Database

VelgBro Web Page

Project Report

By:

- **MarcAntonio Purnama** **2201816480**
- **Amartya Kadarisman Saleh** **2201841210**
- **Marcell Hardja Septian** **2201798906**
- **Figo Aranta** **2201816202**

Table of Content

About Our Team and Roles

Amartya Saleh

As a group leader, my role is to ensure that we as a group have a clear path on the project and implement it maximally. I also make sure that everyone feels comfortable in contributing to the group based on their interest and expertise.

MarcAntonio Purnama

As someone in the team with the experience of web designing, My main responsibility is as the front end developer. Designing and documentation are my secondary role that makes me responsible for the layout and content of every slides and documents.

Figo Aranta

My role is to give subjective response to the team, and as the gitHub organizer to make sure that the files of the code don't clash and also to make sure everything is in order,help building the front-end webpage. And lastly, finalize the program and make sure everything works fine and bug-free.

Marcell Septian

As someone who experiences making databases, my role in the team is to make the database and design them so they could be neat and efficient enough for the front and back end. Furthermore, as I know the best for our database, I create most part of the report. Also, I helped the backend with making some functions which is useful for our project to work perfectly.

Introduction

Problem Statement

The culture of replacing original part into an aftermarket part for cars had been popular since the early 80s. With the advancement of technology especially in web application, it had summoned thousands of websites that does the same thing, selling after parts online.

Our problem is related to people who love cars as we are going to make a website for selling car wheels which is actually one of the main parts that car lovers usually modified. Nowadays, there are lots of online shopping platforms for automotive lovers which are OLX or mobil123 as an example of an online shopping platform for brand new or second hand cars but they don't allow the users to buy wheels with certain criteria which some people actually wanted to modify their cars but cannot find the wanted wheels as they cannot find it anywhere. Mostly, people sell their wheels using instagram but they can't find specific criteria of what they really want.

Due to that problem, our team which consist of a car lover thought of making a specific website which people could search for their specific wheels and users can sell their wheels as well with the criteria that they set so when a user wanted to buy the wheels which is owned by another user. The contact to the seller will be shown so that the buyer can contact the seller and the deal could happen.

Target User

Due to the fact that current online shopping websites are lacking specific criteria for wheels, it is clear that our target users are for the younger or even older people who want to find a specific detail of wheels for their beloved cars, However people with hobby of collecting and selling unique and beautiful wheels could also very much be our potential users.

Real-life Scenario

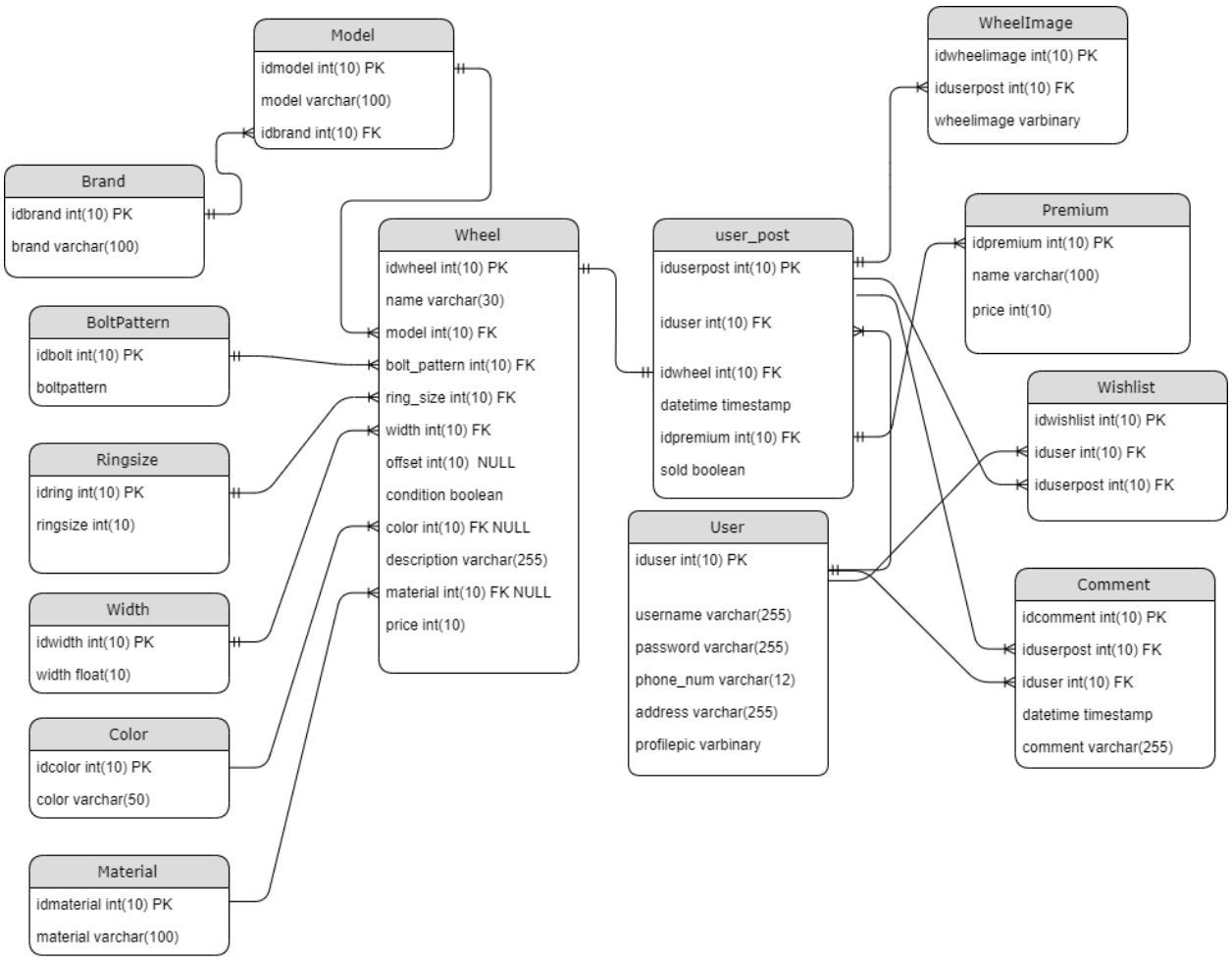
A friend of mine who is obsessed with cars and also likes to modify his car, has always been complaining about how hard it is to find wheels. Up until this day, the way of finding wheels according to him, is always through instagram. And through instagram there is no guarantee of finding the product they want. Potential of scamming is also very likely/high. With our website. Now, we can help hundred of users who want to find wheels with specific details that match with their car. Few benefits are certain. Firstly, Without having to scroll one by one and read the caption carefully, We are 100% certain that it could save a lot of time for the users. Secondly, Selling products becomes much easier, Velgbro website is 100% dedicated for people who interest in finding or even selling wheels. With Velgbro, not only we can help consumers on searching the wheels they want, but also setting a proper place for sellers who just specifically want to sell wheels. Thirdly, Scamming becomes very unlikely, A good seller could get a trusted badge and there is also reviews from users which can help buyer to decide whether this seller is legit or not..

Goals

To help million of users in searching their dream wheels. With our website. We believe that not only we could help users to search the wheels that they always dreamed of. But also to help sellers' desires which, want to settle their business in this industry become possible.

ERD And Database Design

Entity Relationship Diagram



Our entity relationship diagram is shown above and it shows the entities, attributes and relationships. Based on the diagram above, there are 14 entities which have their own attributes. An entity is the name of the table shown above in the ERD and the attribute is the value that they have inside of the table which is shown below the table name. Furthermore, all of the entities above are connected to each other which is shown in the diagram but they have different cardinalities. There are three types of cardinalities which are used and they are one to one relationship, one to many relationships, and many to many relationships. However, there are some relationships that are mandatory which means that there should be a value and it can't be null.

Most of the cardinality used in our diagram is the one to many relationships which are used mostly in the wheel table as it needed that relationship to specify them. For example, the relationship between wheel and bolt pattern, showed that one type of bolt pattern can be used for many wheels which indicates the one to many relationships. Furthermore, the one to one relationship is a relationship in which both attributes should be unique no matter where the foreign key is. For example, the relationship between the user post table and the wheel table which has a relation from the wheel id. The one to one relationship in this situation means that there can only be one wheel id for each post and the id will be unique even though the values of the attributes are the same. Last but not least, the many to many relationships, which is shown from the wishlist, user post, and user table. Inside of the wishlist table there are two one to many relationships which indicate the many to many relationships as there can be many posts which many users can have in their wishlist.

So, there are actually various cardinalities that we used but we don't have a class hierarchy in our ER diagram as our team has discussed and think that we don't need any class hierarchy for our condition because everything is differed by the attributes of the table and there are no subclasses in our diagram which means that we don't have any is-a relation in our ER diagram as it might cause an inefficient database and it might get more complicated if we force to make the class hierarchy in our ER diagram.

Relations

- Wheel((**Primary Key**)idwheel, name, model, bolt_pattern, ring_size, width, offset, condition, color, description, material, price)
    - Foreign Key model references Model(idmodel) On Delete Cascade
    - Foreign Key bolt_pattern references BoltPattern(idbolt) On Delete Cascade
    - Foreign Key ring_size references Ringsize(idring) On Delete Cascade
    - Foreign Key width references Width(idwidth) On Delete Cascade
    - Foreign Key color references Color(idcolor) On Delete No Action
    - Foreign Key material references Material(idmaterial) On Delete No Action

- Model((**Primary Key**) idmodel, model, idbrand)
    - Foreign Key idbrand references Brand(idbrand) On Delete Cascade

- user_post((**Primary Key**) iduserpost, iduser, idwheel, datetime, idpremium, sold)
    - Foreign Key iduser references User(iduser) On Delete Cascade
    - Foreign Key idwheel references Wheel(idwheel) on Delete Cascade
    - Foreign Key idpremium references Premium(idpremium) on Delete Cascade

- Wishlist((**Primary Key**) idwishlist, iduser, iduserpost)
    - Foreign Key iduser references User(iduser) On Delete Cascade
    - Foreign Key iduserpost references user_post(iduserpost) On Delete Cascade

- WheelImage((**Primary Key**)idwheelimage, iduserpost, wheelimage)
    - Foreign Key iduserpost references user_post(iduserpost) On Delete Cascade

- Comment((**Primary Key**)idcomment, iduserpost, iduser, datetime, comment)
    - Foreign Key iduserpost references user_post(iduserpost) On Delete Cascasde
    - Foreign Key iduser references User(iduser) On Delete Cascade

Normalisations

Based on the requirements from the Third Normal Form, our database has followed all of the criteria that are set. Firstly, our data inside of the attribute only have one data type which means an integer value can only consist of numbers not letters. Then, the value that is stored will stay in the same attribute and we used primary key for all of the tables which means all of the data in the table is unique and the order of the data order does not matter because all of them have their own unique primary key that we can check based on the primary key.

Moreover going onto the second normal form, we don't have any partial dependency which means that a column is partially dependent on another column. For example, there are teacher and lesson column in a table. Then, the teacher will depend on the lesson of what they teach which will cause partial dependency but we don't have any partial dependency in our table and nothing actually depends on each other except for the primary key. Going onto the third normal form, it should not have transitive dependency which means a column that depends on another column other than the primary key itself. As it will reduce the amount of data that will be duplicated if we remove the transitive dependency.

All in all, our group have checked up to the third normal form and currently there are no partial and transitive dependency which occurs between our database. All of the attributes and relations have been checked and the rules of the normal forms have been followed.

Relations Sample queries

Search

SELECT `store_post`.`id`, `store_post`.`user_id`, `store_post`.`datetime`, `store_post`.`premium_id`, `store_post`.`sold`, `store_post`.`slug` FROM store_post INNER JOIN products_wheel ON (`store_post`.`id` = `products_wheel`.`post_id`) INNER JOIN products_ringsize ON (`products_wheel`.`ring_size_id` = `products_ringsize`.`id`) INNER JOIN products_width ON (`products_wheel`.`width_id` = `products_width`.`id`) INNER JOIN products_boltpattern ON (`products_wheel`.`bolt_pattern_id` = `products_boltpattern`.`id`) INNER JOIN products_model ON (`products_wheel`.`model_id` = `products_model`.`id`) INNER JOIN products_brand ON (`products_model`.`brand_id` = `products_brand`.`id`) WHERE (`products_ringsize`.`ring_size` LIKE %16% AND `products_width`.`width` LIKE %10% AND `products_boltpattern`.`bolt_pattern` LIKE %5x100% AND `products_brand`.`brand` LIKE %enkei% AND `products_model`.`model` LIKE %RPF-1%)

Add Post

{'sql': 'SELECT `store_premium`.`id`, `

store_premium`.`name`, `store_premium`.`price` FROM `store_premium` WHERE `store_premium`.`id` = 1 LIMIT 21', 'time': '0.000'}, {'sql': 'SELECT (1) AS `a` FROM `store_premium` WHERE `store_premium`.`id` = 1 LIM

IT 1', 'time': '0.000'}, {'sql': 'SELECT `products_model`.`id`, `products_model`.`model`, `products_model`.`brand_id` FROM `products_model` WHERE `products_model`.`id` = 1 LIMIT 21', 'time': '0.000'}, {'sql': '

SELECT `products_ringsize`.`id`, `products_ringsize`.`ring_size` FROM `products_ringsize` WHERE `products_ringsize`.`id` = 1 LIMIT 21', 'time': '0.000'}, {'sql': 'SELECT `products_width`.`id`, `products_width`.

`width` FROM `products_width` WHERE `products_width`.`id` = 1 LIMIT 21', 'time': '0.000'}, {'sql': 'SELECT `products_boltpattern`.`id`, `products_boltpattern`.`bolt_pattern` FROM `products_boltpattern` WHERE ` products_boltpattern`.`id` = 1 LIMIT 21', 'time': '0.000'}, {'sql': 'SELECT (1) AS `a` FROM `products_model` WHERE `products_model`.`id` = 1 LIMIT 1', 'time': '0.000'}, {'sql': 'SELECT (1) AS `a` FROM `products_

ringsize` WHERE `products_ringsize`.`id` = 1 LIMIT 1', 'time': '0.000'}, {'sql': 'SELECT (1) AS `a` FROM `products_width` WHERE `products_width`.`id` = 1 LIMIT 1', 'time': '0.000'}, {'sql': 'SELECT (1) AS `a` F

ROM `products_boltpattern` WHERE `products_boltpattern`.`id` = 1 LIMIT 1', 'time': '0.000'}, {'sql': "SELECT `store_premium`.`id`, `store_premium`.`name`, `store_premium`.`price` FROM `store_premium` WHERE `sto

re_premium`.`name` = 'basic' ORDER BY `store_premium`.`id` ASC LIMIT 1", 'time': '0.000'}, {'sql': "SELECT `store_post`.`id`, `store_post`.`user_id`, `store_post`.`datetime`, `store_post`.`premium_id`, `store_p

ost`.`sold`, `store_post`.`slug` FROM `store_post` WHERE `store_post`.`slug` = 'Xp5UTW9LaEXakhwjuPLDUQIGBUpXo0kh47lHMzLF8y1FfyjAAvlqf1UCldcK2X5gS5ZmbsjvlARXTiucScHCg5NI8xNNjvx7aXbN'", 'time': '0.000'}, {'sql':

"INSERT INTO `store_post` (`user_id`, `datetime`, `premium_id`, `sold`, `slug`) VALUES (1, '2020-01-07 19:08:55.705565', 1, 0, 'Xp5UTW9LaEXakhwjuPLDUQIGBUpXo0kh47lHMzLF8y1FfyjAAvlqf1UCldcK2X5gS5ZmbsjvlARXTiucSc

HCg5NI8xNNjvx7aXbN')", 'time': '0.005'}, {'sql': "INSERT INTO `products_wheel` (`model_id`, `post_id`, `name`, `ring_size_id`, `width_id`, `bolt_pattern_id`, `offset`, `color_id`, `condition`, `material_id`, `p

rice`, `description`) VALUES (1, 8, 'd', 1, 1, 1, NULL, NULL, 'NEW', NULL, 32, '23')", 'time': '0.003'}]

Update Profile

{'sql': 'SELECT `users_profile`.`id`, `

users_profile`.`user_id`, `users_profile`.`profile_picture`, `users_profile`.`phone_number` FROM `users_profile` WHERE `users_profile`.`user_id` = 2 LIMIT 21', 'time': '0.000'}, {'sql': "SELECT (1) AS `a` FROM

`auth_user` WHERE (`auth_user`.`username` = 'kevin' AND NOT (`auth_user`.`id` = 2)) LIMIT 1", 'time': '0.000'}, {'sql': "UPDATE `auth_user` SET `password` = 'pbkdf2_sha256$180000$sYnYYVDuKJgf$07jU5ekEeOOvQWOsJn

VF0IhY9j046RWj6EEaS/i/5SU=', `last_login` = '2020-01-07 19:19:56.754198', `is_superuser` = 0, `username` = 'kevin', `first_name` = 'kevin', `last_name` = 'dimas', `email` = 'dimas@gmail.com', `is_staff` = 0, `i

s_active` = 1, `date_joined` = '2020-01-07 19:19:56.527789' WHERE `auth_user`.`id` = 2", 'time': '0.002'}, {'sql': "UPDATE `users_profile` SET `user_id` = 2, `profile_picture` = 'profile_pictures/tatsumi_nov17_

dino_dalle_carbonare_009.jpg', `phone_number` = '09875644567' WHERE `users_profile`.`id` = 2", 'time': '0.002'}, {'sql': "UPDATE `users_profile` SET `user_id` = 2, `profile_picture` = 'profile_pictures/tatsumi_

nov17_dino_dalle_carbonare_009.jpg', `phone_number` = '09875644567' WHERE `users_profile`.`id` = 2", 'time': '0.002'}]

Show Search Result

{'sql': "SELECT COUNT(*) AS `__count` FROM `store_post` INNER JOIN `pr

oducts_wheel` ON (`store_post`.`id` = `products_wheel`.`post_id`) INNER JOIN `products_ringsize` ON (`products_wheel`.`ring_size_id` = `products_ringsize`.`id`) INNER JOIN `products_width` ON (`products_wheel`.

`width_id` = `products_width`.`id`) INNER JOIN `products_boltpattern` ON (`products_wheel`.`bolt_pattern_id` = `products_boltpattern`.`id`) INNER JOIN `products_model` ON (`products_wheel`.`model_id` = `product

s_model`.`id`) INNER JOIN `products_brand` ON (`products_model`.`brand_id` = `products_brand`.`id`) WHERE (`products_ringsize`.`ring_size` LIKE '%%' AND `products_width`.`width` LIKE '%%' AND `products_boltpatt

ern`.`bolt_pattern` LIKE '%%' AND `products_brand`.`brand` LIKE '%%' AND `products_model`.`model` LIKE '%%')", 'time': '0.001'}]

Delete Post

"SELECT `store_post`.`id`, `store_post`.`user_id`, `store_post

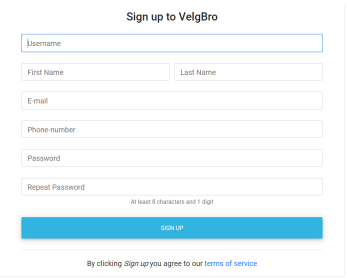
`.`datetime`, `store_post`.`premium_id`, `store_post`.`sold`, `store_post`.`slug` FROM `store_post` WHERE `store_post`.`slug` = 'B4W9ggABgrIivDVsdoMKJebJH1eAueFBRVdJ4CyrhwZiKBpGG7rqJzTtPvWqKKlumDQLM9dOKhtOWixXW

j0UFpSQDy2gd6EqoBid' ORDER BY `store_post`.`id` ASC LIMIT 1", 'time': '0.000'}, {'sql': 'DELETE FROM `store_wheelimage` WHERE `store_wheelimage`.`post_id` IN (9)', 'time': '0.003'}, {'sql': 'DELETE FROM `products_wheel` WHERE `products_wheel`.`post_id` IN (9)', 'time': '0.000'}, {'sql': 'DELETE FROM `users_wishlist` WHERE `users_wishlist`.`wheel_id` IN (9)', 'time': '0.000'}, {'sql': 'DELETE FROM `store_post` WHERE `

store_post`.`id` IN (9)', 'time': '0.001'

User Interface

What is a good database but can't be used by it users? Besides making sure that our database is most efficient, we also focuses on how it will be projected to the screen and making sure that all the buttons are where they need to be and easy to use.

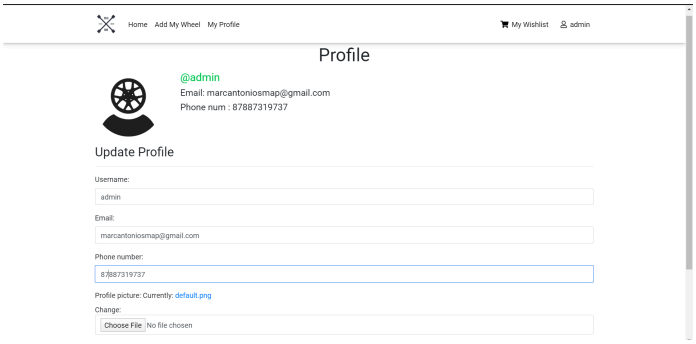


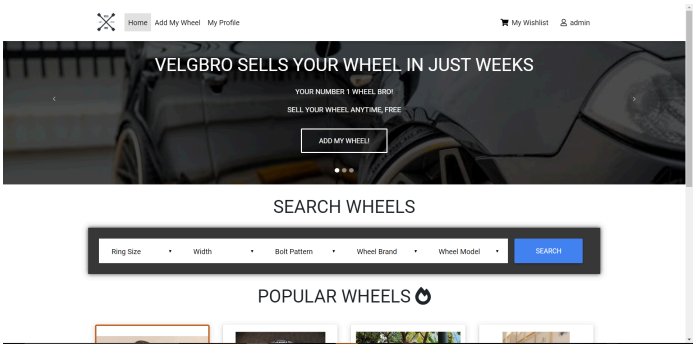
Subtle Design

The use of good contrast enable user to identify sections of the web page.

Label and placeholder

Those help the user to identify the value of each form and the error message will help the user to recover from the error.



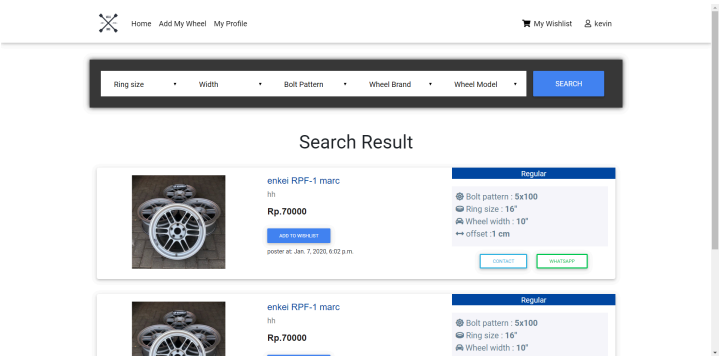


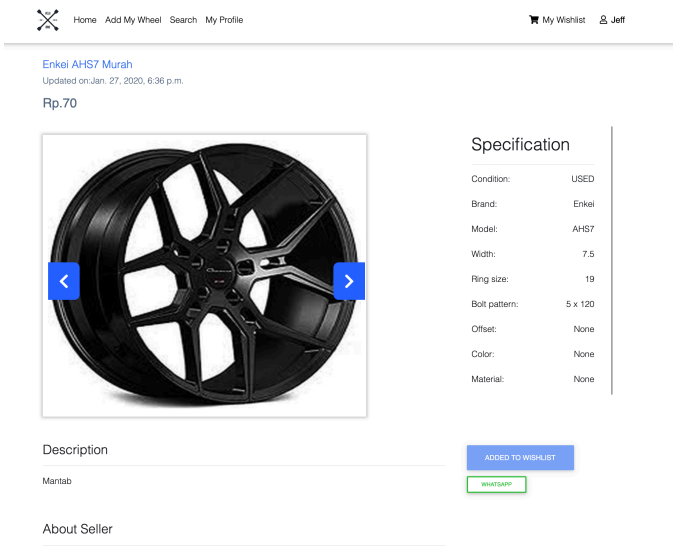
Navigation bar

Navigation bar or navbar helps user to jump from one page to another with ease. Made it sticky so it will stay on top of the page all the time.

Clear product card

The card that represents its product have a clear specification needed for user to click it.



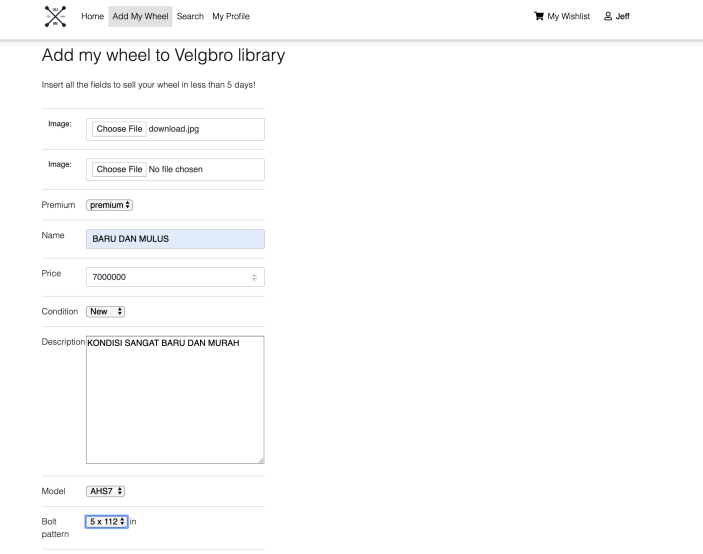


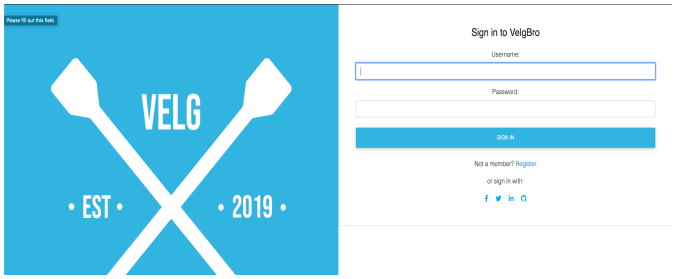
Product Detail

Product Detail View provides product with detailed specification and clear description of the product and information about the seller.

Add Wheel Form

Enables user to fill multiple fields of product's criteria in order to acquire clear specification information of the product.



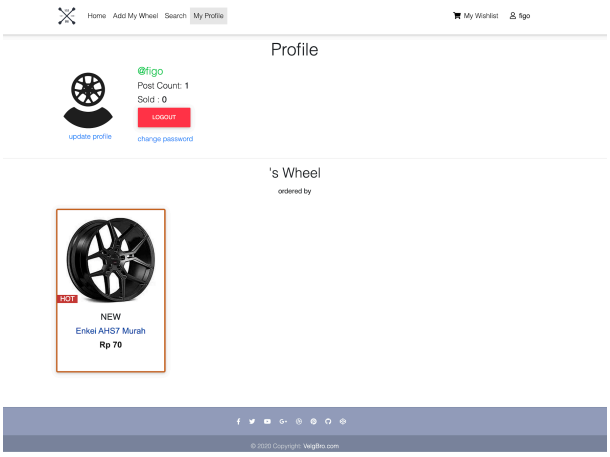


Login Form Page

The login page is needed if the user wish to log into the website as a users. Username and password are required in the login form, In order for the user to sign in.

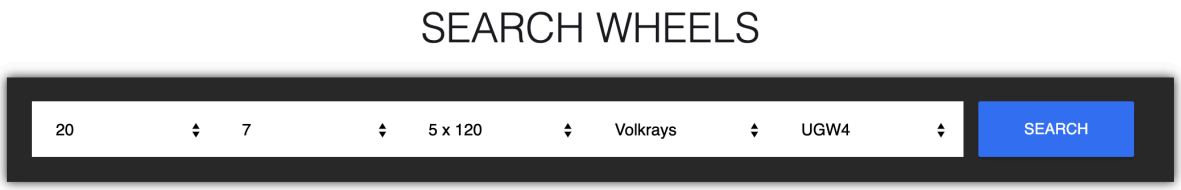
User's Profile Homepage

This Homepage provides user's information about number of wheels the user are currently selling, number of wheels that the user have sold. And also comprehensive information about the wheels.



User's Wishlist list

Enables the user to save the desired products for later. All the saved items that users have looked through, will be stored in wishlist menu.



Specific Detail Search Criteria

The Search bar have already lay out explicit and specific criteria for the user to select according to the user's desired criteria of wheels.

Database Security

The security of the database and the website, will be handled in the backend of the program and the database itself. Since django already has its built-in security features. We took the advantage of exploiting its security protection

Here are some built-in security features from django:

Cross site scripting (XSS) protection

Using Django templates protects you against the majority of XSS attacks

Cross site request forgeries (CSRF) protection

Django has built-in protection against most types of CSRF attacks, providing you have enabled and used it where appropriate. However, as with any mitigation technique, there are limitations. For example, it is possible to disable the CSRF module globally or for particular views.

SQL injection protection

Django's querysets are protected from SQL injection since their queries are constructed using query `parameterization. A query's SQL code is defined separately from the query's parameters. Since parameters may be user-provided and therefore unsafe, they are escaped by the underlying database driver.

Clickjacking protection

Django contains clickjacking protection in the form of the X-Frame-Options middleware which in a supporting browser can prevent a site from being rendered inside a frame. It is possible to disable the protection on a per view basis or to configure the exact header value sent.

Host header validation

Django uses the Host header provided by the client to construct URLs in certain cases. While these values are sanitized to prevent Cross Site Scripting attacks, a fake Host value can be used for Cross-Site Request Forgery, cache poisoning attacks, and poisoning links in emails.

Referrer policy

Browsers use the Referer header as a way to send information to a site about how users got there. By setting a Referrer Policy you can help to protect the privacy of your users, restricting under which circumstances the Referer header is set. See the referrer policy section of the security middleware reference for details.

Session security

Similar to the CSRF limitations requiring a site to be deployed such that untrusted users don’t have access to any subdomains.

Program Manual

Here are the steps to run VelgBro on your localhost server.

Mac Operating System

1. Install pip with this following command in your terminal

```
$ sudo easy_install pip
```

2. Install python environment to isolate the libraries from other outside environment. Use this command to create a new pyenv (python environment).

```
$ python3 -m venv ~/.virtualenvs/djangodev
```

Run the pyenv with this command

```
$ source ~/.virtualenvs/djangodev/bin/activate
```

3. Clone or download zip of VelgBro from Github with these following commands in your terminal

```
$ git clone https://github.com/amartya18/velgbro.git
```

Install required libraries with this command

```
$ pip install -r requirements.txt
```

4. Migrate the tables into the database with these following commands

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Run the website on your localhost server

```
$ python manage.py runserver
```

Windows Operating System

1. Install pip with this following command
   (python version 3 is required, download from this link)

   ```
   $ python get-pip.py
   ```

2. Clone or download zip of VelgBro with these following commands in your terminal

   ```
   $ git clone https://github.com/amartya18/velgbro.git
   ```

   If git is not installed in your computer, consider downloading the zip file through the link.

   Then Install required libraries with this command

   ```
   $ pip install -r requirements.txt
   ```

3. Migrate the tables into the database with these following commands

   ```
   $ python manage.py makemigrations
   $ python manage.py migrate
   ```

   Run the website on your localhost server

   ```
   $ python manage.py runserver
   ```

Conclusion and Recommendation

In conclusion, we have tried our best to make the best from designing the website until finishing the features of the website. Designing the website from scratch, combining the frontend and the backend had been our goal since the beginning of creating this project while making tables of database with relations which were connected to each other with different kind of cardinality that we thought might be useful for the website. Then, making features to search for posts with specific criteria, adding posts and creating accounts, updating posts and username profile, deleting the post had been really tiring for us but it was a fun task as we got some new knowledge to create the website which were not familiar to us that made us need to learn for the syntax first and try to find and modify according to our goals. Lastly, designing the frontend or the visual of the website was one of the most important features as well since most people said that website is all about the visual and designing a web page to be attractive was one of the hardest part for us as well.

Based on our discussion, we think that we can improve more on adding more tables and relationships for the database which will add new features and frontend design if there were more time that we can use to create the website as we tried to maximise our time that we had during the holiday to create the website and learn the syntax which was unfamiliar in the beginning but as we went by the flow to create a website, there were new syntax that we found but still we got to be more knowledgeable and we did our best for the time being.

Related Links

Link for demo video:

https://drive.google.com/file/d/1fE9LO9RvLYUzAefcYgmZx6bvF91dJecR/view?usp=sharing

Link for GIT web:

https://github.com/amartya18/velgbro