MaY 10, 2020

ETERNAL TEAM PROJECT WORK

[WEBSITE LINK](http://www.mumbuy.uz)

MUMBUY

You and your mum’s money. Incredible symbiosis

[GITHUB LINK](https://github.com/iuthub/ip-group-project-eternal-team.git)

Members:

Group leader: Rustamkhonov M. U1810128

Khasanov A. U1810256

Alidjanov S. U1810234

Khursandov S. U1810131

Karimov B. U1810148

Kasimova K. U1810134

# REPORT

## MAIN IDEA

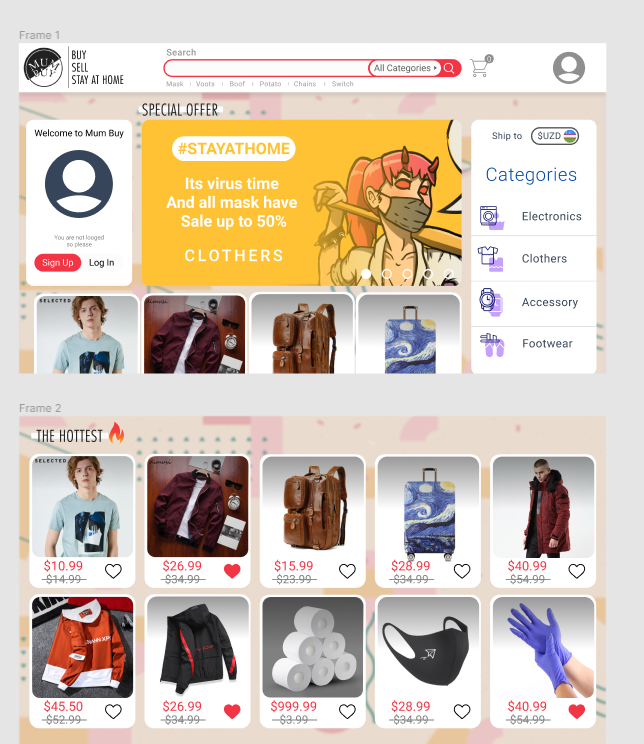
Based on the current situation, when receiving design work, our group immediately took up the idea of ​​an online store. The group approached very creatively in choosing the name of the project. In the end, we chose the name of the project that is associated and is a direct request to our mothers to buy what we want. It was also very in tune with the name of the city. It would be symbolic if this company was based in the city of Mumbai. The division of the work was not difficult, since everything was described in the instructions. Our team was divided into two groups, the first began to register all the functions, registration, work with databases, while the second group set about designing the site.

### PROJECT STATUS

|  | Tasks | Mark |
| --- | --- | --- |
| **Content (weight 20%)** | Landing page, About page, product and users pages. |  |
| **Look and Feel (weight 10%)** | First design was created using figma.com([link](https://www.figma.com/file/4ctQyGTXhRygTerGVu6DGk/Mum-Buy)). Then looking to it layouts, navbars and contents were created. More or less looking like in design above. |  |
| **Forms and Input Validation (weight 10%)** | In the controllers, for each user input implement validations and created forms for sending new added items in form format |  |
| **Database (weight 10%)** | Decided to create database after completing all controllers, and why you will know in explanation below. |  |
| **Server-side Framework - Laravel (weight 10%)** | Project fully made in Laravel Framework as it provides the easiest way of organizing database and auth capabilities right from the box. |  |
| **Client-side Scripting (weight 10%)** | Обязанность |  |
| **Github and Hosting (weight 10%)** | Eskiz.uz buy hosting and domain mumbuy.uz Github Link (https://github.com/iuthub/ip-group-project-eternal-team.git) |  |
| Report | To give a full view of the capture to the reader |  |

## sCREENSHOTS OF MAIN FEATURES

**Design Prototype**



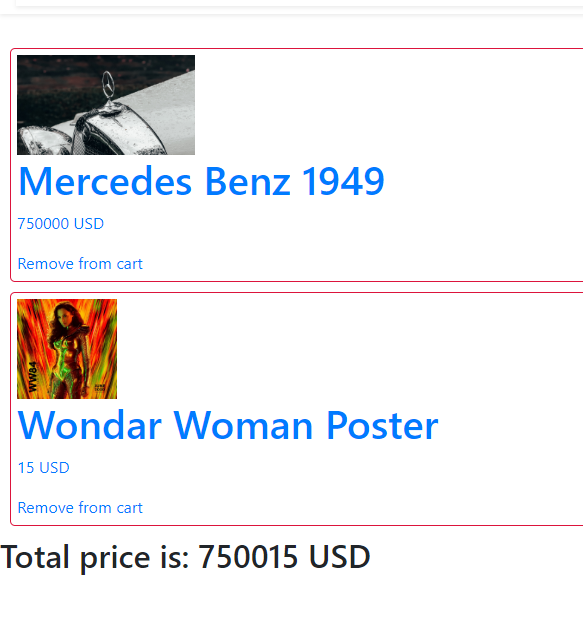
**Firstly, we decided to build something like AliExpress, but in really small scale. However, we took a different direction using this handsome design in our Mum Buy app. This is more or less is trading floor, C2C business model. Everyone can sell, everyone can buy.**

**Simple, fast, light – these principles laid in development of Mum Buy.**

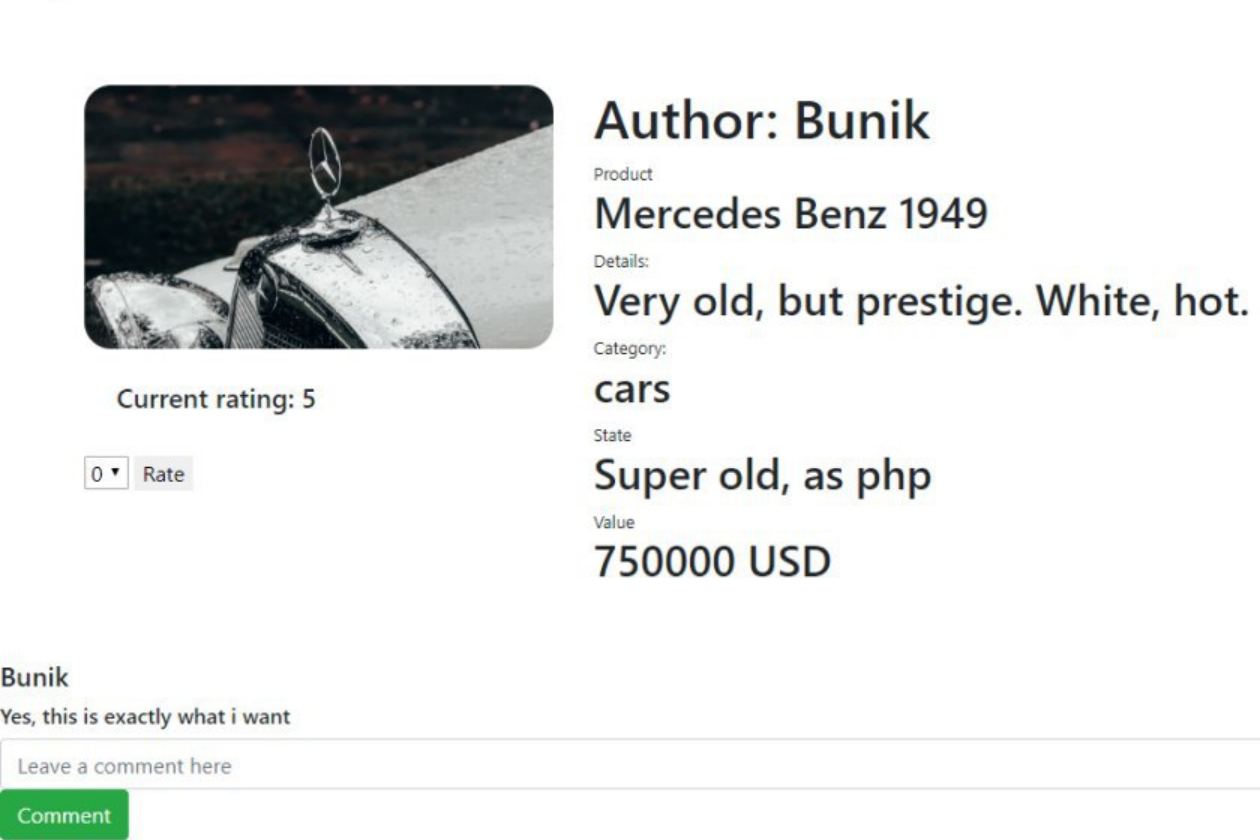
1. **User can scroll down the grid of Items and add it to its cart.**
2. **User can click some buttons in his Dashboard section, pressing the avatar sign on the navbar**



1. **He can manipulate his cart, adding items there and removing them from there. Additionally, cart has to display total amount of shopping.**



1. **User can rate the item, leave a comments there.**



**Adding lot**

Only Authorized users can Add Items to the store, so it prevents from data overflow and shaky data.

Additionally, input from the user goes through “name, price, image” validation. That will prevent from accidental information and will limit the image size for effective media storage.

Image uploading:  
So the function takes the image actual extension and save it to public uploads folder. After what, the real extension with new name(current time in seconds) is passed to Item model in “image” column.

Uses post method.



**Editing item**

Another important point is to make user able editing his items for different purposes.

postEditItem function in ItemController uses the form from the view, validates the input,

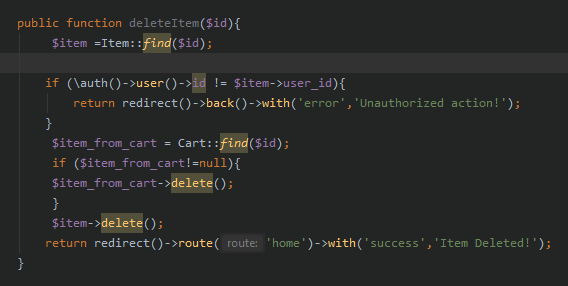
finds the desired item from Item model by the passes as parameter id of the item.

Then controller overrides the specified parameters taken from the view form. For image the same process goes on as in adding logic. In a nutshell, everything is just overridden. Uses post method.



**Deleting item & Search item**

There’s nothing simpler but deleting an item. You need the id of the item, then you need to validate the user, then just delete using ->delete() function from Model. That’s it, redirect with success!)



**New user creation**

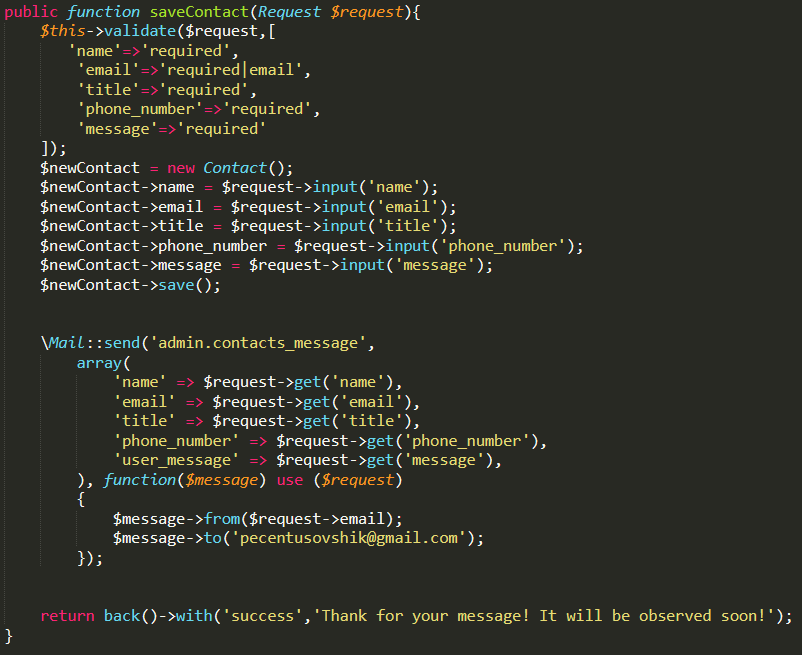
New users are created through standard Laravel make auth template. Login is also operated by logic provided by Laravel. The only thing is we updated form for registration to take cellphone number in case something happens.

***Contact us Page and Logic***

The logic was inspired by this very useful link.

<https://codingdriver.com/create-contact-us-form-in-laravel-with-email-sending-example.html>

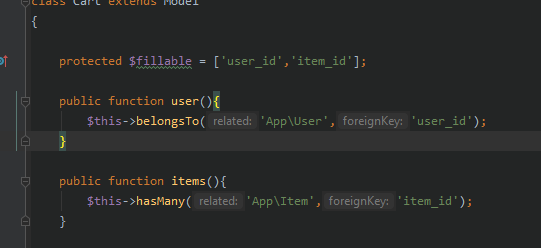
We have created ContactController for realizing messages from users. It uses simple form with multiple input tags, of course, validated. Then it collects and pastes the necessary info to the corresponding html layout(message template) and by mailtrap credentials sends the message by specified email.



**Cart manipulation**

1. **Adding item to cart**

When user wants to add an item to cart, corresponding function is called and by using many-to-many relationship the item and user ids are copied to a new cart table row as one object.

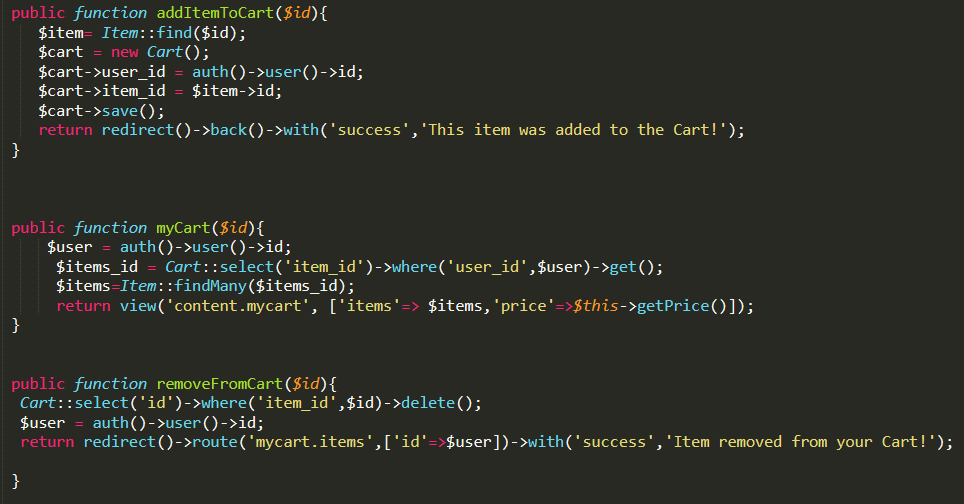


1. **Viewing cart**

Just taking all items for particular user from cart table and displaying them and at the end displays total cost by calling getPrice method.

1. **Removing from cart**

Simply uses delete method and removes corresponding item from cart table in db.



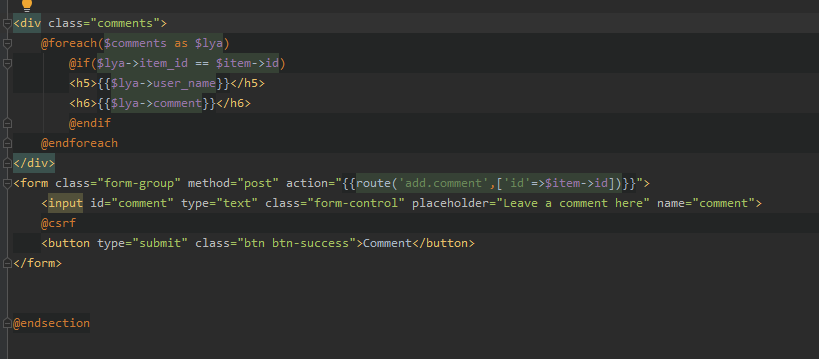
**Comments & Rating**

If the product was really good, you can rate it in scale from 1 to 5. Here used rating table where we have id’s of all products and rating. Whenever the rating is given, the average rating is calculated of mark\_given\_by\_user and displayed in corresponding area inside ItemInfo.blade.php

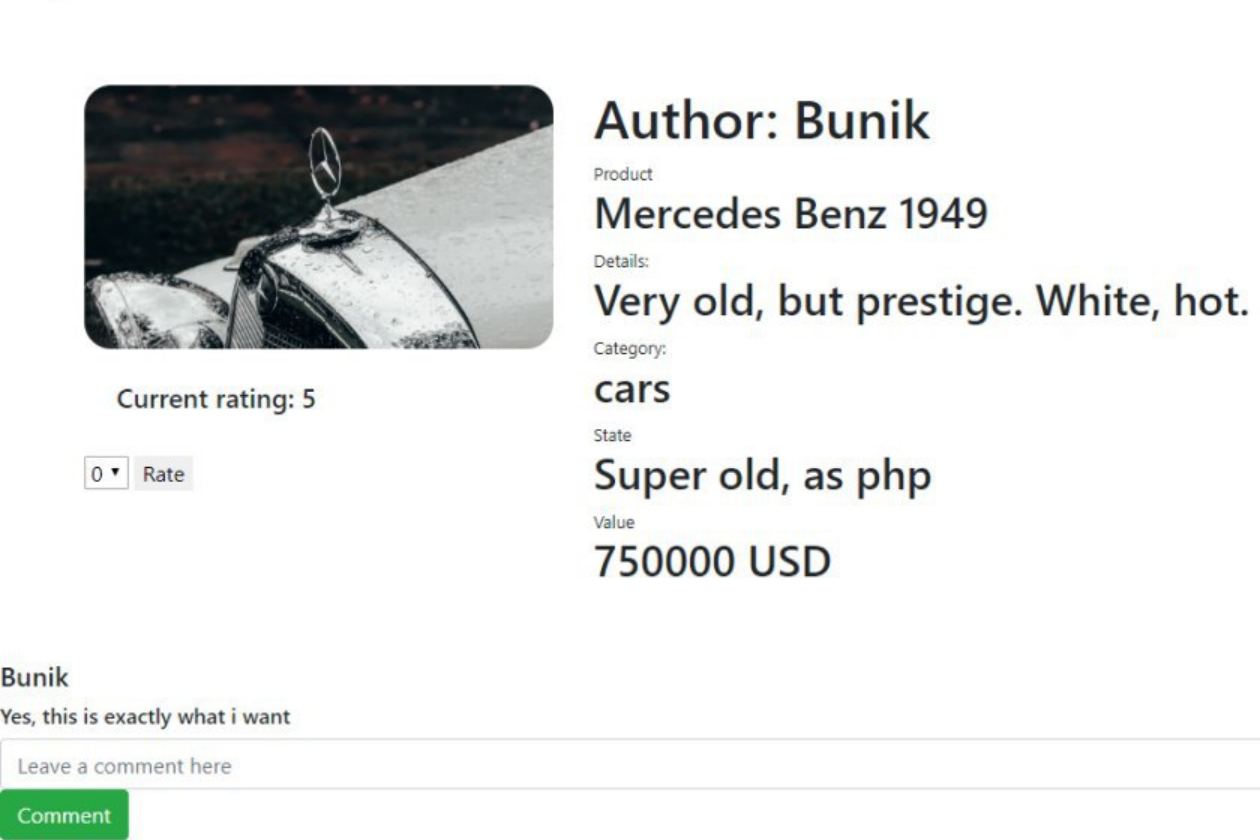


Also its possible to leave comments under each product. For this purpose there is comment method which uses post method and creates one-to-many relationship where one post can have multiple comments. Using the name of the user before comment itself, the name of user is displaced. And comments will be written to comment table. See function and implementation below.





Comments are outputted by some iteration in itemInfo.blade.php by corresponding item id.



Here is the example of comment left by Bunik user.

## sUMMARY OF EACH MEMBER’S CONTRIBUTION

|  | Work done |
| --- | --- |
| Rustamkhonov M. | Server Side Scripting, Report, Database, charismatic leader |
| Khasanov A. | Report, Server Side Scripting |
| Alidjanov S. | Css Html, Client Side Scripting |
| Khursandov S. | Client Side Scripting, Css html |
| Karimov B. | Design, Content |
| Kasimova K. | Server Side Scripting, Css |

## loGIN AND PASSWORDS

| Login | Type | Password |
| --- | --- | --- |
| b.karimov@student.inha.uz | user | 123456789 |
| m.rustamkhonov@student.inha.uz | user | 12346578 |
| admin@gmail.com | admin | Admin123 |

**Some little Irony:**

**We wanted to build super fancy, highly-contrasted app with useful tools, however, our expectations were much higher than our desire and ,might be , cohesion.**