

Jhaj: An E-Commerce Website

Supervised By

Tangila Islam Tanni

Lecturer

Department of Computer Science and Engineering

University of Liberal Arts Bangladesh

Submitted By

Md. Safayat Hossain

ID: 153014027

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of
Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF LIBERAL ARTS BANGLADESH

June 2021

Abstract

This project presents a full functional web application for a small sized business enterprise. It will provide convenient support for the parties relating to the business, customers, staff, owners etc. This application will help the small business to sell their product through Website and will help the personnel manage and distribute the product effectively. In addition, related information of product online payment features and customer support are available here. The key intent of this project is to maintain a professional standard to compete in the current market of online business.

Letter for Acceptance

The Project “Jhaj: An E-Commerce Website” is submitted by Md. Safayat Hossain (ID:153014027), to the Department of Computer Science and Engineering, University Of Liberal Arts Bangladesh (ULAB), Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for the Award of the Degree of Bachelor of Science in Computer Science and Engineering on June 2021.

Head of the Department

Supervisor

Dr. Syed Akhter Hossain

Professor and Head of the Department
Department of Computer Science and
Engineering
University of Liberal Arts Bangladesh

Tangila Islam Tanni

Lecturer
Department of Computer Science and
Engineering
University of Liberal Arts Bangladesh

Acknowledgements

I am thankful and expressing my gratefulness to Almighty who offers us divine blessings, patience, mental and psychical strength to complete this project. The progression of this project could not possibly be carried out without the help of my Supervisor who, directly or indirectly responsible for the completion of this work. I am deeply indebted to my project supervisor Tangila Islam Tanni ma'am. Her scholarly guidance, especially for her tolerance with my persistent bothers and unfailing support. She gives me the freedom to pursue aspects of reversible fault tolerant computing which I found interesting and compelling. This helped my project to achieve its desired goals.

I wish to thank the great people of Department CSE at University Of Liberal Arts Bangladesh. A special thank goes to all faculties for their well-disposed instructions and Encouragements.

Finally, I would like to thank my friends and family. Their continued tolerance with my moods and tendency to disappear for weeks at a time gave me a much needed break from the world computing.

Table of Contents

Abstract	II
Letter of Acceptance	III
Acknowledgements	IV
Chapter 1: Introduction	
1.1. Motivation	1
1.2. Key Features	2-3
1.3. Scope	3-4
Chapter 2: Requirement Analysis	
2.1. Functional Requirements	5
2.2. Non-functional Requirements	5
2.3. Data Flow Diagram	6
2.4. Use Case Diagram	7
2.5. Activity Diagram	8
2.6. Chart Diagram	9
2.7. ER Diagram	10
2.8. Software Requirements	11
2.9. Summary	11

Chapter 3: Application Functionalities

3.1.	User Registration and Verification	12
3.2.	Cart Functionalities	13
3.3.	Add to Cart	13
3.4.	Addition, Deduction& Removal On-The-Go	13
3.5.	<i>Product Count and Total Price</i>	13
3.6.	Delete Cart	13
3.7.	User Profile	14
3.8.	Checkout	14
3.9.	Product Details	15
3.10.	Related Products	16
3.10.1	Algorithm	17
3.11.	Customer Support	18
3.12.	Sale Management	18
3.13.	Product Management	18
3.14.	User Management	18
3.15.	Summary	19

Chapter 4: Implementation Plan

4.1.	A Quick Glance at Laravel Framework	20-21
4.2.	Summary	21

Chapter 5: Technical Procedures

5.1	Ease of Use	22
5.2	Search Result Sorting	22-24
5.3	Automatic Sale Deletion	25-26
5.4	Summary	26

Chapter 6: View Optimization

6.1	Using Service Providers	27-28
6.2	Summary	29

Chapter 7: Conclusion

7.1	Learnings	30
7.2	Conclusion	30
7.3	Future Works	31

List of Figures

Figure 2.3: DFD Diagram	6
Figure 2.4: Use Case Diagram	7
Figure 2.5: Activity Diagram	8
Figure 2.6: Chart Diagram	9
Figure 2.7: ER Diagram	10
Figure 3.2: Cart Functionalities	13
Figure 3.7: View of a User's Profile	14
Figure 3.8: Checkout	14
Figure 3.9: Product Details	15
Figure 3.10: Related Products	16
Figure 5.2.3: In Checkout	24
Figure 5.2.4: Order	24
Figure 6.1: Categories	28

Chapter 1

Introduction

An e-commerce website that makes the trade for buyers and sellers in few clicks. This application helps the business personnel easily manages the users and products, apply or remove sales, keep track of orders etc. In addition, Customers can search and buy products through secure transactions online and can see their transaction histories later. It saves times for customers.

1.1 Motivation

The natural spices to enhance our recipe and provide a smell that is difficult to resist. That is the objective of my page and this objective is related with my motivation. Actually I want to help my customer by saving their time through my quick service and I want to give them the opportunity to take authentic flavor.

In recent years, and because of Corona Pandemic e-commerce has been dominating the world of trades and business. Due to the ease-of accessibility of the internet and microcomputer devices along with the simple manner of usability, it has been a sector wheremore and more people are merging every day. We wanted to get a little insight of the practical events and issues related to this sector.

Our goal was to build a professional standard project, which would help us to get an idea about how things work in real world (i.e., software development firms). Working with Laravel has enabled us to become a little familiar to such kind of a workflow.

In short, two of the largest sectors that not only keeps growing fast in different parts the globe but also shows a great deal of promise in Bangladesh's near future as well- was the actual motivation behind our project.

1.2 Key Features

- Menu
- Product Browsing
 - Product List
 - Category List
 - Related Products
- Catalog Management
- User Management
 - **Proper user redirection** after different tasks
 - **Update User information**
- Free from data redundancy
 - Thoroughly used Laravel's ORM (Object Relational Mapping)
 - Framework's **OOP** concept gives extra benefit in the security intact and keeping the data safe.
- Product Browsing
 - Related Products
- **Search** products conveniently
 - Sort by Categories
 - Find products by Category
- Shopping Cart
 - Update Cart
- Checkout
- **User Info** can be added / updated later
- Administrators and Managers can **add Products, Categories**
- Add **Sale** on products
 - Easily add/remove product from Cart
- **Product Checkout** is done safely and securely using stripe.
 - Account authentication is validated
- **Order status** can be monitored by managers/admins afterwards

- Users' **Cart** info is saved in **Order** table inside database
 - Order includes payment_id, user_id etc.
- Related Products
 - Suggest related products

1.3 Scope

Guests: (Unsubscribed users)

- Search products
- View product details
- Add to cart
- Can buy products after registration

Subscribers:

- Search products
- View product details
- Add reviews on products
- Add to cart
 - Increment, decrement products by 1
 - Discard a product from cart
- Delete cart
- Checkout
- Check previous transactions
- Change/Reset password
- Store/Change users' advanced info
 - View self-profile
- Contact staff

Managers:

- Create/update products
- Create/update categories, brands
- Order status checking
- Update product stock
- Create Sales (Sales are set to be deleted automatically at ending date)
 - Delete sales manually
- Assign Sales on products
- Give supports and solution to users

Administrator:

- All of the tasks that a manager can carry out
- Complete power over users (including managers)
- Add/delete users
- Change activation status
- Change user roles etc.

Chapter 2

Requirement Analysis

Requirement Analysis is also termed as SRS (Software Requirements Specification) is the process of finding the requirements for developing a software, i.e., data-dependencies, software- dependencies, time-constraints, feasible features to be applied etc. Usually this includes interviewing personnel from client or buyer side, who provides with the primary expectations of what the application might or should do.

2.1 Functional Requirements

- Users should be able to search for a products
- Users should be able to view a product details
- Users should be able to add or delete products to cart
- Subscribers should be able to view and edit their profile info
- Subscribers should be able to change/reset password
- Subscribers should be able to view transaction histories later
- Products should be able to be set on sale by office staff
- Managers should be able to add, update and delete products
- Managers should be able to add, update and delete brands, categories etc.
- An interactive communication system should be set between managers and subscribers for user support.
- Managers should be able to see and change orders' current statuses
- Admin must be able to carry out all the tasks a manager can do, additionally should have complete power over all the users.

2.2 Non-functional Requirements

- Server-side validation
- Proper routing
- Ease-of-use
- Redundancy free Database

2.3 Data Flow Diagram



Fig. 2.3: DFD Diagram

2.4 Use Case Diagram

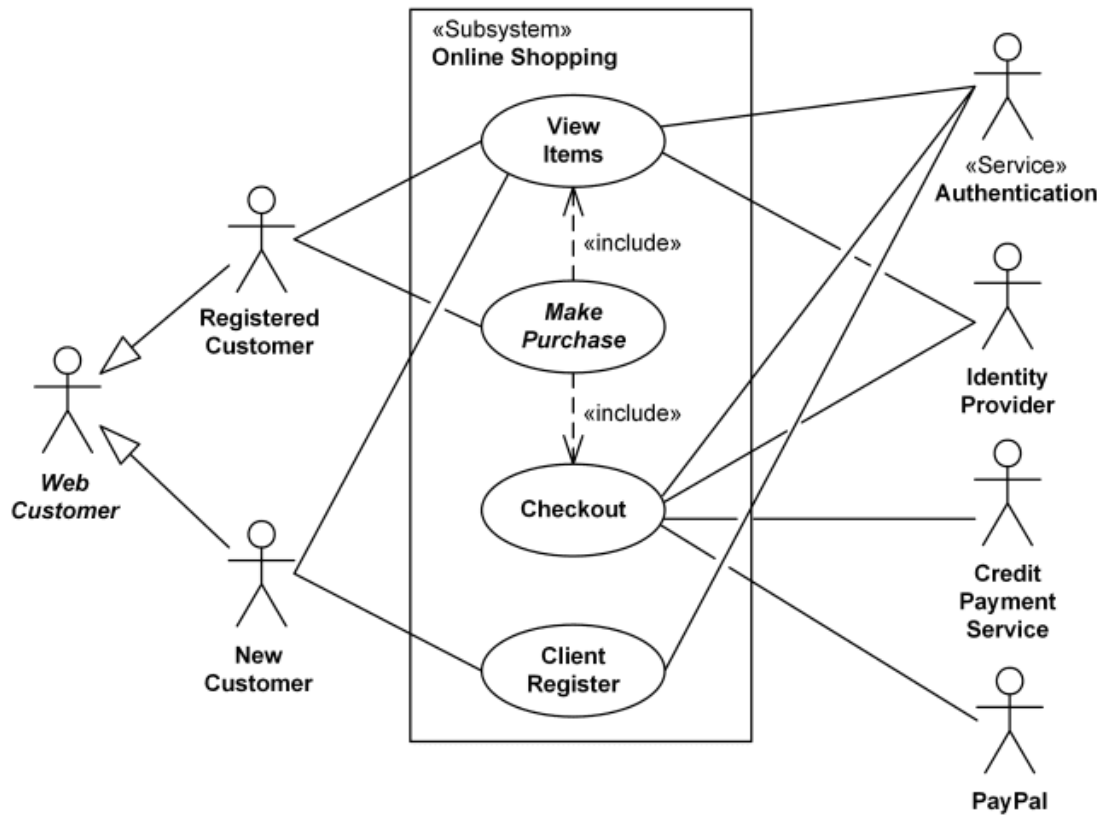


Fig. 2.4: Use Case Diagram

2.5 Activity Diagram

The activity diagram below shows the process of product purchasing of products using the website.

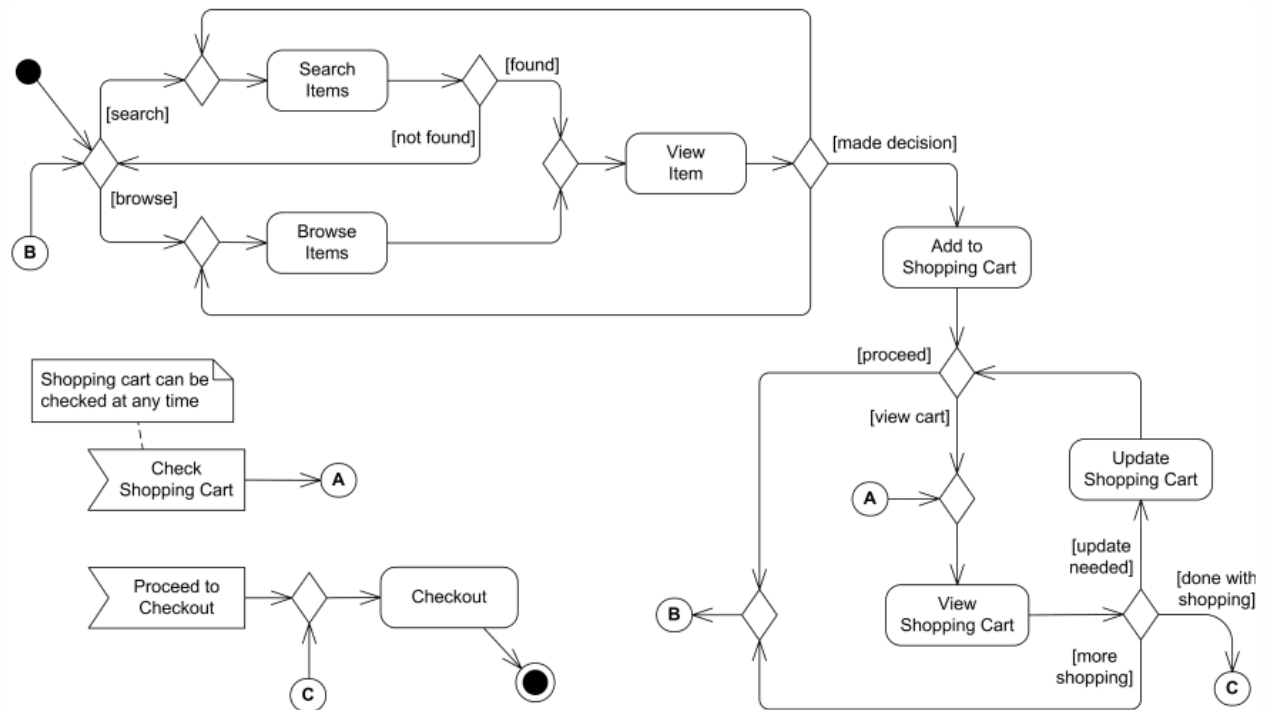


Fig. 2.4: Activity diagram for users purchasing products

2.6 Chart Diagram

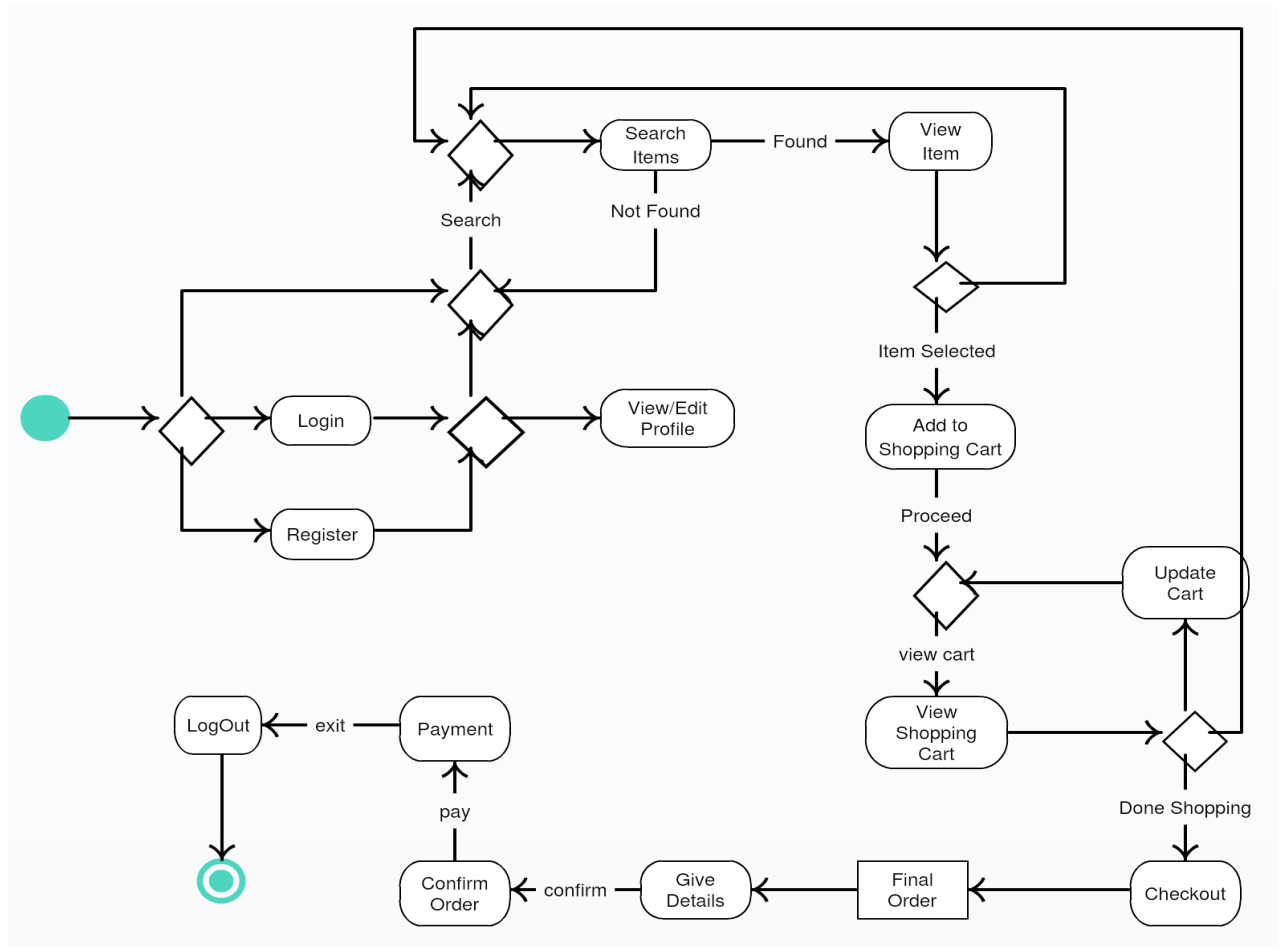


Fig. 2.3: Chart Diagram

2.7 ER Diagram

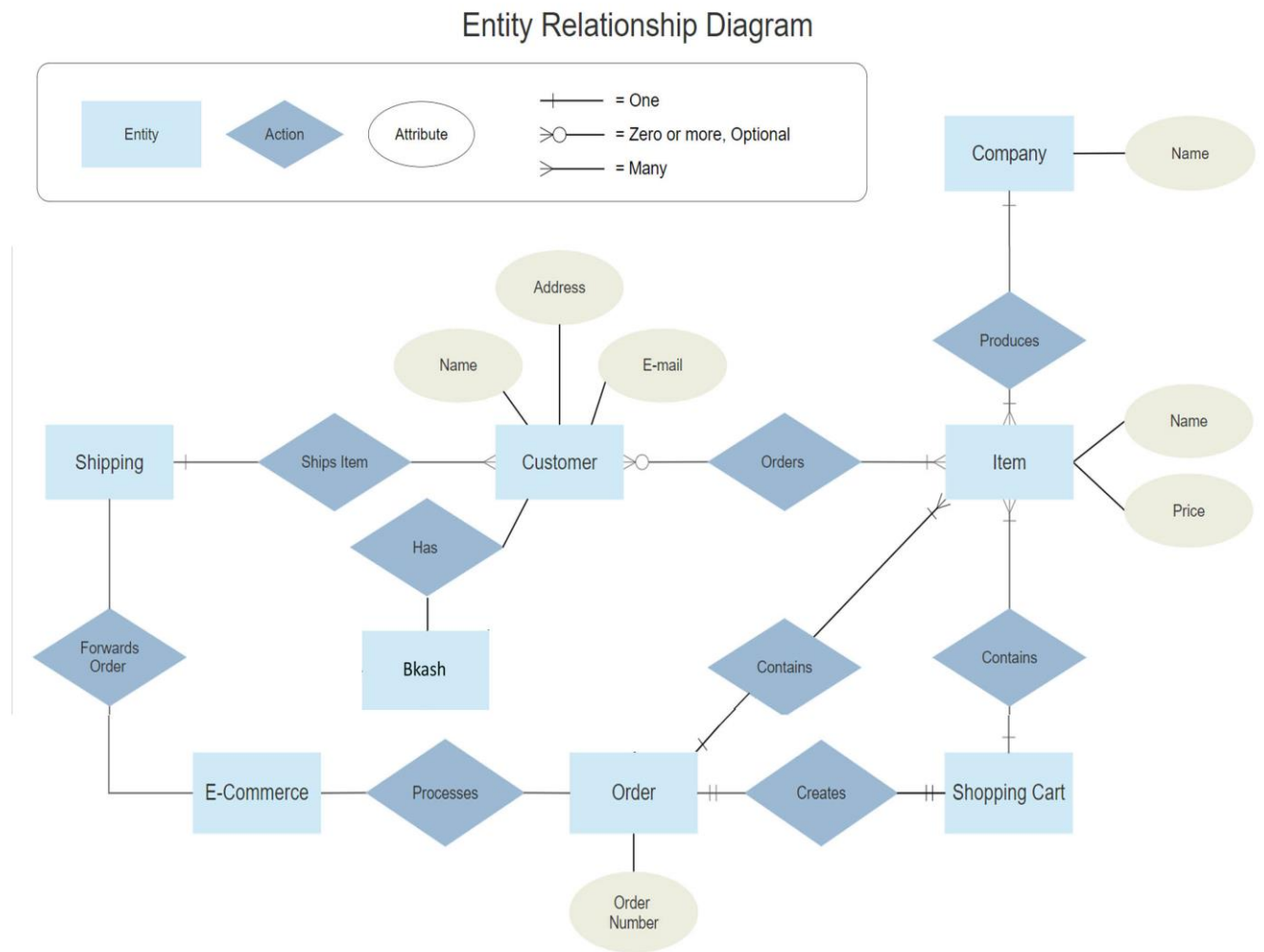


Fig. 2.3: ER Diagram

2.8 Required Software

For establishing the system I need these following software packages:

DBMS	: MySQL (Included in XAMPP)
Server-side Language	: PHP (Included in XAMPP)
Web Server	: Apache (Included in XAMPP)
PHP Dependency Manager	: Composer (Included in XAMPP)
Version Control Software	: github
IDE	: Any Text Editor as per user choice.

2.9 Summary

This chapter discusses the basic structural and supplemental requirements for the application using software requirement analysis, sets the plot for a better understanding of the following chapters.

Chapter 3

Application Functionalities

The application offers various tasks for the users, different user groups carry out different tasks. From searching to purchasing products, from adding sales and removing products- a moderately wide range of tasks are performed by subscribers and office staff. In this section I try to mention the primary functionalities of the web application.

3.1 User Registration and Verification

The web application “Jhaj” will let “guests” browse and search through the product details and add the products they want to purchase to the cart. But to access the cart and advance through the process, they have to login. If the user is not- register on the website by verifying their e-mails.

The registration page will take users’ Name, a unique E-Mail id, and confirmed password initially.

A confirmation e-mail is sent to the user-given e-mail address with verification code.

User is kept inactive (`is_active = 0`), hence unable to carry out subscriber tasks until the e-mail verification is confirmed.

After completing the registration is completed, the user can access the cart, edit their profiles, contact the customer support and advance through the purchase options.

3.2 Cart Functionalities

Jhaj's cart system provides features for users to help product purchasing more interactive.

3.3 Add to Cart

Users add their to-be purchased products to the cart, either from the shop views or from the product details view. They can select their desired quantity to be added to the cart but not more than the stock permits. This stock limit validation is done through multiple stages, which is discussed later in the "Application Implementation" part of this paper.

3.4 Addition, Deduction& Removal On-The-Go

After a user adds a certain number of products to a cart, he/she might come to a decision to add another piece of the product to the cart, or deduct the product by one, or the choicemight even be to remove the product from the cart. This feature is implemented on the cart to increase customer satisfaction.

3.5 Product Count and Total Price

Throughout the whole browsing session of a user the total number of products added to the cart and accumulated price is shown beside the cart's button. This feature can be useful for users to get a better estimation of their purchase status.

3.6 Delete Cart

Customers can delete product, update product quantity cart form Cart Page. They can go back to the shop page and can check out for order from the cart.

Product	Weight	Price	Quantity	Subtotal	
 বুট ডাল	1 Kg	100.00 Tk	1	100 Tk	 
< Continue Shopping				Total: 100 Tk	Chechout >

Fig. 3.2: Cart Functionalities

3.7 User Profile

Before placing an order to the checkout stage, the user's profile containing related advanced info must be submitted. The idea is to make sure of the availability of shipping address, phone number and such user contact-info before hand, so the purchased products does not go to a wrong address. The user profile info can updated and viewed from going to the “Edit Contact Info” button.

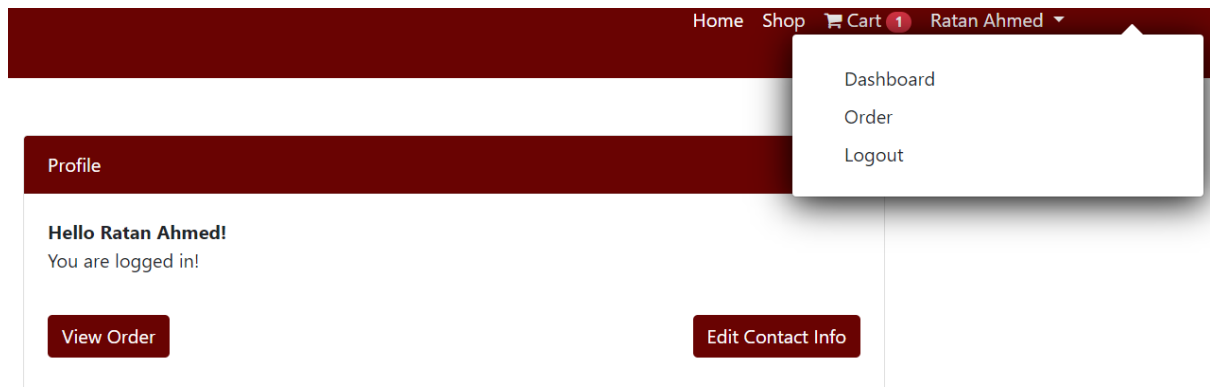


Fig. 3.7: View of a User's Profile

3.8 Checkout

Users can checkout only by registering or login. The validity and authentication is managed by **Stripe**, accredited API for handling payment procedures globally, and supporting user with by providing convenience.

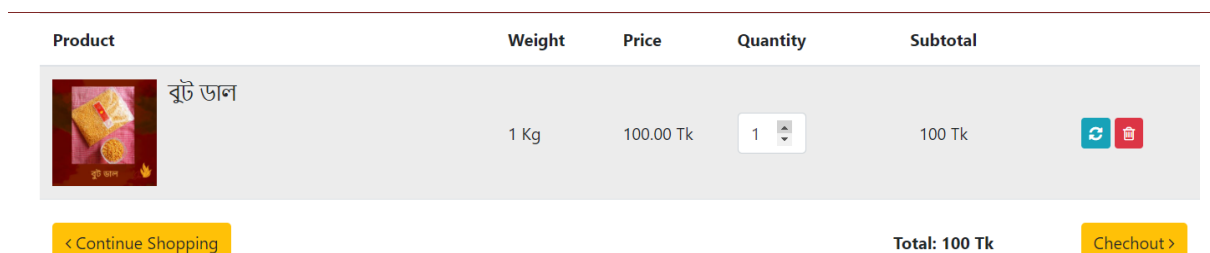


Fig. 3.8: Checkout

3.9 Product Details

Product detail page shows a products detailed information, price, category, related products etc. The users add products to the cart or from the same. Product details page accommodates a number of features in a packed space.



জিরা বাটা

বিশুদ্ধ ও পরিষ্কার জিরা দিয়ে প্রস্তুত এবং অর্ডার পাওয়ার পর প্রস্তুত করা হয়। জিরা পানিতে ভিজিয়ে রাখা হয় এবং পরবর্তীতে এটি বাটা মশলায় রূপান্তরিত করা হয়। বাটা মশলাটি ফ্রিজে রাখতে হবে।

উপাদান : বিশুদ্ধ শুকনো আদা।

Quantity	Price
100gm	150.00

Add to cart

Fig. 3.9: Product Details

3.10 Related Products

This part shows the related products to the current one in a sense that the users could potentially be interested to purchase them. (same category products) These products suggestions are derived and fetched from accessing the cart data from database.

RELATED PRODUCTS

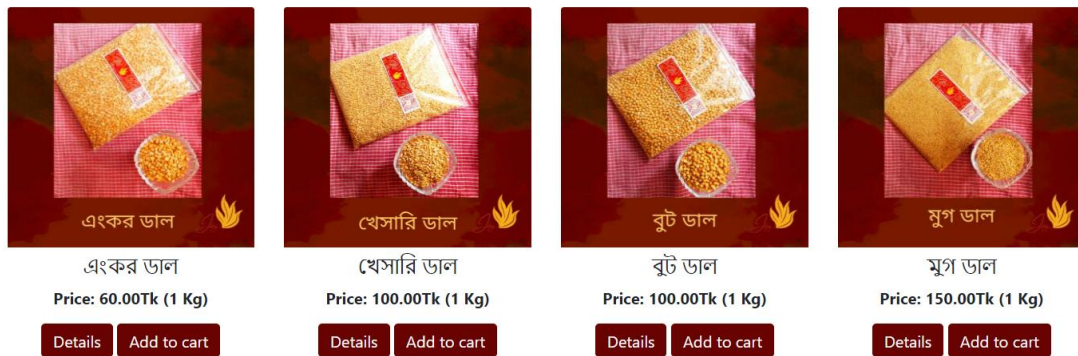


Fig. 3.10: Related Products

3.10.1 Algorithm

```
Foreach cart in carts
  If current_product_category exists in cart
    store the other ids in an array
  End foreach
Foreach cart in carts
  If current_product_category exists in cart
    Foreach related_ category
      If related_ category exist in cart
        Store related_ categorys in array
      End if
    End foreach
  End if
End foreach
Count number of related_ category occurrences
Sort from high to low
```

3.11 Sales Management

Sales can be set on products by Jhaj staff (admin, managers). Product removal from sales is also managed in the same way. The most useful features about sales is that **sales are ended** at the exact ending date **automatically**.

3.12 Product Management

- Products can be added, deleted and edited by admins and managers
- Categories and Brands are to be created by the staff
- An useful feature is **addition of new brands on-the-go** (non-redundant approach)

3.13 User Management

Admins can add, edit and activate users manually, and assign or change roles of existing users. User data manipulation is not accessible to the managers.



Fig. 3.11: User Management

3.14 Summary

An overview of mentionable key functionalities of the application has been shown in the chapter. For better understanding, figures and algorithms have been shown in the chapter also. Following chapters discuss the implementation and technical procedures of these functionalities.

Chapter 4

Implementation Plan

Before I move on to the next section where the technical implementations of our project is discussed, I would like to discuss some of the mandatory to know key-terms first.

Both Laravel, and PHP are products of open-source projects. Hundreds of skillful contributors are constantly working on both of them to make them more competent, more secure. In that sense, these two make the perfect, powerful duo to be used to make powerful, robust web applications.

We used this open-source duo to build our project along with HTML, CSS & Javascript. We used **bootstrap 3**, a web-styling framework for front-end designing.

Apache serves as the local-host, **MySQL** is the **DB Management System**. **PHP** is the server-side language. A package manager **Node.js** is used to manage gulp elixir.

4.1 A Quick Glance at Laravel Framework

Laravel is an open-source new generation web framework that uses a vast number of advanced features to make web application development secure, and time-saving for developers. Implementation of **MVC** pattern gives the developers a level ground to start their app development in a guided sense.

4.1.1 Key Terms:

- **MVC:** MVC is abbreviation for **Model-View-Controller** pattern in software development. Laravel framework uses this highly efficient pattern in the process of web development.
- **ORM:** ORM is short form of **Object Relational Mapping**. ORM Helps make an entity with unsupported data-type into an “Object”, which can be later manipulated using **OOP** (Object Oriented Programming) concept.
- **Eloquent:** Laravel’s eloquent simplifies the task of information retrieval by using “Models” to access the respective data relations. Hence simple functions can be implemented on the
- **Migrations:** Migrations are used for creating, deleting, and modifying databases and relations. The key convenience of migrations is that, the databases can be rolled back and forth of its state (e.g., remove a column that has just been inserted, rename a column back to its previous name) during development process at ease. Large chunks of data can be “seeded” into the database with significantly less effort.
- **Middleware:** Middlewares work as passages through which view requests go through to the controller to be manipulated. In a sense a “Middleware” controls the view- controller communication.
- **Service Provider:** Service providers are the start-up entities those register event listeners, middleware, routes and other variable at boot level.
- **.blade:** Naming convention of Laravel views follows adding a “.blade.php” extension (e.g., index.blade.php) at the end of the view filename.
- **Route:** All the existing routes in the web application must be declared in the “web.php” file inside the “Route” folder.

4.2 Summary

Planning the implementation procedures and maintaining cooperation is a key task to be sorted out before going into project development. Proper cooperativeness leads to smooth mergers and easier issue-solving for a project.

Chapter 5

Technical Procedures

For the technical part of this project the issues I considered with utmost priority were-

- i. Ease of use
- ii. Secured server-side validation has used
- iii. a Redundancy free database

5.1 Ease of Use

Maintaining usability of an application may seem negligible if not much thought is given into the matter, but if I try to look at it carefully, it becomes pretty clear that, this is one of the, if not, the only significant quality an application should have. It is because of the fact that, applications are built for the users particularly. If they do not feel comfortable using the application, then the sole purpose of building one fails. We took careful notes, and tried to build the application as much easy to use as I could within our extent.

Few features those were included to support our goal-

- Search result sorting
- Consistent fail-safe pop-ups
- Effective redirection to proper routes

Fail-safe manner of the interactions makes sure that the users navigate through their browsing sessions inherently.

5.1.1 Search Result Sorting

Search results are primarily shown to the user in an alphabetical order. But those can be sorted later according to the users' needs. Sorting by Price both high-to-low and low-to- high are two useful options. Sorting by ascending/descending alphabetical order and category, brand names are other options those are included for convenience.

5.2 Server-side Validations

Proper server side validations are performed before fetching any data from the database or storing into it. The significance of server side validation is that even if someone manages to inject HTML codes through to the server, the server-side validation will check if the request is a legal/ non-violating one before the request actually reaches the database. This way any kind of harm to the database can be prevented effectively.

5.2.1 During Registration

- During the registration process users are prompted to use unique e-mail addresses. The process of checking the uniqueness of the user given e-mail is performed through server-side validation. Inside the file “RegisterController.php” - the controller that manages the user registration process, checks the validity of the uniqueness of the given e-mail. To make the request posting safe, a CSRF token is sent with the form request to the server.
- Password format and the correctness of the password is checked and confirmed in the controller.
- Registering users are instantly assigned the “Subscriber” role to their account.
- The confirmation code is generated by encrypting the current UNIX time and re-shuffling it before sending to the user’s e-mail address.

5.2.2 In Cart Functionality

Users can conveniently add products to the cart and deduct or remove them at will. This process of addition and subtraction may seem very simple but in reality, even a tiny little loophole can break the database during this process.

5.2.3 In Checkout

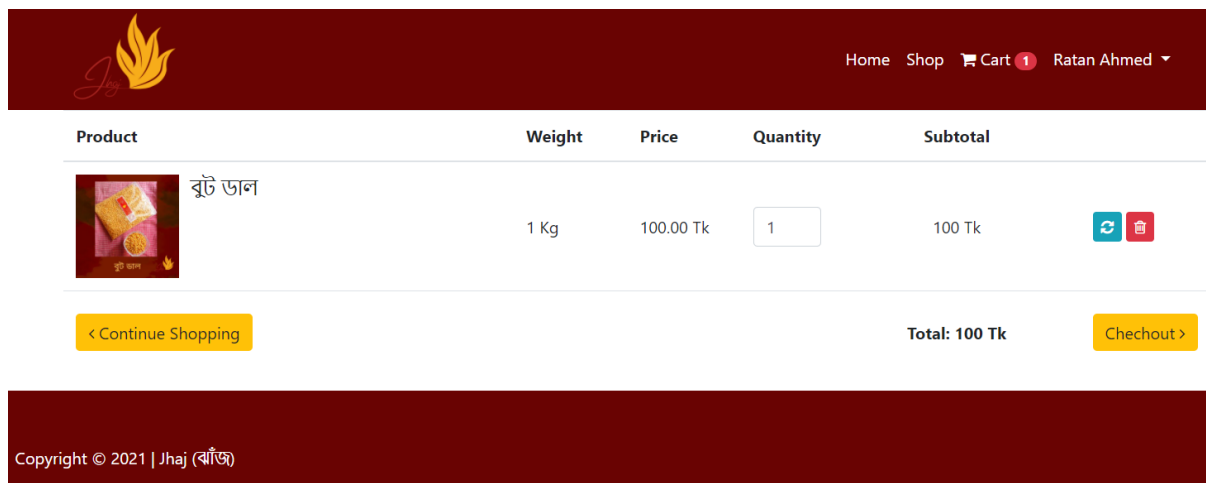


Fig. 5.2.3: In Checkout

As the user proceeds forward to checkout, the controllers will look for users advanced information. If the info are not available, the user cannot go forward, the user must fill out the information form to continue the purchase session. If advance info pre-existed, the user can choose to either proceed instantly or edit the appropriate info to continue. This process is also validated server-side. After the order is placed, the current stock is checked accessing the database before forwarding, if the product has been reduced meanwhile, the user is redirected back or else the checkout process forwards and updates the database accordingly.

Your Order					
Product	Weight	Price	Quantity	Subtotal	Status
 বুট ডাল	1 Kg	100.00 Tk	3	300 Tk	Received
 মুগ ডাল	1 Kg	150.00 Tk	1	150 Tk	Received

Fig. 5.2.4: Order

There are quiet a few more places here and there inside the controllers to ensure the data validation, such as the 'ProductsController.php' checks if the feedbacks shown on a product are of the current user, if they are not, they cannot be deleted by the current user. Another mentionable one is inside the 'ProductsController.php', where the staff can choose to add a new brand for the product on the go and it checks server-side if the requested brand exists or not (discussed later in 'Redundancy Free Database'). Concisely, I considered any visible/potential loopholes as threats to the system and put efforts within our extent to remove them.

5.3 Redundancy Free Database

Throughout the whole project I carefully tried to maintain the non-redundant approach for our database system. From storing users' advanced info to not updating the product stock until the checkout is completed, less-to-no redundancy was one of our top priorities.

5.3.1 In Product Management

During addition of new products to the database (by office staff) there might come a scenario where a product of the same name and same brand exists but the staff failed to notice that beforehand. In such case, adding a product having same name results in data redundancy.

To prevent this kind of potential record duplication, the application checks if a product having the same name and the same brand pre-exists in the database. If a matching product is pre-existent, the stock number for the product is increased adding new stock quantity to the previous one and price for the product keeps the lower amount as the updated price.

Another quite similar a scenario arises during addition of a new product, when a new brand name for the product has to be entered but the database does not contain the brand

name. In such case the staff can simply insert the name of the brand instead of an existing one. Now, the controller receives the request, checks for a match if the brand pre-exists. If it does, the is assigned to the product. If the brand is a new one, the controllers inserts it into the database giving it a new id and saves it after assigning it to the product.⁽⁷⁾ This way any potential redundancy arising from duplicate data existence is avoided.

Note that, product's 'Categories' and 'Brands' are kept in different tables for the convenience of managing them in advanced fashion (e.g., use in navigation bars, sub- categorize), or manipulating later in time by the staff. This keeps the option of adding extra attributes to categories/brands open for future use.

5.3.2 In Advanced Info for Users

For the advanced user info, I reduced redundancy by keeping the table corresponding to advanced user info separated from the main users table. This way the users do not necessarily have to insert their advanced info until they try to purchase something; so the database and the table is more likely kept relatively smaller. This approach also helps to minimize the database access cost when a user performs regular tasks like adding feedbacks, contacting customer support, watching histories etc., where the only entity the user needs is the ID to get access to those task initially. Had the other data been in the same table, more cost would have caused due to fetching bigger data chunks. On deletion of any user the advanced info also gets deleted.

5.4 Summary

Key background features and their implementation procedures have been discussed in the chapter. Ease-of-use, proper validation and the process of non-redundancy go hand in hand in an interactive project. Implementing them and making sure they all work together is a delicate task.

Chapter 6

View Optimization

The UI (user interfaces), or ‘View’ as termed in MVC pattern is a significantly vital part of an application. That is because the users would likely have no interest in knowing what kind of features are used in the application, how fail-safe the application is or how less redundant the application’s database is. If the users do not like the view or if they feel uncomfortable using the app they would most likely lose their interest on the application. In this section I briefly mention a few of the procedures I undertook to optimize the application’s views.

6.1 Using Service Providers

Service providers helps load variables, routes, event listeners and such entities during the start-up, before any views or controllers are loaded. We passed our frequently used models such as categories, brands to our main layout view using the ‘AppServiceProvide.php’ at the application’s booting time. The categories and brands are fetched from the database and sent to the main layout (app.blade.php) page during program initiation. This way, I maintain non-redundancy in our views too.

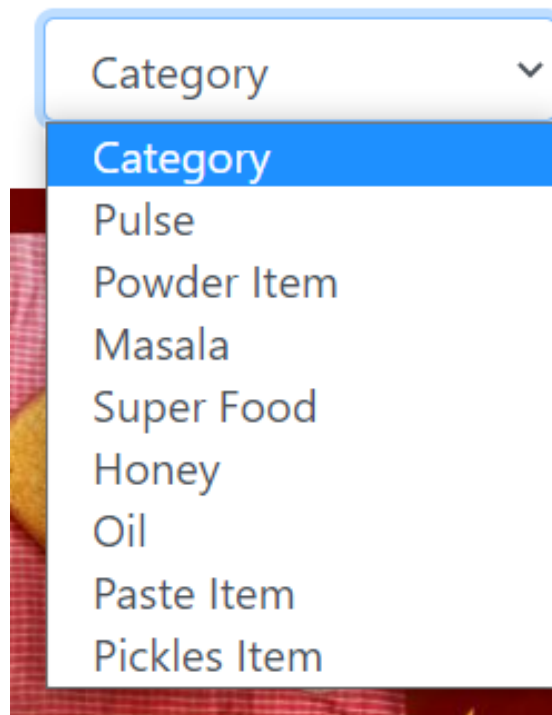


Fig. 6.1: Categories

Now, if a user starts adding the product to the cart, the product's available quantity keeps reducing and shown accordingly to the user, and as long as the available quantity does not reach 0 the 'In Stock' text is kept shown. If the user adds all the available stock to the cart the text then shows 'Limit Reached' in red colored text. All of these mentioned functionalities are carried out without affecting the database right away- as I discussed in the earlier section, which adds a sense of vitality to the view manipulation.

6.2 Summary

Laravel provides developers with a subtle view management system to avoid any unnecessary work procedure. Idea of extending a main layout, and/or partial layouts in views is an effective feature for reducing overhead; hence increasing the throughput quality.

Chapter 7

Conclusion

7.1 Learnings

There are a few learnings worth mentioning those I learned working on the project. These realizations would surely help if kept in mind for future projects.

- Need for new modules will always persist
- Building a web application is a continuous process
- Bug-free modules are products of countless re-testing
- Careful time-management is very crucial
- Regular team meeting during development is a must
- Maintaining good coding practices is necessary for bigger projects

7.2 Conclusion

In the project I explored and demonstrated a few advanced, user-friendly features to carry out the regular tasks. Working on this project, I came to a better understanding of how e-commerce and engineering techniques can work hand in hand to bring more advancement in the sector. We hope people interested in this sector would get a clear view on the workflow of these kind of businesses and take that extra step to explore the sector for the purpose of improvement.

7.3 Future Work

- Development of UI
- Product review and rating feature
- Sub-categorize products
- Home delivery & tracking system
- SMS verification system
- Search Bar with Artificial Intelligence
- Email Variation
- Digital Payment Gateway
- Chat Bot

Appendix

Code Snippets:

1.

```
public function index()
{
    // return Auth::user()->name;
    if(Auth::check() && Auth::user()->name == "Admin"){

        $data = [
            'users' => User::all(),
            'orders' => Order::all()
        ];
        // return $data;

        return view('admin')->with($data);
    }

    else
        return view('home');
}
```
2.

```
public function main()
{
    $product = [
        'products' => DB::table('products')->limit(8)->get(),
        'latest_products' => DB::table('products')->limit(4)->get()
    ];
    // return $products;
    return view('main')->with($product);
}
```
3.

```
public function index()
{
    $products = Product::all();
    return view('products', compact('products'));
}
```
4.

```
public function details($id)
{
    $details = Product::find($id);
```

```

$cat = $details->category;
// $related = Product::find
$data = [
    'details' => Product::find($id),
    'related' => Product::all()->where('category', $cat)
];

```

```

return view('details')->with($data);
}

```

```

5.      public function cart()
        {
            return view('cart');
        }

```

```

6.      public function addToCart($id)
        {
            $product = Product::find($id);

            if(!$product) {

                abort(404);

            }

```

```

$cart = session()->get('cart');

```

```

// if cart is empty then this the first product
if(!$cart) {

```

```

    $cart = [
        $id => [
            "id" => $product->id,
            "name" => $product->product_name,
            "quantity" => 1,
            "price" => $product->price,
            "product_photo" => $product->product_photo,
            "weight" => $product->weight
        ]
    ]

```

```

];
// return $cart;
session()->put('cart', $cart);

```

```

        return redirect()->back()->with('success', 'Product added to cart successfully!');
    }

    // if cart not empty then check if this product exist then increment quantity
    if(isset($cart[$id])) {

        $cart[$id]['quantity']++;

        session()->put('cart', $cart);

        return redirect()->back()->with('success', 'Product added to cart successfully!');

    }

    // if item not exist in cart then add to cart with quantity = 1
    $cart[$id] = [
        "name" => $product->product_name,
        "quantity" => 1,
        "price" => $product->price,
        "product_photo" => $product->product_photo,
        "weight" => $product->weight

    ];

    session()->put('cart', $cart);

    return redirect()->back()->with('success', 'Product added to cart successfully!');
}

```

```

7.      public function update(Request $request)
    {
        if($request->id and $request->quantity)
        {
            $cart = session()->get('cart');

            $cart[$request->id]["quantity"] = $request->quantity;

            session()->put('cart', $cart);

            session()->flash('success', 'Cart updated successfully');
        }
    }
}

```

8. public function remove(Request \$request)
 {
 if(\$request->id) {

 \$cart = session()->get('cart');

 if(isset(\$cart[\$request->id])) {

 unset(\$cart[\$request->id]);

 session()->put('cart', \$cart);
 }

 session()->flash('success', 'Product removed successfully');
 }
 }

9. public function checkout()
 {
 session()->flash('success', 'Ordered successfully!');
 return view('checkout');
 }

Short Forms& Acronyms

MVC	:	Model-View-Controller
ORM	:	Object Relational Mapping
API	:	Application Programming Interface
OOP	:	Object Oriented Programming
SRS	:	System Requirements Specification
UI	:	User Interface
DBMS	:	Database Management Systems
IDE	:	Integrated Development Environment
SHA1	:	Secure Hash Algorithm 1

References:

- [01] <https://www.w3schools.com>
- [02] <https://en.wikipedia.org/wiki/Laravel/>[03]<https://laracasts.com/>
- [04] <http://stackoverflow.com/>
- [05] <http://www.laravel.io/>
- [06] <http://www.tutorialspoint.com/laravel/>
- [07] <http://www.youtube.com/>
- [08] <http://bootsnipp.com/>
- [09] https://en.wikipedia.org/wiki/Design_specification/
- [10] <http://www.getbootstrap.com/>