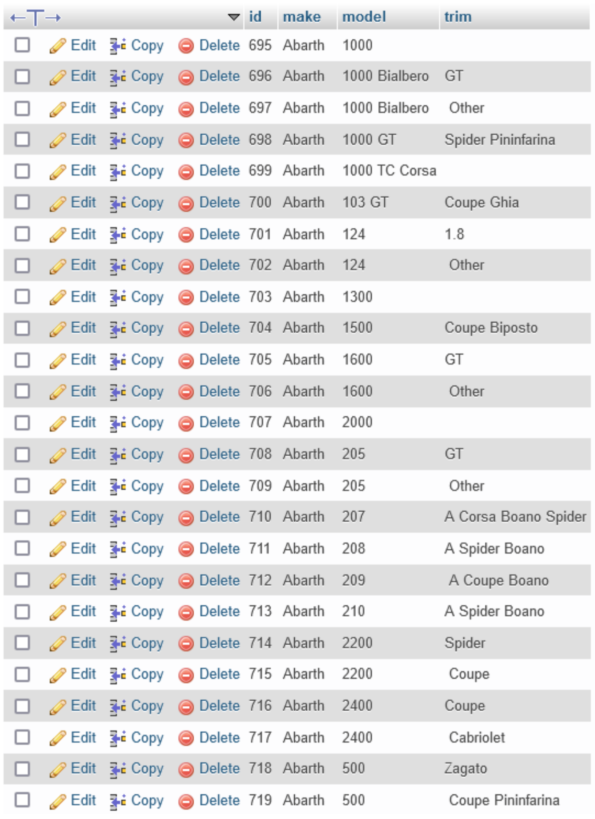
Below we can see the "users" table (Figure 4) which stores information on the user upon registration. It contains a unique ID, name, email and password. For the password, we can see that it is hashed rather than stored in plaintext. This is to ensure that if the database is breached, a malicious actor would not be able to see the actual password. Decrypting the password would take too long to consider even trying therefore it is important to protect it.



←T→				id	name	email	password
☐	Edit	Copy	Delete	1627602539	Tester	tester@tester.com	$2y$10$/Ma0D.lmJiGKygCS.PU0Y.LjMAkzlDTbX9cUnyPcd/o...

Figure 4

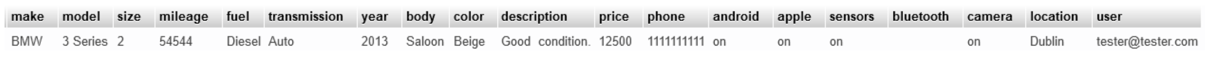
Next is the "cars" table (Figure 5) which of course contains all the different car makes and models. This is specifically used for the "Sell my Car" form and filter. At the very start of the project, the idea was to put in some of the most common makes and models and leave it at that, but I decided to try something different. After some research, I was able to come across a dump of different car makes and models which results to 12312 rows in total. While it may be too much for this type of project, I wanted to test my capabilities and the limits of the project to see if performance and other factors are affected. It contains the make, model and trim in some cases depending on the car model. This is accessed using PHP which takes the car make and then assigns the correct models to choose associated to that make.



←T→				id	make	model	trim
☐	Edit	Copy	Delete	695	Abarth	1000	
☐	Edit	Copy	Delete	696	Abarth	1000 Bialbero	GT
☐	Edit	Copy	Delete	697	Abarth	1000 Bialbero	Other
☐	Edit	Copy	Delete	698	Abarth	1000 GT	Spider Pininfarina
☐	Edit	Copy	Delete	699	Abarth	1000 TC Corsa	
☐	Edit	Copy	Delete	700	Abarth	103 GT	Coupe Ghia
☐	Edit	Copy	Delete	701	Abarth	124	1.8
☐	Edit	Copy	Delete	702	Abarth	124	Other
☐	Edit	Copy	Delete	703	Abarth	1300	
☐	Edit	Copy	Delete	704	Abarth	1500	Coupe Biposto
☐	Edit	Copy	Delete	705	Abarth	1600	GT
☐	Edit	Copy	Delete	706	Abarth	1600	Other
☐	Edit	Copy	Delete	707	Abarth	2000	
☐	Edit	Copy	Delete	708	Abarth	205	GT
☐	Edit	Copy	Delete	709	Abarth	205	Other
☐	Edit	Copy	Delete	710	Abarth	207	A Corsa Boano Spider
☐	Edit	Copy	Delete	711	Abarth	208	A Spider Boano
☐	Edit	Copy	Delete	712	Abarth	209	A Coupe Boano
☐	Edit	Copy	Delete	713	Abarth	210	A Spider Boano
☐	Edit	Copy	Delete	714	Abarth	2200	Spider
☐	Edit	Copy	Delete	715	Abarth	2200	Coupe
☐	Edit	Copy	Delete	716	Abarth	2400	Coupe
☐	Edit	Copy	Delete	717	Abarth	2400	Cabriolet
☐	Edit	Copy	Delete	718	Abarth	500	Zagato
☐	Edit	Copy	Delete	719	Abarth	500	Coupe Pininfarina

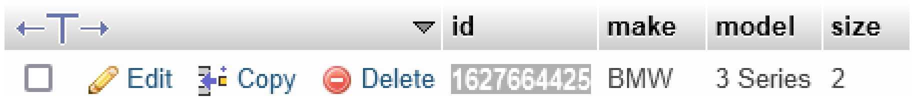
Figure 5

This is the “vehicles” table (Figure 6) which contains all the currently active adverts on the website and the associated information with it. This is all the information derived from the “Sell my Car” form which contains the make, model, engine size, mileage, fuel type, transmission type, body type and so on. While it may be difficult to make it out from this image, it will be highlighted more clearly during the presentation. This is associated with the user ID therefore only the user who has posted the advert can delete it or it can be removed/edited from the backend by an administrator. This would also be used if the user wanted to edit their listing if a small mistake was made.



make	model	size	mileage	fuel	transmission	year	body	color	description	price	phone	android	apple	sensors	bluetooth	camera	location	user
BMW	3 Series	2	54544	Diesel	Auto	2013	Saloon	Beige	Good condition.	12500	1111111111	on	on	on		on	Dublin	tester@tester.com

Figure 6



←T→	id	make	model	size
Edit Copy Delete	1627664425	BMW	3 Series	2

Figure 7

The final table is the “vehicle_images” table (Figure 7). This is where the images are stored when a user uploads them. They are tied to the vehicles table by ID so that the pictures can correctly correspond to the listing correctly. For example, in the “vehicles” table shown previously, the “vehicle_id” matches to that of the “vehicle_images” table as highlighted.

Architecture

General

Regarding the architecture it is relatively straightforward, it begins with displaying the information where a user can then proceed with inputs such as logging in or registering. It is then validated through the system before being compared to the database and changes are made accordingly. The information that exists can be used or new information can be stored, when registering or posting a car for sale for example. It is then majorly left for the backend, more specifically the database where changes can be made by the administrator.

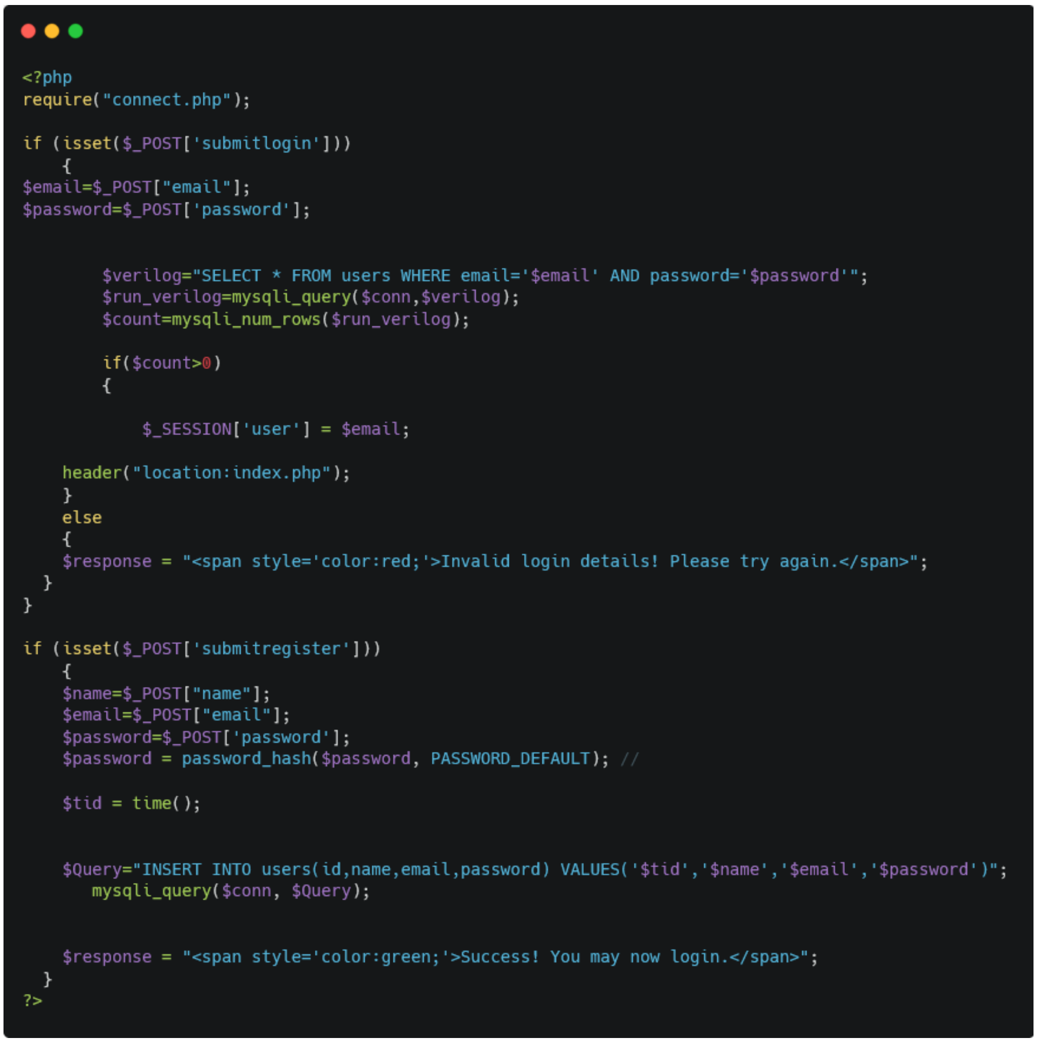
Login/Register
For the login, upon entering your details, it will do a quick check against the database. Due to passwords being hashed, it will temporarily take the actual password, hash it and compare it to the existing hash on the database before deciding if they match. If it matches, it will log the user in and set their session as “1” or “logged in” which will display different information compared to a user who is not logged in.

When registering, it follows the same principle of checking if the user already exists. If the user does not exist, it will add that information to the database where the user can then login under those credentials. The password is hashed immediately upon registering and it is not stored anywhere else.

Backend/Database
The backend primarily consists of the database where it can be accessed by the database administrator. From there they can manage tables and make changes if required. It deals with information being accessed by the website and new information being stored. If a user without permissions accesses the database, they cannot take much information from users due to the security system implemented. There are no security features for the actual database because a lot of it is performed using code, therefore the database is only for reading and storing information.

2.3. Implementation
Login/Register
Here we have a snippet of the login and register system (Figure 8) which is necessary when wanting to post an advert. The code is implemented on each page since I have used an overlay which renders on top of the content for the user to login or register. For the login, the users’ email and password is required and those are defined as POST methods $email and $post respectively. Once the details are entered and the user submits the information, a query is executed to select this information from the database and check if they are the same. If they are the same, then the user can login successfully. The user is then redirected to the homepage (index.php).

In terms of registering, the user must provide a name (which is used for contact details on a listing), email and password. The most important part here is the $password variable with the password_hash function. As discussed before, when hashing the password, it essentially cannot be “decrypted” as hashing is different from encrypting. When the user is logging in, a temporary check is made to compare the hash and the entered password to see if they match. In the case that they do the user is then able to login, otherwise they will have to try again.

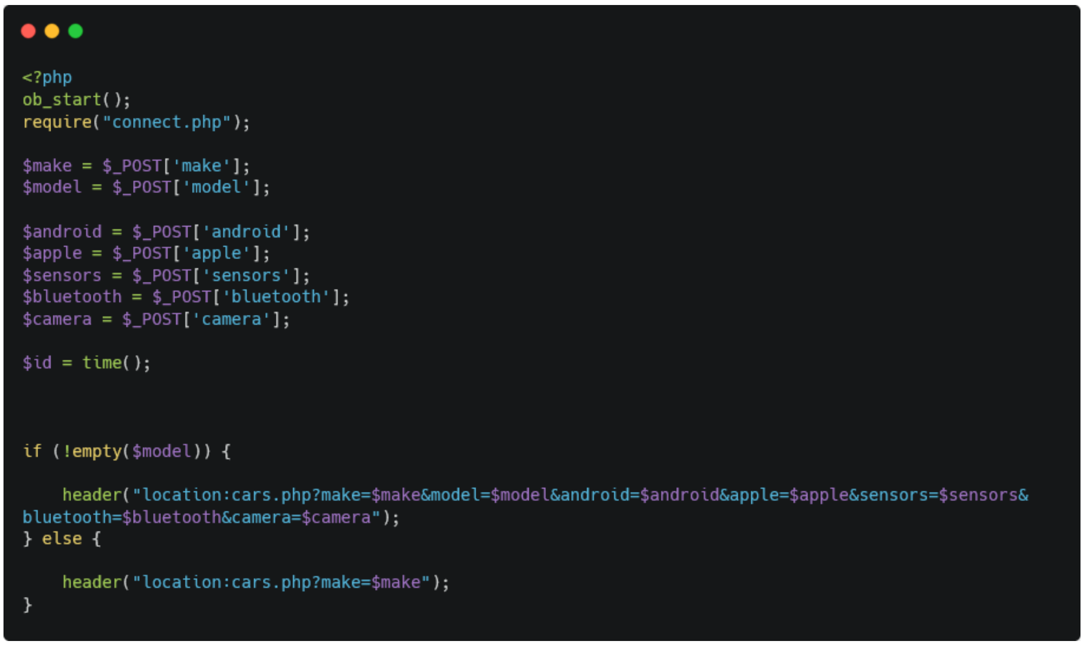


```
<?php
require("connect.php");

if (isset($_POST['submitlogin']))
    {
$email=$_POST["email"];
$password=$_POST['password'];


        $verilog="SELECT * FROM users WHERE email='$email' AND password='$password'";
        $run_verilog=mysqli_query($conn,$verilog);
        $count=mysqli_num_rows($run_verilog);

        if($count>0)
        {

            $_SESSION['user'] = $email;

    header("location:index.php");
    }
    else
    {
    $response = "<span style='color:red;'>Invalid login details! Please try again.</span>";
  }
}

if (isset($_POST['submitregister']))
    {
    $name=$_POST["name"];
    $email=$_POST["email"];
    $password=$_POST['password'];
    $password = password_hash($password, PASSWORD_DEFAULT); //

    $tid = time();


    $Query="INSERT INTO users(id,name,email,password) VALUES('$tid','$name','$email','$password')";
       mysqli_query($conn, $Query);


    $response = "<span style='color:green;'>Success! You may now login.</span>";
  }
?>
```

Figure 8

Filter System

This is the filter system (Figure 9), essentially what is happening is there are definitions for the make, model and features which is currently the parameters a car can be found with. This is tied with cars.php which displays the relevant cars to the search made through the code below. The important part here is selecting and attaching the filters together to display the wanted results and that can also be seen within the URL. It works by having a string with the definitions which equal to it, an example of a definition can be $make=BMW and $model=320 and so on.

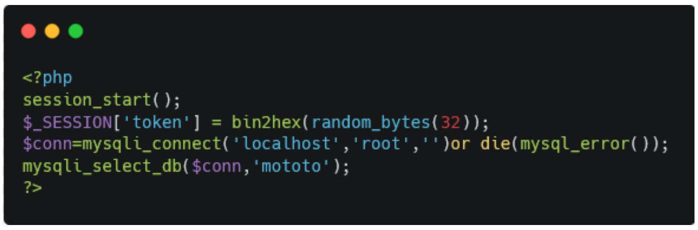


```php
<?php
ob_start();
require("connect.php");

$make = $_POST['make'];
$model = $_POST['model'];

$android = $_POST['android'];
$apple = $_POST['apple'];
$sensors = $_POST['sensors'];
$bluetooth = $_POST['bluetooth'];
$camera = $_POST['camera'];

$id = time();



if (!empty($model)) {

    header("location:cars.php?make=$make&model=$model&android=$android&apple=$apple&sensors=$sensors&
bluetooth=$bluetooth&camera=$camera");
} else {

    header("location:cars.php?make=$make");
}
```

Figure 9

Connect to Database

Here is how we connect to the database (Figure 10). To create or resume a session we use session_start and this is needed for POST or GET requests which are included in the login, register, sell car forms and more. It is simply connecting by defining the variables such as localhost (as we are hosting the database locally), root (as the administrator) and we are leaving the password blank because it currently does not need one. In the case that it is deployed online then a password will be added. The last step is to define which database to connect to and in this case, it is "mototo". The important part to highlight here is tying a token to the session when it is started. This is also known as a Cross Site Request Forgery (CSRF), and it is when a malicious user can gain access to a regular users' session and can executed actions which can be damaging. To circumvent this, CSRF protection was added.

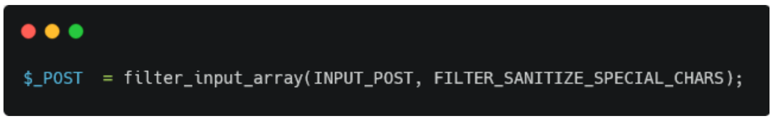


```php
<?php
session_start();
$_SESSION['token'] = bin2hex(random_bytes(32));
$conn=mysqli_connect('localhost','root','')or die(mysql_error());
mysqli_select_db($conn,'mototo');
?>
```

Figure 10

SQL Sanitization

Depending on the type of $POST or $GET requests in forms, filtering/sanitization is used to prevent SQL injections. In my case I have used a filtering system for forms (Figure 11). In this example we have one for changing the password then the new password is submitted. The FILTER_SANITIZE part can be changed to SPECIAL_CHARS or EMAIL if wanted, again, depending on the type of information being submitted. Luckily PHP has a lot of these variables available, what they do is in the case of an EMAIL variable is that you may only enter numbers, letters and '@' – you cannot enter % for example. Another simple way to do this for emails would be defining the input field as email but to be extra safe it is better to filter/sanitize these requests.

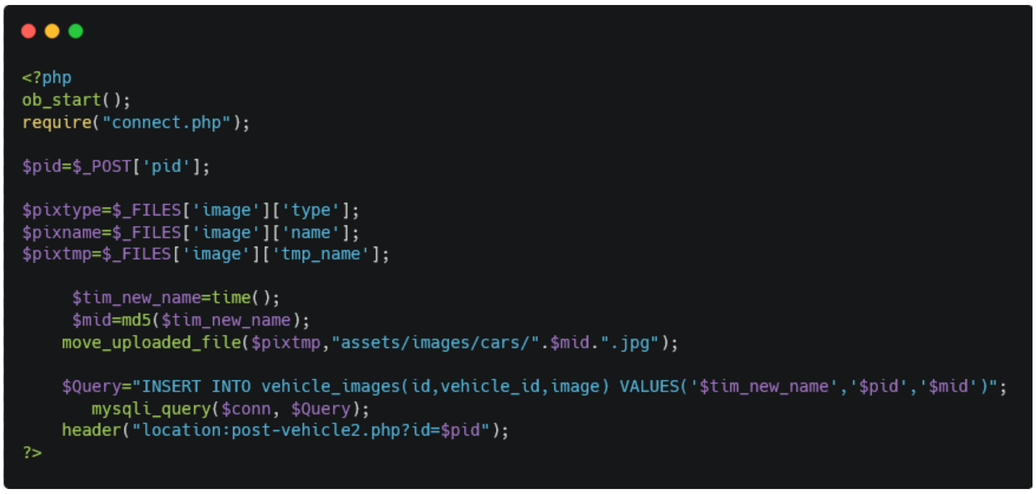


```php
$_POST  = filter_input_array(INPUT_POST, FILTER_SANITIZE_SPECIAL_CHARS);
```

Figure 11

Uploading Images

When uploading images, they are tied to the user ID, meaning that you can see which user(s) has uploaded which picture(s). I have also made it that when it is uploaded to the database, it also applies a quick MD5 to give a unique name. It then stores the picture in the images folder locally and when there is a query to view that specific picture, it will fetch it using the MD5 to make sure it is the correct one when displaying it. The reason it has been done like this is because I was having difficulty making a system where the image is uploaded in the database and displayed from there, luckily, I found a workaround which has worked for me.



```php
<?php
ob_start();
require("connect.php");

$pid=$_POST['pid'];

$pixtype=$_FILES['image']['type'];
$pixname=$_FILES['image']['name'];
$pixtmp=$_FILES['image']['tmp_name'];

     $tim_new_name=time();
     $mid=md5($tim_new_name);
    move_uploaded_file($pixtmp,"assets/images/cars/".$mid.".jpg");

    $Query="INSERT INTO vehicle_images(id,vehicle_id,image) VALUES('$tim_new_name','$pid','$mid')";
       mysqli_query($conn, $Query);
    header("location:post-vehicle2.php?id=$pid");
?>
```

Figure 12

2.4. Graphical User Interface (GUI)

Homepage

Here is the homepage of the website (Figure 13). When you are not logged in, you are able to login or register by clicking the “Login/Register” button on the top right of the website. You are then presented with an overlay to either login or register.

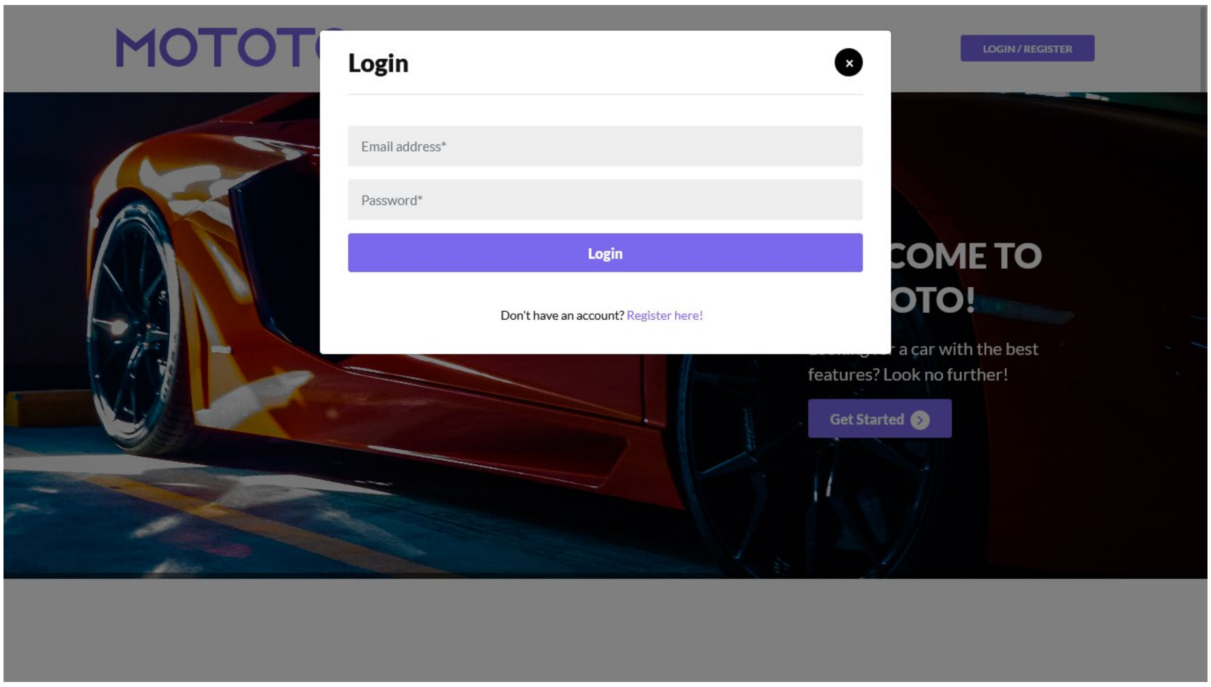


Figure 13

Options

When you are logged in and click on the user icon, you are presented with the ability to access your settings, your car adverts and the ability to post a car for sale (Figure 14). From here we can also sign out, or we can use the button on the top right to do so. As you can see, depending on the user session, the functions are changed. Before logging in there was a "Login/Register" button and it has now been replaced with a "Logout" button.

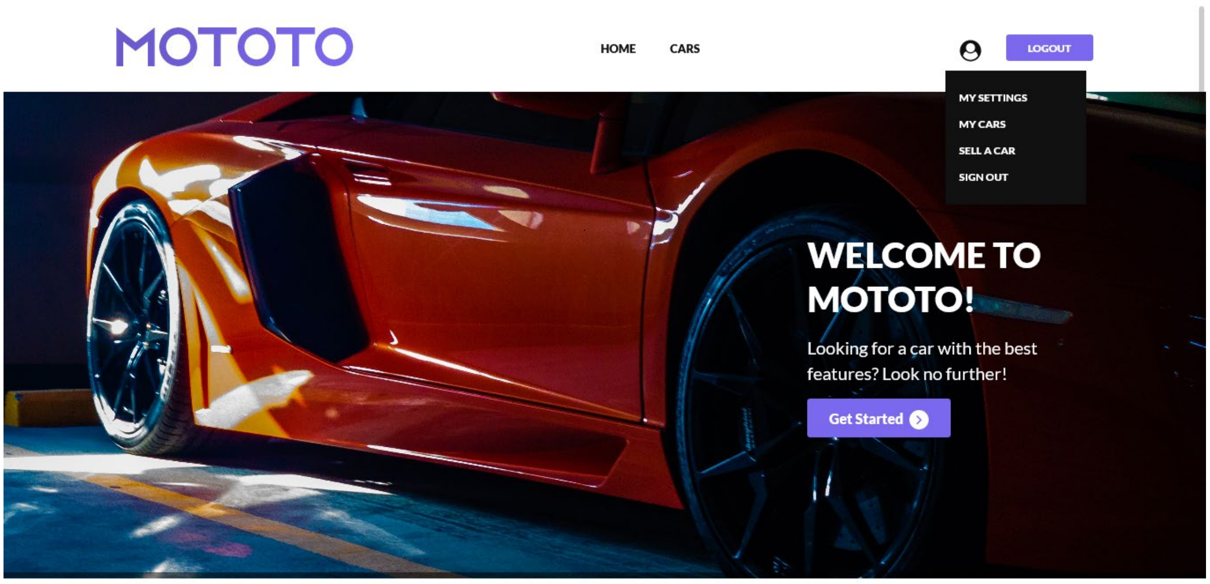


Figure 14

Latest Cars For Sale

If we scroll down on the homepage, we are greeted with the most recent cars for sale (Figure 15). For demonstration purposes I have put up a car for sale through a sample account. The user can see the car make and model, year, price and location.

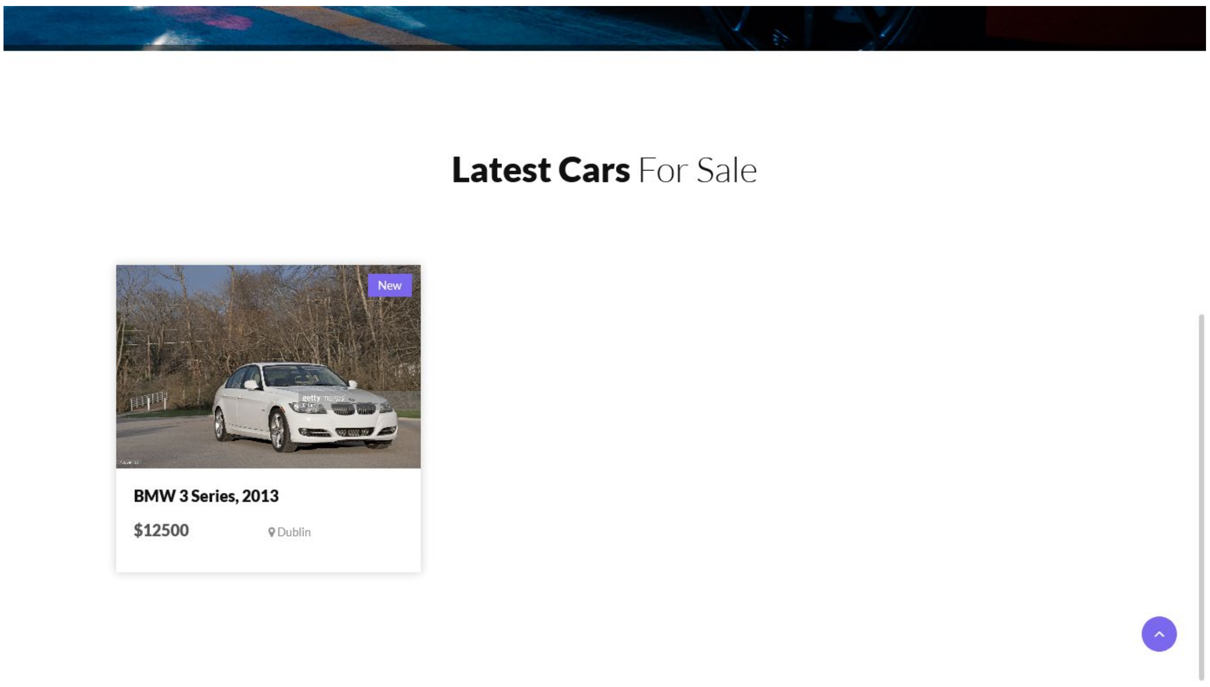
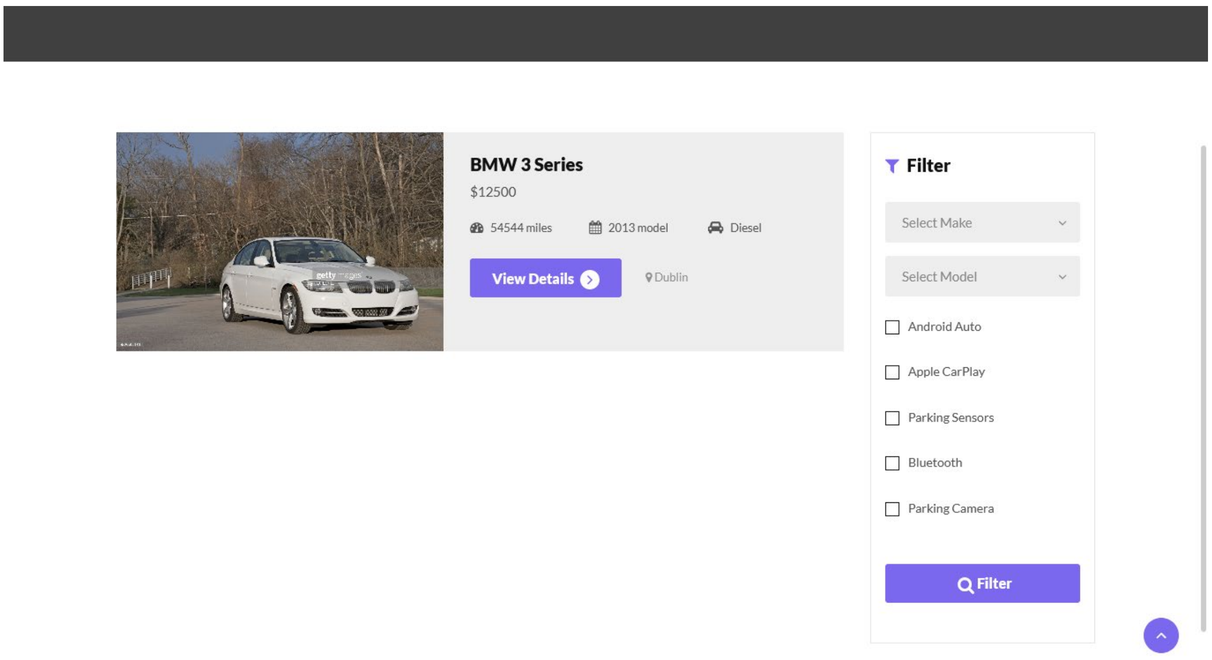


Figure 15

Cars Section

Here is the “Cars” section (Figure 16), as we can see it is the game car that was on the homepage listed under “Latest Cars For Sale”. Here, we can also see the filter which currently supports the make, model and the features. Currently, the filter is limited to the make, model and features for ease of use and to simply demonstrate how it works. The plan down the line is to implement much more parameters.



BMW 3 Series

$12500

54544 miles 2013 model Diesel

View Details Dublin

Filter

Select Make

Select Model

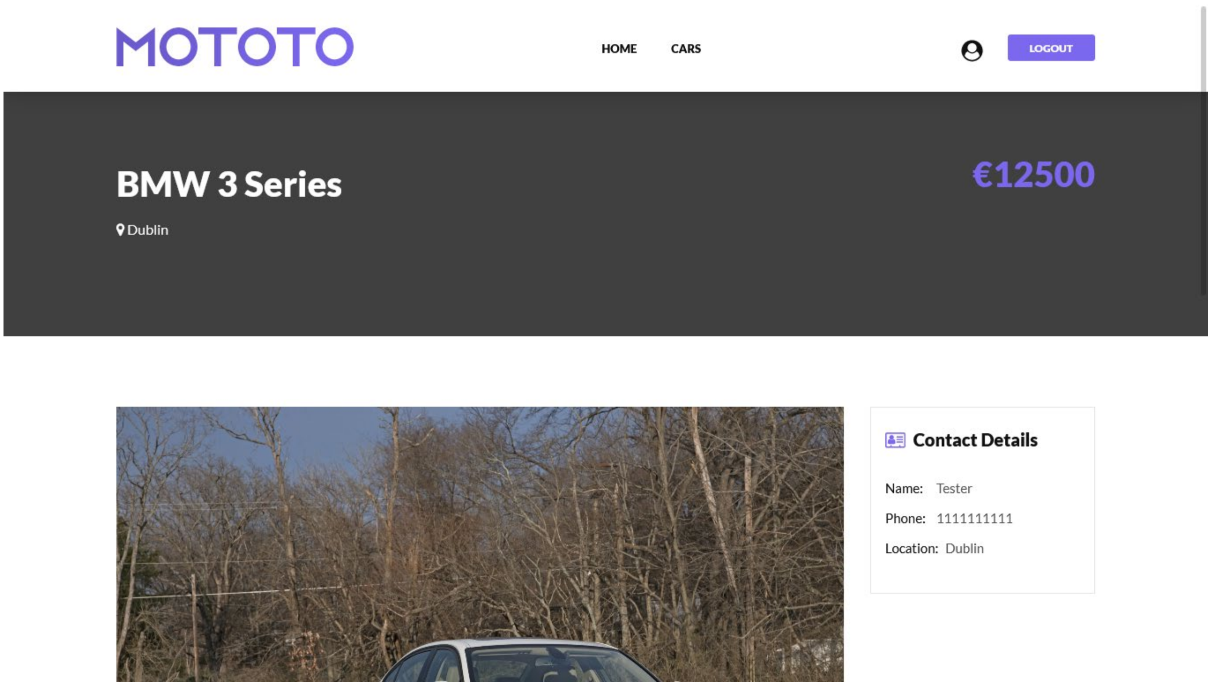
- [ ] Android Auto
- [ ] Apple CarPlay
- [ ] Parking Sensors
- [ ] Bluetooth
- [ ] Parking Camera

Filter

Figure 16

Car Details

If we click on “View Details” (Figure 17) it will present us with the following. At the very top it shows the make, model, price and location. This also contains the contact details of the seller. At the time of taking these screenshots, we are currently logged in as Tester for demonstration purposes. The name is filled out using the name that was provided upon registration which is stored in the database.

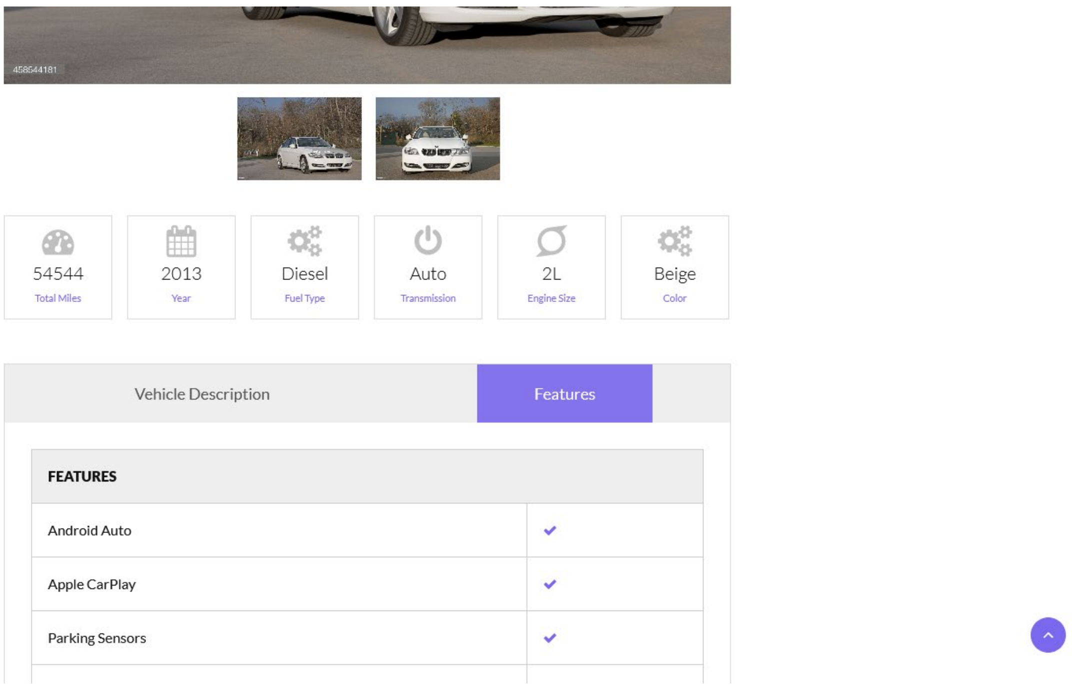


Name: Tester
Phone: 1111111111
Location: Dublin

Figure 17

Additional Car Details

On the same page, if we scroll down furthermore (Figure 18), we can see additional information that the seller has provided. Here we can see the milage, year, fuel type, transmission type, engine size and the colour. There are two tabs, “Vehicle Description” and “Features” down below where the seller can add additional information if necessary. On the “Features” tab, we can see that the seller has put in the features the car has. The feature list can be expanded, if necessary, also.



FEATURES	
Android Auto	✓
Apple CarPlay	✓
Parking Sensors	✓

Figure 18

My Cars

Upon clicking on “My Cars” (Figure 19) we are presented with the following below. Here, we can see which car(s) we have posted, this is because a user is not limited to just one advert, they can post as many as they like and delete them if necessary. Now, there is only a “Delete” function but in the future, I would like to implement a system to edit car postings too, in case a mistake was made.

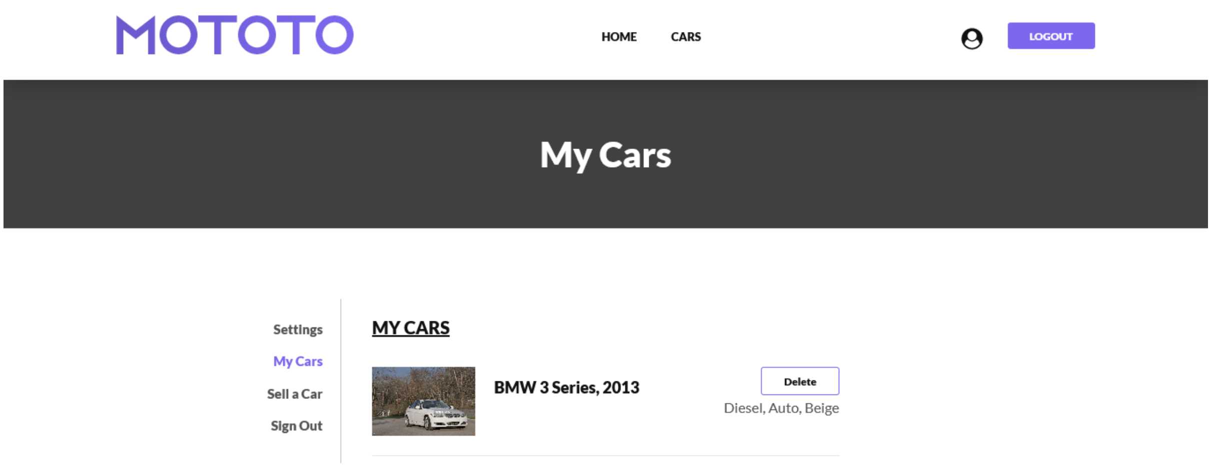
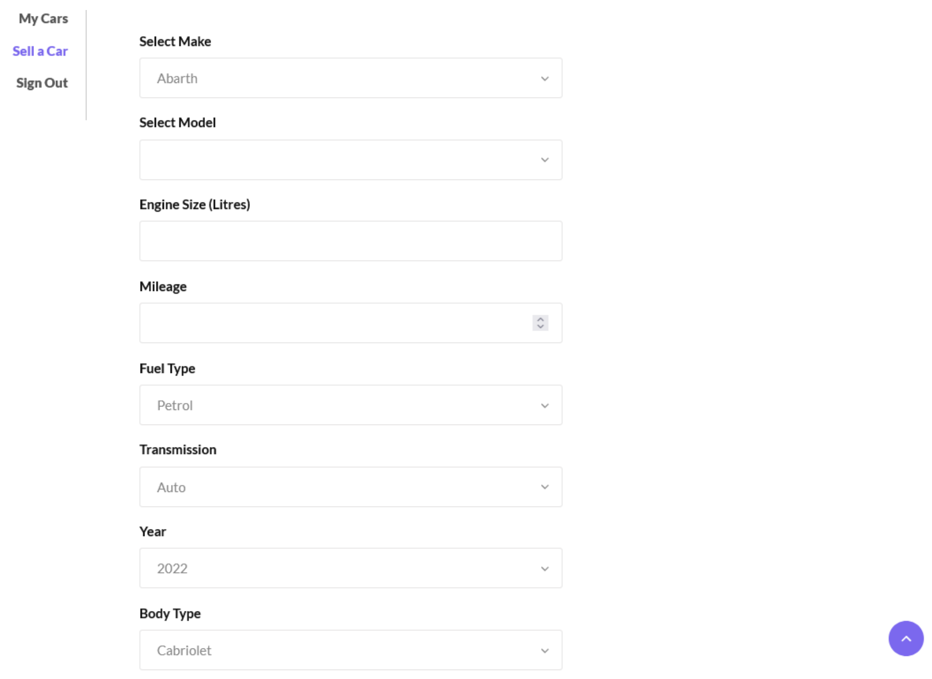


Figure 19

Sell a Car

When clicking on “Sell a Car” (Figure 20) we are greeted with a form. Here we can select the make, model, enter the engine size, mileage, select the fuel type and so on. This list continues furthermore to the most important part which is the features.

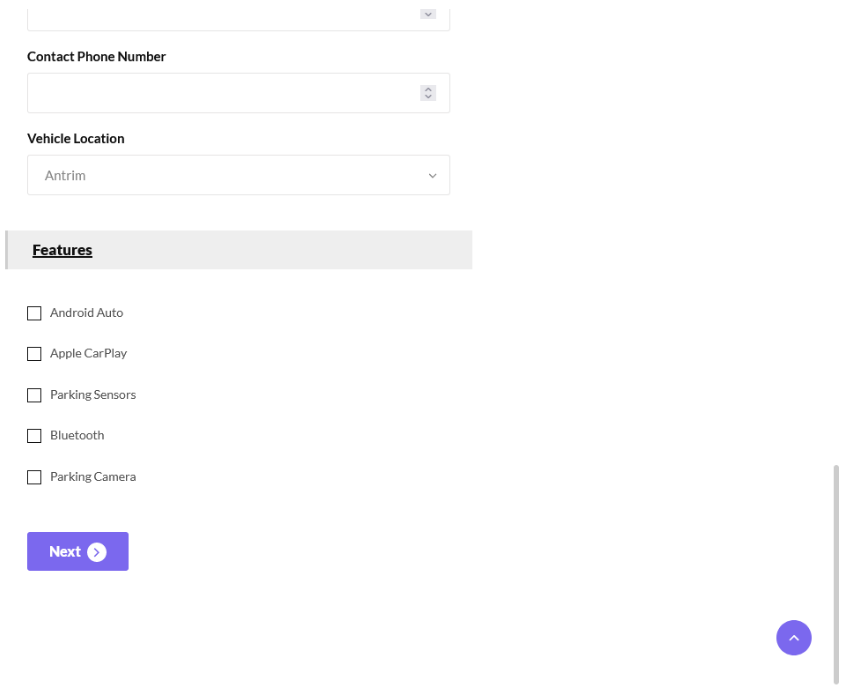


Select Make	Abarth
Select Model	
Engine Size (Litres)	
Mileage	
Fuel Type	Petrol
Transmission	Auto
Year	2022
Body Type	Cabriolet

Figure 20

Features

Here is the features section (Figure 21) where the seller must select these features if they are present in the car they are selling. As mentioned earlier, these features may be extended if necessary to contain even more information such as A/C, heated seats and much more.



Contact Phone Number

Vehicle Location

Antrim

Features

- [ ] Android Auto
- [ ] Apple CarPlay
- [ ] Parking Sensors
- [ ] Bluetooth
- [ ] Parking Camera

Figure 21

2.5. Testing

In terms of testing, I tried numerous ways to manipulate the website myself without any luck and then with the help of testing tools, in particular OWASP ASST which I will go over including results. I found that OWASP ASST covered a lot of the fundamental security assessments regarding the type of website I have created. Finally, I will be providing some frontend user results to get an idea regarding ease of use and issues that may have been found along the way.

OWASP ASST

According to OWASP, it is the only tool that scans for PHP insecurities. It is also able to scan other files such as JavaScript and general SQL queries that are found within PHP itself. It provides information on many different security areas which I will discuss in detail below.

Injection Vulnerabilities
This is directly related to SQL attacks/injections where an SQL query is used on an input field such as the login or other forms that are tied to the database. If the injection is successful, a malicious user may be able to view sensitive information from the database and can even modify it using SQL queries such as update, insert or delete. Therefore, sanitization is implemented to prevent this issue.

Broken Authentication Vulnerabilities
It can be defined in many ways such as credential stuffing where an attacker has a list of usernames and password which they may have got from a breached website. They can use this list to set up an automated system to try logging into a website with some of those credentials. This can also be considered brute force and can typically target weak passwords, meaning that if my website was to have a user under the username and password as "admin" then that leaves a huge vulnerability in the system.

Sensitive Data Exposure Vulnerabilities
This is when communication channels are not secured under HTTPS, especially when it comes to sensitive information such as banking credentials. This can also be quite problematic under GDPR if the website was to be deployed publicly. While not all information may need to be protected, using different encryption forms over communication methods is adequate.

Cross-Site Request Forgery Vulnerabilities
A type of attack that can force an authenticated end user to perform unwanted actions on a website. This is typically a social engineering attack where it can trick a user into sending a malicious request on the website. In that case, it can perform requests such as sending money, changing emails addresses. Nowadays, this is mostly tied down to cookies.

Security Results
When performing a test on my website locally, I was presented with this error (Figure 22). From my investigation, this is a false-positive as the current method used for authentication is session management. There are no credentials stored anywhere else other than the actual database. While two-factor authentication may improve this, it is also suggesting checking if Google reCAPTCHA is implemented properly, although I am not currently using it. While some sort of system such as Google reCAPTCHA may be effective, it is also quite frustrating for the user (at least from my experience.)

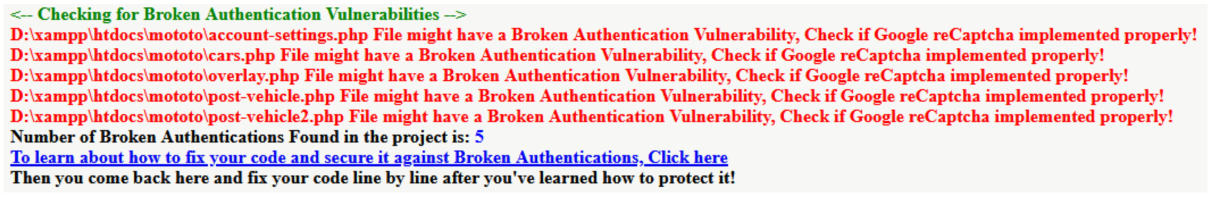

```
<-- Checking for Broken Authentication Vulnerabilities -->
D:\xampp\htdocs\mototo\account-settings.php File might have a Broken Authentication Vulnerability, Check if Google reCaptcha implemented properly!
D:\xampp\htdocs\mototo\cars.php File might have a Broken Authentication Vulnerability, Check if Google reCaptcha implemented properly!
D:\xampp\htdocs\mototo\overlay.php File might have a Broken Authentication Vulnerability, Check if Google reCaptcha implemented properly!
D:\xampp\htdocs\mototo\post-vehicle.php File might have a Broken Authentication Vulnerability, Check if Google reCaptcha implemented properly!
D:\xampp\htdocs\mototo\post-vehicle2.php File might have a Broken Authentication Vulnerability, Check if Google reCaptcha implemented properly!
Number of Broken Authentications Found in the project is: 5
To learn about how to fix your code and secure it against Broken Authentications, Click here
Then you come back here and fix your code line by line after you've learned how to protect it!
```

Figure 22

I was also presented with another vulnerability involving Cross Site Request Forgery Vulnerabilities (Figure 23). Initially, I had implemented this system when focusing on security but when performing the test, it had told me otherwise. After looking into this furthermore, there was something I overlooked involving variables of the token within forms, more specifically fields. While I had the initialization part covered, meaning that the token was generating, it had nowhere to compare that token to for each field of the form. A simple fix to this issue was to add a variable to each field so when the form is submitted, each field is compared to a unique token. Luckily, this does not affect performance in any way.

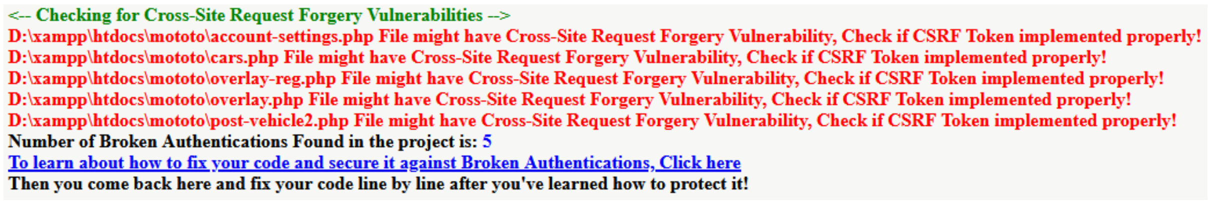

```
<-- Checking for Cross-Site Request Forgery Vulnerabilities -->
D:\xampp\htdocs\mototo\account-settings.php File might have Cross-Site Request Forgery Vulnerability, Check if CSRF Token implemented properly!
D:\xampp\htdocs\mototo\cars.php File might have Cross-Site Request Forgery Vulnerability, Check if CSRF Token implemented properly!
D:\xampp\htdocs\mototo\overlay-reg.php File might have Cross-Site Request Forgery Vulnerability, Check if CSRF Token implemented properly!
D:\xampp\htdocs\mototo\overlay.php File might have Cross-Site Request Forgery Vulnerability, Check if CSRF Token implemented properly!
D:\xampp\htdocs\mototo\post-vehicle2.php File might have Cross-Site Request Forgery Vulnerability, Check if CSRF Token implemented properly!
Number of Broken Authentications Found in the project is: 5
To learn about how to fix your code and secure it against Broken Authentications, Click here
Then you come back here and fix your code line by line after you've learned how to protect it!
```

Figure 23

Frontend User Results

For the frontend users, I decided to let my course mates try the website as-is and allow them to create an account, login and then put a car up for sale as these are the main functions of the website. All tests were performed on a computer.

User 1 – Luke Sheehan

User 2 – Wiktor Wolsza

User 3 – Dovydas Girskis

Register and Login

User	Time	Assistance	Comments
User 1	01:22	None	Took a little while to find the register button, should be clearly defined.
User 2	01:13	None	Agrees with User 1.
User 3	00:43	N/A	Agrees with User 1.

After having a conversation and taking the comments into consideration, I would have to agree that the register button could be a bit clearer rather than having a small piece of text with a redirection. Since I am used to it, I did not think it could pose a problem for the frontend users' ease of use. With this feedback in mind, changes have been made.

Selling a Car

User	Time	Assistance	Comments
User 1	01:57	None	Relatively straightforward, easy to use and intuitive.
User 2	02:03	None	A lot of car makes and models. Maybe too much, but overall, very easy to use.
User 3	01:44	N/A	Came across a mistype which was fixed immediately.

Overall, very happy with the results, no issues were found (only on my end) and that was exactly what I wanted to hear. I am glad that it was easy and straightforward to use as that was my intention. Although I can agree with User 2 regarding having too many makes and models, I addressed this at the very beginning in the database design. The idea was to test the system as much as possible and have everything there rather than missing something and having to put it in later.

2.6. Evaluation

Overall, having the opportunity to test the website using OWASP ASST and receiving input from frontend users has drawn a clear line for the project and what changes must be made regarding security and overall ease of use. In terms of testing with OWASP ASST, while there was a great number of vulnerabilities found, I have managed to tackle the most important ones.

This doesn’t mean the website is currently free of all vulnerabilities as there are some improvements that can be made, especially if the website was to be hosted online. I would like to implement some sort of two factor authentication in the future as well as address and clean up some of the SQL queries in a more secure fashion. Nonetheless, with the website in its current state, I would classify it as a low risk in terms of vulnerability.

Regarding user feedback, I was more than happy to make adequate changes to the overall experience. While it was small things, these are generally what make a huge different for a frontend user. My goal for the website was to make it straightforward and easy to use and I think I have achieved that with the addition of said changes. While it may look complete, I would like to add more features down the line, including a more expanded version of the filter when looking for cars.

3.0 Conclusions

At the moment, it wouldn’t be reasonable to say that the website can compete with the likes of DoneDeal where they have set the foot in the industry for many years. It’s the go to platform but we have seen great alternatives such as Facebook Marketplace. The biggest disadvantage to Mototo is the filter, which is currently limited to the car make, model and features. This is something that can be expanded in the future and can be made fully operational but unfortunately due to the complexity it did not make it. One advantage against competitors would be the features section which was a big part of the project.

On the other hand, regarding security, while there were some complications along the way, I managed to address most of them. As the database is relatively simple and straightforward, securing it didn’t pose as many challenges, especially when there is not a lot of sensitive information being stored. Nonetheless, I would like to improve on several security aspects such as cleaning up and securing SQL queries, but for the meantime, they are secure to a standard.

I feel that Mototo has the potential to grow in terms of a platform where security and user experience is fully addressed. It has the potential to one day be deployed online and compete with the Irish automotive market. Mototo in its current state is very flexible in terms of overall design and security. To achieve this, the next step would have to be migrating to some sort of framework such as Laravel where a lot of this can be addressed.

4.0 Further Development or Research

I think that with additional time, Mototo's next step would be migrating to a framework such as Laravel. From there, Laravel would be a better environment to address some of the security vulnerabilities that still exist. The reason I didn't go for Laravel initially is because I wanted to challenge myself. When I started the project, I had many security-related ideas in mind, but it posed to be a difficult task, especially when so many things need to work together and at the same time. Unfortunately, it took a lot of time to the point where I had to move onto the next task, or I would fall behind.

Something else I considered was implementing an API for the car registration, meaning that when you are posting a car for sale, you enter the car registration, and all necessary information is filled out for you automatically and more importantly accurately. When looking into different APIs available, a lot of them I found were paid but it would address the issue of users posting cars for sale with inaccurate information.

As mentioned before, the filter system was planned to be more extensive, unfortunately that also posed some difficulties along the way and the idea had to be limited down to the true, original plan which was adding a features section and finding the exact car you are looking for. In the future I would like to add a filter system like Mototo's competitors and add even more features to it than it currently has.

Two factor authentication was amongst the initial plans as it is found mostly everywhere nowadays. It's a great way to secure a website from an unauthorized user attempting to login to your account. Unfortunately, when attempting to implement such a security feature, it posed its difficulties and towards the end the idea had to be removed as it was taking too much time to make it work. In the meantime, I have already looked at alternatives if I was to revisit this in the future and I plan to give it a try, nonetheless.

I would also look at refining the different car makes and models that suit the Irish market. There are some variants that are more than likely never going to be used as some of these cars are for the American or Asian market. An example of this is something like Honda which is the European alternative to Acura. In very rare instances would you see an Acura in Ireland for the simple fact that they are too costly to import and have different variations such as emissions.

5.0 References

Bootstrapmade.com. 2021. *Free Bootstrap Themes and Website Templates | BootstrapMade*. [online] Available at: <https://bootstrapmade.com/>

GitHub. 2021. *GitHub - VinceG/Auto-Cars-Makes-And-Models: JSON string, DB Dump, PHP Array of all cars makes and models*. [online] Available at: <https://github.com/VinceG/Auto-Cars-Makes-And-Models>

Owasp.org. 2021. *OWASP ASST*. [online] Available at: <https://owasp.org/ASST/>

Stack Overflow. 2021. *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Available at: <https://stackoverflow.com/>

Tutorialspoint.com. 2021. *PHP & MySQL - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/php/php_and_mysql.htm>

W3schools.com. 2021. *PHP Tutorial*. [online] Available at: <https://www.w3schools.com/php/>

6.0 Appendices

6.1. Project Plan

Chart

Project Proposal

Objectives

The objective of my project is to create a website that will assist users with finding the ideal car they are looking for. The user will be able to go through an extensive list of car parameters and selecting the ones relevant to them. Car features such as: parking sensors, reversing parking cameras and the important ones such as: the car make, model and engine capacity will be integrated. This will significantly narrow down the users' search. While many websites offer a good selection of parameters, they are not as extensive as many people (including myself) would want them to be.

I think that this would be beneficial to many users, especially if they are looking for a new car purely based on features that can assist them when driving. Rather than having to go through a dozen pages to find a car with all the features you want, you can narrow it down with the extensive list of parameters and find exactly what you are looking for.

Background
My friend was looking to get a car for himself, and he said that he would love it if there was an easier way to find the exact one he is after. He mentioned that he wanted a particular make and model of a car which had reversing parking cameras, parking sensors and so on. I decided to help him, and I realised that it is very true in his case. While we did stumble upon a few cars which he liked and had the features he wanted, it took a very long time, nonetheless.

Also, my brother got a new job recently and his job revolves around finding cars. I think that this website would benefit him greatly too. When I had a discussion with him, he said that there is no way to select a car based on the factory package to narrow down his search. What he means by this is factory packages offered by BMW such as M Sport, Audi S Line and so on. Typically, these factory packages include more features than the standard edition of the car.

Despite there being many websites as this, my plan is to compete with similar websites in the Irish market and offer a feature-rich website which will fill in the missing pieces of others, providing a better user experience overall.

Technical Approach
I will start with setting up a framework for the website. While there is a variety of frameworks to choose from, I think that I feel most confident with Bootstrap and I have worked with it extensively. Essentially, starting with a blank canvas, I will start to incorporate the main functionalities of a typical website and additionally tie in some software features.

As cybersecurity is my specialisation, I will draw a lot of focus to integrating many security features and optimizing them as much as possible for the user to have a smooth browsing experience.

Special Resources Required
Access to the internet and a computer or phone.

Project Plan
Having issues downloading Microsoft Project to my device. I am currently looking for a workaround to this.

Technical Details
I will be using a variety of tools such as HTML5, Bootstrap, MySQL, and others. I will be using GitHub to save and keep track of my project.

Evaluation
I will evaluate my website by presenting a multitude of tests. For functionality, I will ask my friend to test it and provide me with some feedback. When integrating new features, I will have to make sure that they can work with each other without any problems, and I must be especially careful when incorporating security features when doing this.

6.2. Reflective Journals

May
This month I decided to redo my plan and look at different approaches to the project. With that in mind I started to research different types of security implementations which would need to be put in place for the different types of functions, especially when it came to SQL queries. I also planned to make some changes to the database so it can be adapted for the selection of makes and models in forms and the filter.

I was able to start working with SQL queries that work with the login and registration system while also implemented and mitigating security issues along the way. I initially started with making my own table for cars and makes on the database but got the idea to expand it further by using a dump. With that in place, I started to work on the form for selecting the car make and model to make sure it worked as intended. I then transferred the same function with small modifications to be adapter for the filter system.

The next step was to implement the filter system and look at different ways to work with the tables in the database. I also wanted to revamp some of the login and registration functions and add additional security to that so I started researching what security implementations can be applied. I decided to leave this step after implementing a fully functional filter system.

June

This month I was able to implement a barebones system for the filter system and adapt it to the database. I initially had some issues with SQL queries and focused on researching alternatives or fixes. At the same time, I prepared some security implementations for the filter system so when I was finally able to fix the issues it, I could easily add them and test their functionality.

With some luck, I was able to create the filter system and moved onto adding some security functions which out of several only one adapted to what I had without issues. Nonetheless, it still mitigated an issue for injection which was the main concern for SQL queries when fetching for information from the database to display it to the user.

The next step was to start looking at the “Sell a Car” function where I had to create forms and then submit them to do the database. The idea was to take some of the functionality from what I currently had because it was already adapted with security features in mind and after some testing it did not present any issues. In general, some things did not go as smooth as intended but I was committed to finish one thing and move onto another. Although, there were instances where I ran into issues, I worked on another aspect of the project and returned to the same problem with an idea in mind which ended up fixing as I gathered more information.

July

This month I was able to add a form for the “Sell a Car” function an additional add a feature to add images. Since there was consistency from many of the functions I had already it did not take as much time as I thought it would, but I did come across a different security issue for forms which was CSRF. Luckily, this was mitigated quickly as PHP provides great functions for these types of vulnerabilities.

I was able to add additional functionalities such as viewing and deleting cars, changing your password and importantly the features. At this point I was having issues with %GET and $POST methods since I had not implemented any sort of sanitization. After doing some research I was able to find a system where these issues can be mitigated with the user of filters. The remaining was to do some testing and do a thorough look through the code and how everything works together. I wanted to make sure that I did not stumble upon any unwanted or unseen complications. Some functionality had to be changes but nonetheless that fixed some additional issues I did not notice before.

The final step was to revamp the login and registration form. I already had this done prior, but I was not happy with the code. I simplified the queries and was then able to use the remaining time to work on the overall feel and design of the website. As I like design, this was one of my favourite parts of the project, but there were times where I stumbled upon some issues along the way and found some alternatives.

Overall, throughout these months, I was able to focus solely on the project and got into a good work ethic. There were times where I thought I would not be able to fix or add some sort of functionality but with many hours in I was able to mitigate most issues or find alternatives. Unfortunately, there were some ideas that did not make it to the project although I plan to revisit the project in the future to challenge myself and perhaps deploy it online when minor issues have been addressed.