Crafty

Crafty is an e-commerce website that provides a platform for craftspeople to sell and market their items. whereas each craftsman gets his or her dashboard where he or she may post products, keep track of orders, and prepare them in return for a fixed rent per month instead of a high percentage ratio on each product sold.

In addition, the site’s users can view product categories, add products to the cart and proceed for checkout while from the administration side the admin can view sales, users, add products and categories, and can edit, update and delete all the items mentioned previously.

**Public Side:**

* Landing Page
* Shop Page
* Single Product Page
* Cart Page
* Profile Page
* Login and Register Pages
* About Us
* Join Us

**Requirements:**

* users
* The user should register to make the payment.
* Users can log in to the website.
* The user has a profile and can edit it.
* Users can add Products to the cart.
* Users can delete or update their product in the cart
* Users can review products once time
* Users can make a complete checkout process
* Users can View /her Order on the profile page
* Users can join the website as an Owner by ‘Join Us’ Form
* Owner (Craftsmen)
* Owner Can view some statics
* The owner can add products
* The owner can view his/her order
* The owner can update the order status
* Super Admin
* Admin can Add, Edit, Remove and View Users
* Admin can Add, Edit, Remove and View Products
* Admin can Add, Edit, Remove and View Categories
* Admin can Add, Edit, Remove and View Roles
* Admin can Confirm or Decline Comments
* Admin can View and Update Order Status
* Admin can View Order Details

**Technologies:**

* Mockup and Wireframe

1. Figma
2. Mockplus

* Front-End

1. HTML 5.
2. CSS3.
3. Java Script.
4. Bootstrap.

* Back-End

1. Laravel.
2. My SQL.
3. **A picture containing graphical user interface

   Description automatically generatedDatabase schema**

Fig. (1)

After determining the business needs and based on the functionality required. I created the tables and build the relation as below

1. “One to Many” relation

* Each user has one role, and each role has many users
* Each user has many comments, and each comment belongs to one user.
* Each user has many orders, and each order belongs to one user.
* Each order has many order Details, and each order detail belongs to one. order.
* Each product has many comments, and each comment belongs to one product.
* Each category has many products, and each product belongs to one category.
* Each product belongs to one owner, and Each owner has many products.

1. “One to One” Relation

* Each address belongs to one user, and each user has one address.
* Each owner’s info belongs to one user.

**2. Middleware’s**

I created three middleware:

* “Is\_admin” to check if the role is the admin.
* “Is\_owner” to check if the role is the owner.
* “Redirect” to prevent access dashboard through the URL.

**3. Register and Login System (**Authentication)

After using Laravel Authentication for the registration system. I created the “role\_id” column to check the user role through the middleware and redirect the user to the appropriate page as below.

* Role id = 1, redirect to Admin dashboard.
* Role id = 2, redirect to owner dashboard.
* Role id = 3, redirect to the public site.

**4. Admin dashboard**

In the main admin dashboard, the admin can create, read, edit and delete users, roles, categories, and products. the admin can read and update order status, accept and decline comments.

**5. Owner dashboard**

The Owner can access the owner dash bord after his role had been changed by the admin. The owner can add, update, read and delete his products. His or her orders can be seen, and they can be status updated.

**6. Public website**

* Landing page

1. Contain Category Slider.
2. Display New products slider.
3. Show on sale products slider.
4. Service section.
5. Search field.

* Shop page

1. Show product from all categories
2. Sidebar contains filter by category or price

* Single product page

1. Slider for product Image.
2. Product details.
3. Add to cart.
4. Comments section
5. Suggestion product section

* Cart page

1. Display the products that had been added to the cart.
2. Cart total.
3. Update the quantity of the products.
4. Proceed to check out.

* Profile page

1. User information
2. User Orders table.
3. User address information.

* Join us Page

It’s the form for crafters who want to join the website

**Topics required research**

* Display forms validation error message

Refiring to Laravel documentation, we can display errors by using the $errors variable, which is defended by Laravel, or by using the $message variable for specific fields.

* Display flash message after done one the crud operation

Refiring to Laravel documentation, the flash method will flash the current input to the session so that it is available during the user's next request to the application.

* Give a class name to the HTML element based on the URL

It’s done by request()->is(URL), by request method we can get the URL then compare it using is() method.

**Mockup :**

https://www.figma.com/proto/mrJlfaBCBiitYPRSzIPuI8/Master-Piece-Crafty?node-id=1%3A2&scaling=min-zoom&page-id=0%3A1

**Wireframe :**

https://app.mockplus.com/run/rp/VYnZDoWOGnAnG/rxejKhPEZfd4W?cps=expand&rps=expand&nav=1&ha=1&la=0&fc=1&out=0

**Presentation:**

https://www.canva.com/design/DAE6v3-ro3A/eriFz\_\_Jwo9Om2zfDhlTyQ/view?utm\_content=DAE6v3-ro3A&utm\_campaign=designshare&utm\_medium=link2&utm\_source=sharebutton

**Business Model :**

https://next.canvanizer.com/canvas/rYc4JX16qcXBl