

ASSIGNMENT

LEVEL 5

COMP50016-2: Server-Side Programming

IF2321COM

Individual Assignment

**THENURI SANDARA HETTIARACHCHI
CB010529**

Hand Out Date: 1th October 2023

Hand in Date: 17th July 2023

Table of Contents

Acknowledgement	4
Introduction.....	4
Link to Mind Map.....	4
GitHub Link	4
System Objectives.....	5
The High-level goal of this system	5
Target Audience.....	5
System Architecture	6
Backend Architecture	6
Frontend Architecture.....	6
Architecture for the mobile	7
ER Diagram.....	7
Mind Map.....	7
System Features (Screen shots of the system).....	15
Features	15
Screen shots of the mobile app.....	16
Screen shots of the CRM system	19
Screenshots of the code & postman requests	30
Assumptions.....	45
Future Upgrade plane.....	45
Expanding as a SaaS solution	46
Test Cases.....	47
Performance Testing.....	50
LightHouse Testing	50
Conclusion	51

Figure 1-Front end Architecture-use of alpine JS	6
Figure 2 – ER Diagram	7
Figure 3 - Mind Map.....	14
Figure 4 - Mobile App Design	18
Figure 5-CRM Login page.....	19
Figure 6-CRM Registration Page.....	20
Figure 7 - Admin Employee Dashboard part1	21
Figure 8 - Admin & Employee Dashboard Part 2.....	21
Figure 9 - Admin & Employee part 3	22
Figure 10 - Customer Tab	22
Figure 11 - Product Page.....	23
Figure 12 - Promotions Page.....	23
Figure 13 - Employee Display	24
Figure 14 - Supplier Page	24
Figure 15 - Order Page.....	25
Figure 16 - Delivered Orders	26
Figure 17 - Employee display for employees	26
Figure 18 - Driver Dashboard	27
Figure 19 – mailtrap.....	28
Figure 20 - Mail that the customer can see	28
Figure 21 - pusher	29
Figure 22 - Migration Files	30
Figure 23 - Blade files.....	30
Figure 24 - Livewire files	31
Figure 25 - Blade Files.....	31
Figure 26 – Controllers	31
Figure 27 - Livewire Controllers	32
Figure 28 – Models	32
Figure 29 - Cart Controller	33
Figure 30 - API Add to cart.....	34
Figure 31 - API API Controller.....	35
Figure 32 - API Controller	36
Figure 33 - API Register	37
Figure 34 - API Login	38
Figure 35 -API Getting Customer Details	38
Figure 36 - API Loyalty Points	39
Figure 37 - API Promotions	39
Figure 38 - Order Controller	40
Figure 39 – Order.....	41
Figure 40 - Getting Product categories.	41
Figure 41 - Getting Products by category name.	42
Figure 42 – WEB.php	43
Figure 43 - API.php.....	44

Acknowledgement

I would like to make this an opportunity to convey my gratitude to our lecturer Mr.Vipula Anandapiya for guiding us to do our course work as well as the encouragement to be novel and creative in our course work.

Introduction

This documentation provides a comprehensive overview of the system I have developed. I have developed a CRM system for a grocery store (FRESH EXPRESS) that provides fresh items such as Meat, Vegetable, Fruits, and seafood. The CRM system manages inventory, measure sales, keep stack of the client information, order management, supplier management, send promotion details to customer. On completion of dispatch of each order system will send an email to the customer saying the order has been dispatched with the order information this is been done by mailtrap. This document covers the system objectives, features, architecture of the system and finally system presentation with the necessary screenshots.

Link to Mind Map

<https://gitmind.com/app/docs/mzw7yn5i>

GitHub Link

<https://github.com/Thenuri/fresh-express.git>

System Objectives

The High-level goal of this system

optimize inventory management: Implement efficient inventory management and tracking all order details are matched with the dispatches ensuring highest customer satisfaction. Also, automatic update of stock levels enables purchasing to handle reorder levels efficiently. This process ensures every item that is delivered to the customer is fresh and this helps minimize the stockout.

Streamline delivery: Delivering the customer orders to their required location.

Enhance customer Engagement: Making the customer's shopping experience as convenient as possible with customizable shopping list and offering loyalty points to each order that customer makes.

Track Sales: For each week the system tracks the total revenue, and it finds out the most moving items categories as well as the sales of each product.

Gather customer purchases: The administration can find out each customer's purchases and customers' address book as well.

Promotion Alerts: The system facilitates including customer promotions and these could be sent to each customer. With this facility customer can choose to do online transactions using the necessary card. These promotions will get displayed in the promotions tab in the mobile app.

Target Audience

The target audience for this system is the administrator who is responsible for tracking sales and as well as the employees who are responsible for dispatching customer orders. And customers who want to make their life easy and prefers delivering their order to the doorstep.

System Architecture

Backend Architecture

Laravel Jetstream: The backend of the CRM is developed using Laravel Jetstream, it's a robust framework for building web applications. This follows the (MVC) architecture pattern and provides essential features for user authentication and API support. The Laravel Jetstream handles various backend features in our CRM system such as:

User Authentication: this provides secure authentication to the system, and it gives necessary permission to the administrators and the employees.

API Integration: In the CRM I have used Laravel sanctum since I'm using a mobile app to get customer interaction this is easy to interact with the backend system.

Data Handling: This enables data retrieval and store the data in a structure's manner, and it manages the user profiles effectively.

Livewire: Since I am using Laravel Jetstream it is easy to use livewire. This is a great stack to make the application dynamic and reactive. In the CRM system I have made all the crud functionalities using livewire.

Frontend Architecture

Tailwind CSS: I used this to streamline the styling process and make the interface look more look alike. So, this gives the user a pleasant look at the system.

Alpine JS: This is used to enhance the interaction between the user and the system. I have used alpine JS to display the number of orders in the sidebar and to show the flash messages after an interaction is done.

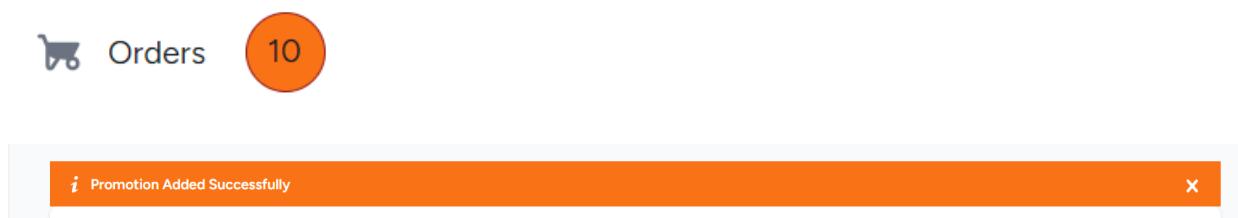


Figure 1-Front end Architecture-use of alpine JS

Architecture for the mobile

For the mobile application I have used flutter framework for the frontend and dart for the backend.

ER Diagram

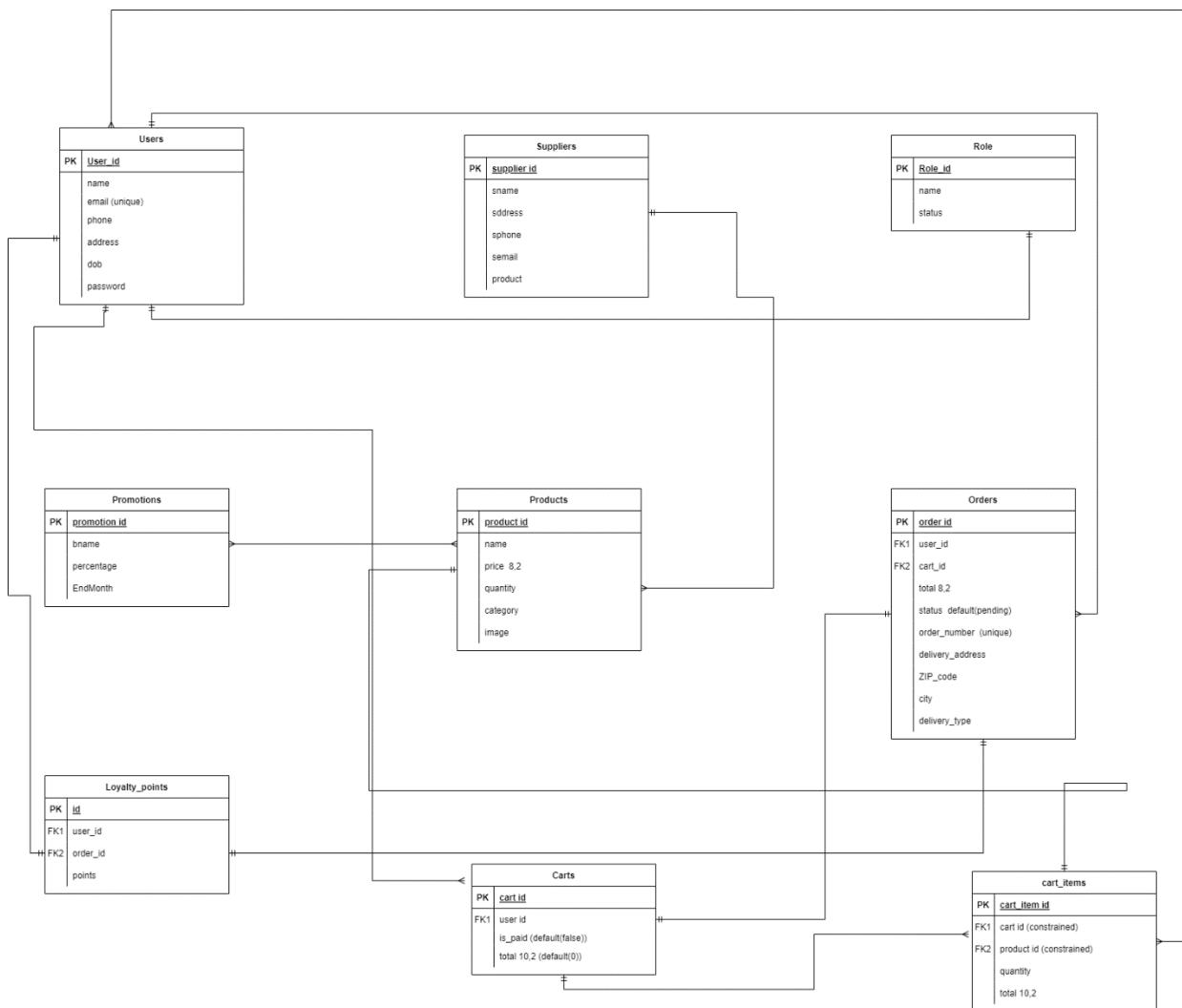
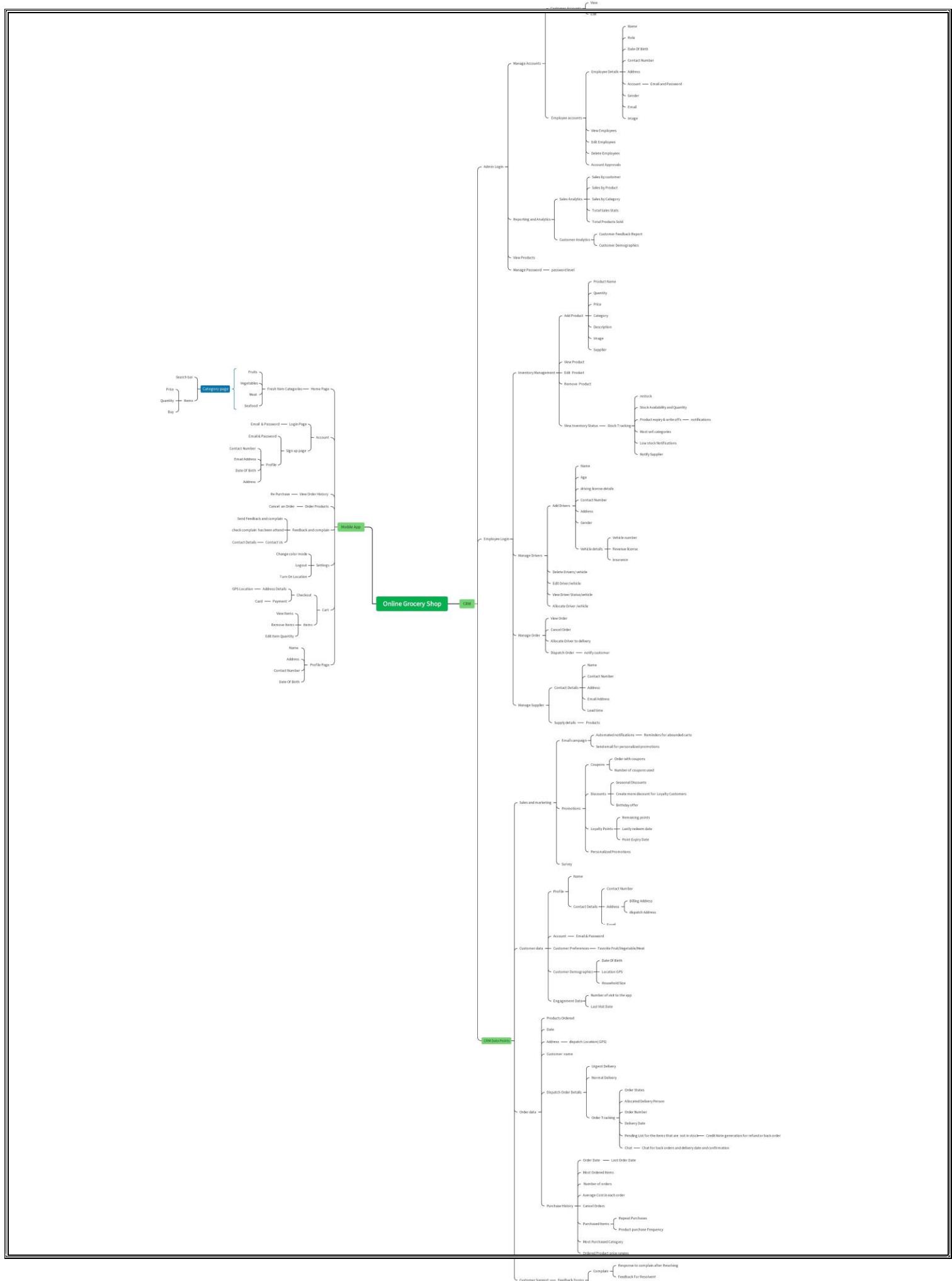
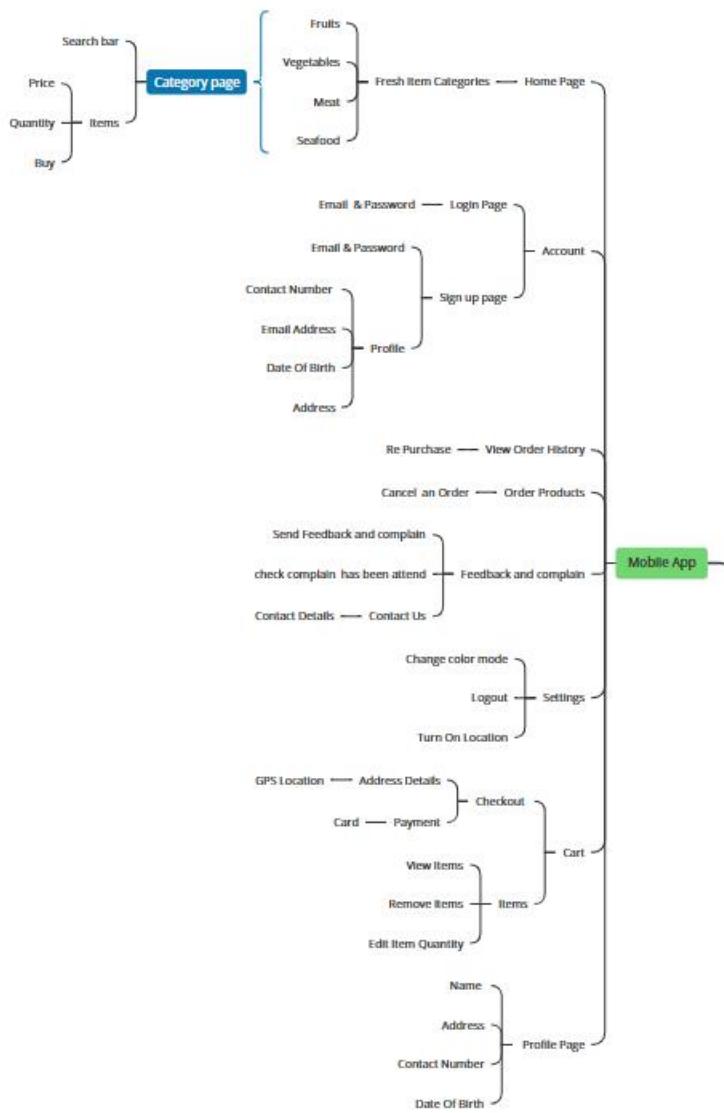


Figure 2 – ER Diagram

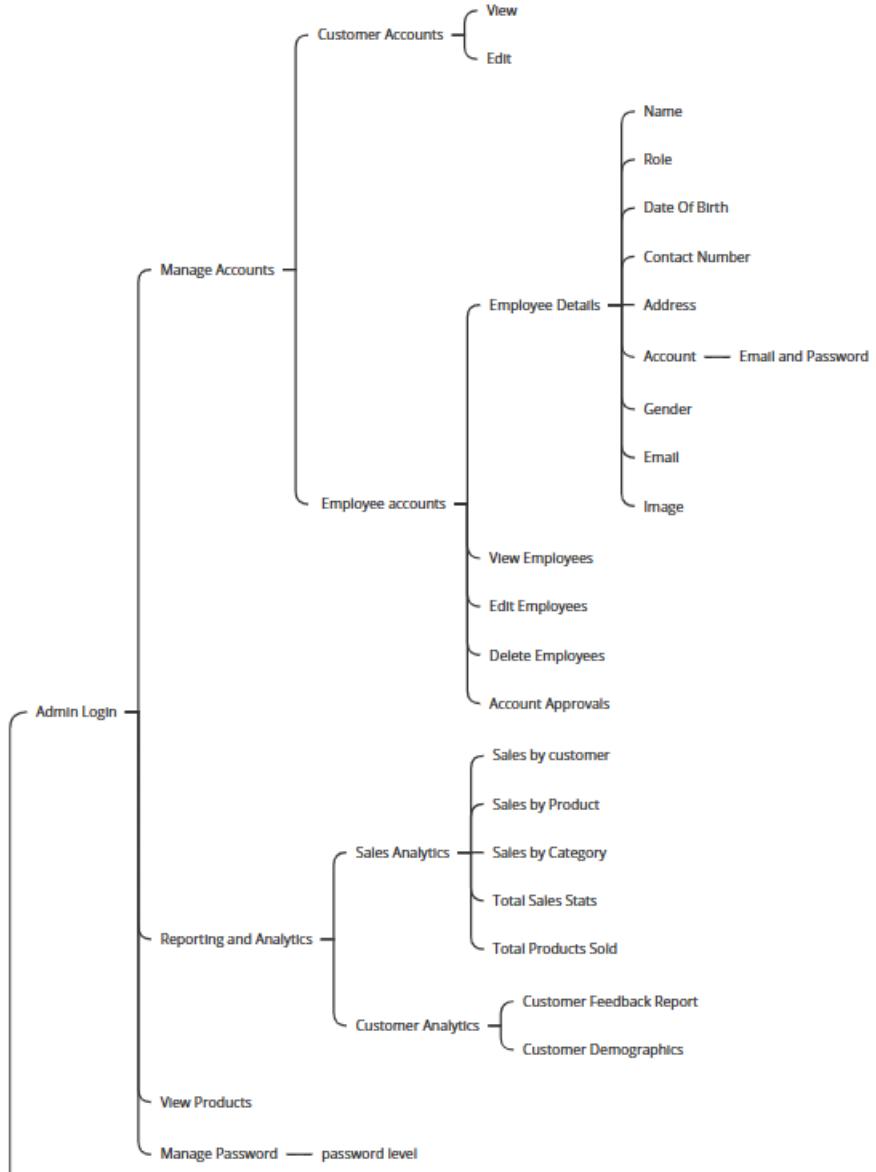
Mind Map



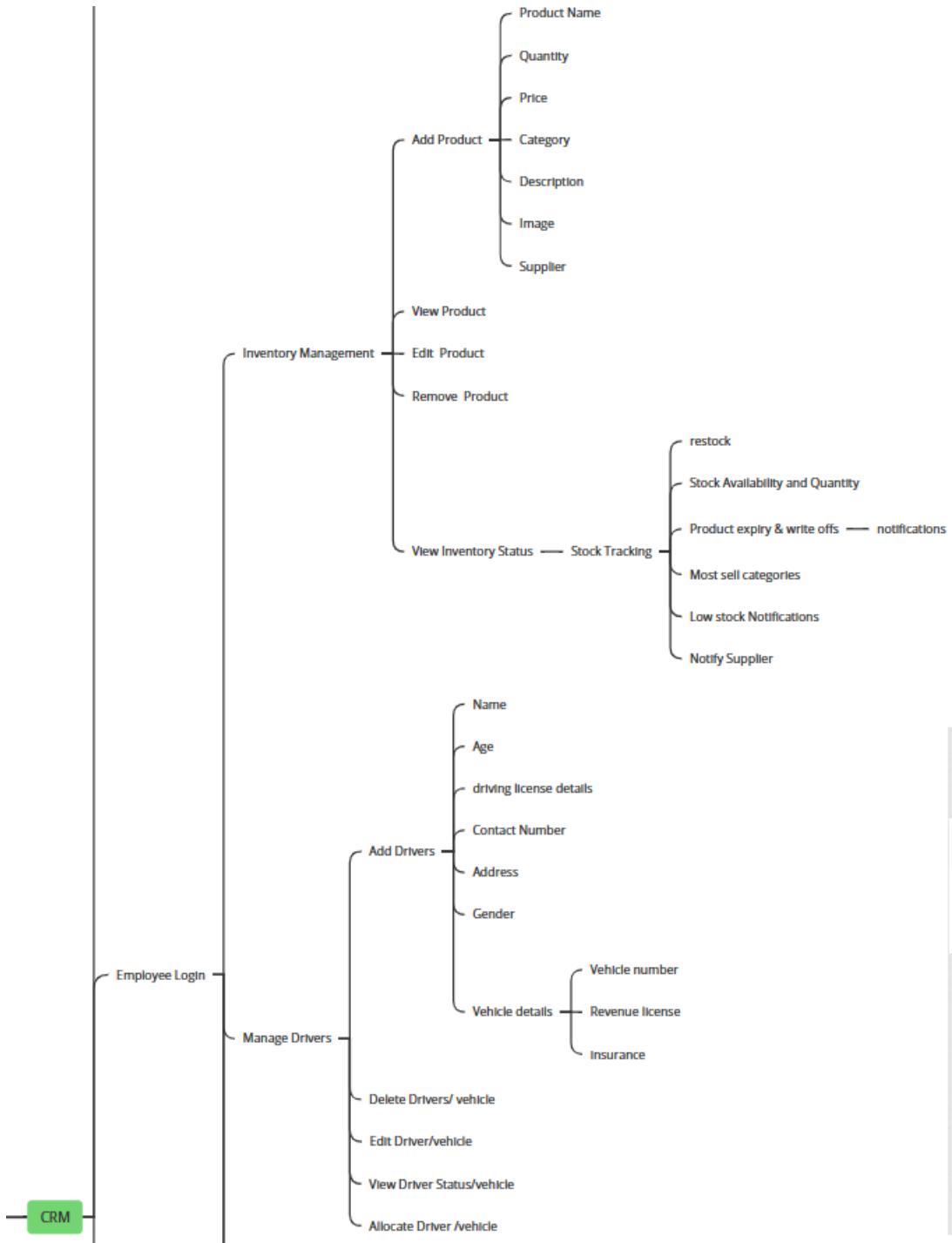
Mobile App

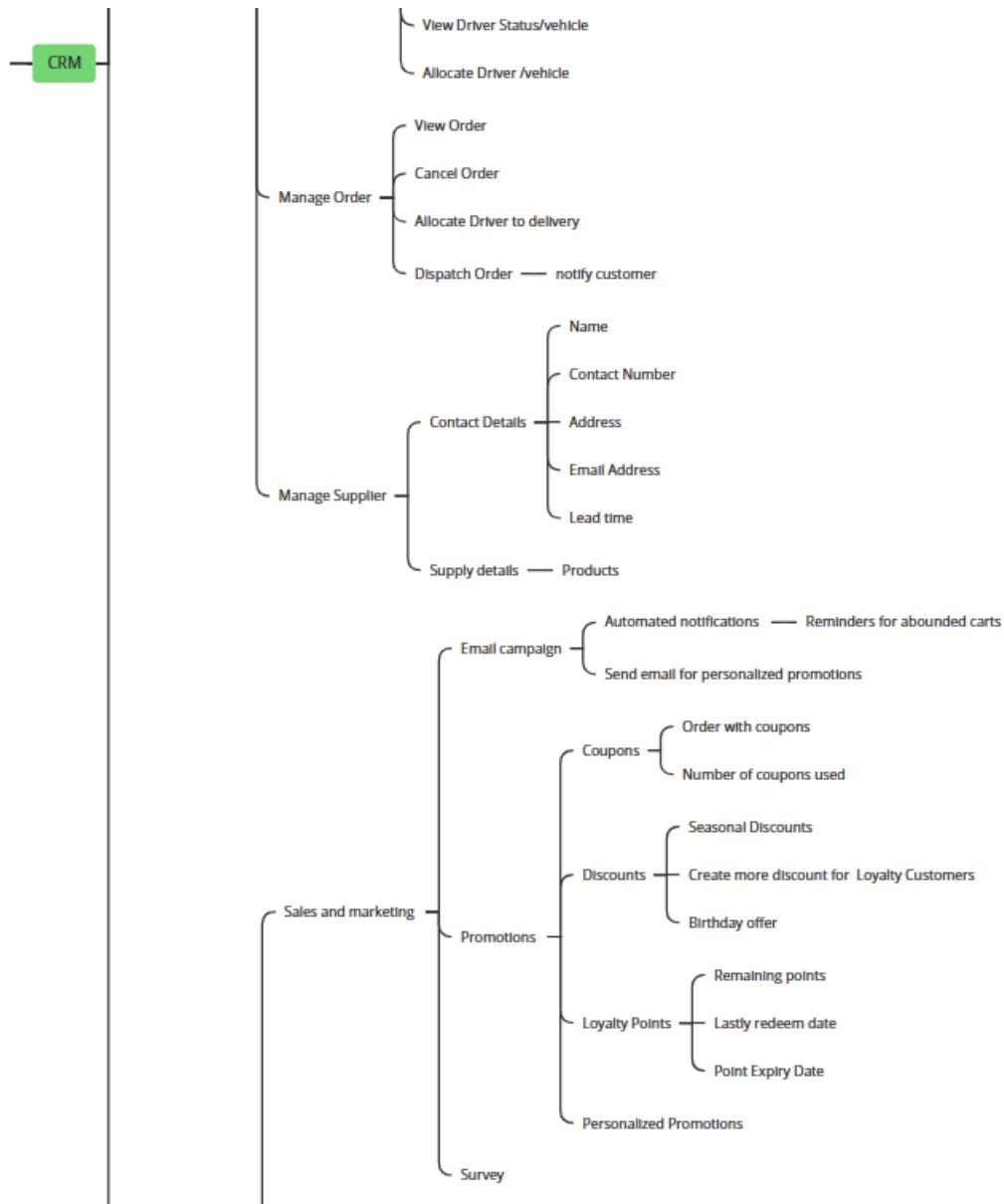


CRM



CRM





CRM Data Points



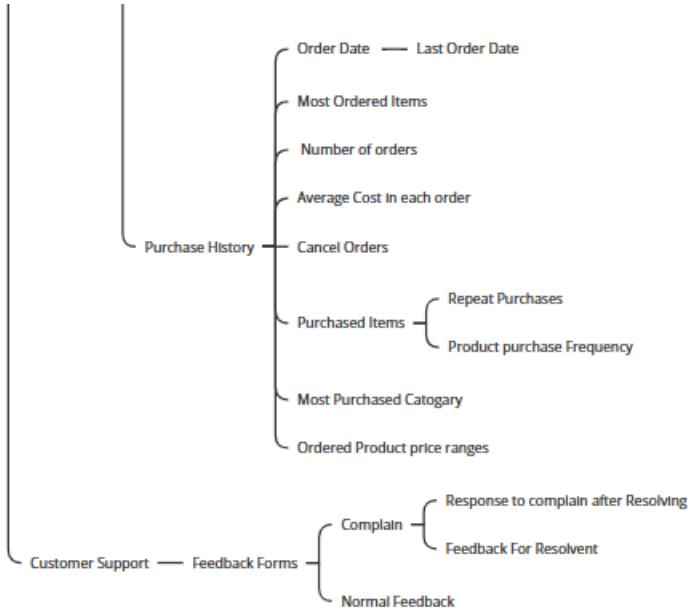


Figure 3 - Mind Map

System Features (Screen shots of the system)

Features

Customer:

- Add item to cart
- Redeem loyalty points
- Choose favorite item
- Customer can see promotions
- Login
- Register
- Update Profile

Employee

- Login
- Register
- Can see customer details
- Can see the drivers in the system
- Add Promotions
- Add Products
- Dispatch an order

Admin

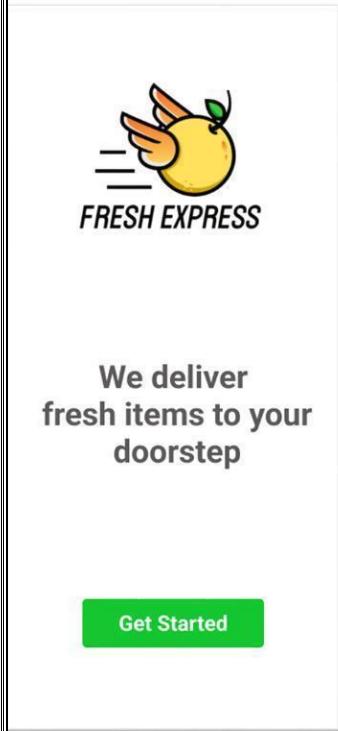
- Login
- Register
- Can see customer details
- Can see the drivers and other employees in the system
- Add Promotions
- Add Products
- Dispatch an order

Driver

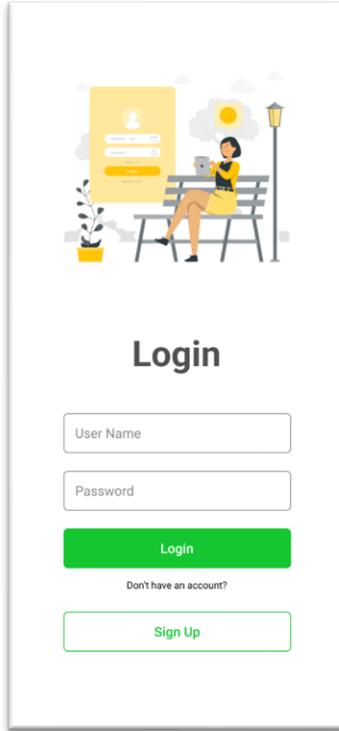
- Can see the customer orders that should be delivered

Screen shots of the mobile app

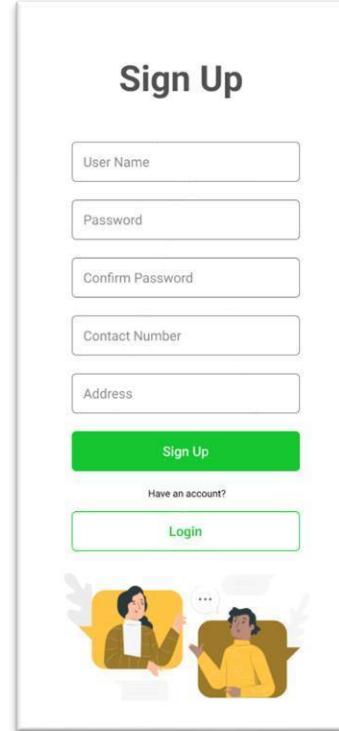
Main Page



Login Page



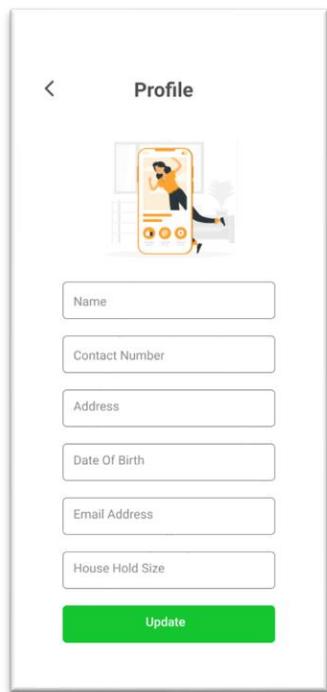
Sign Up Page



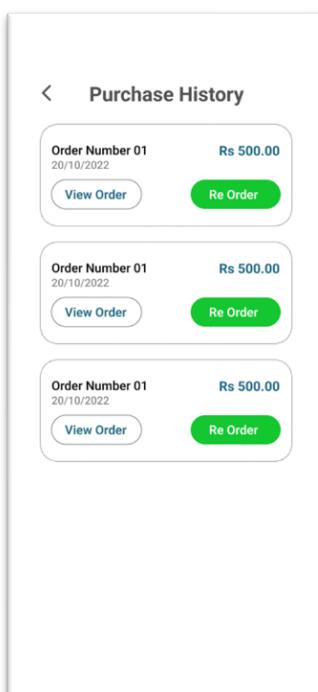
Home Page



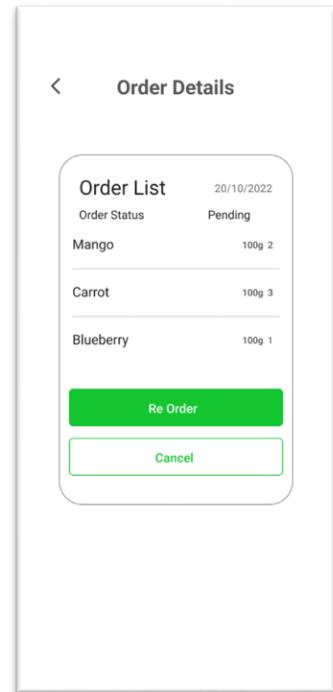
Profile



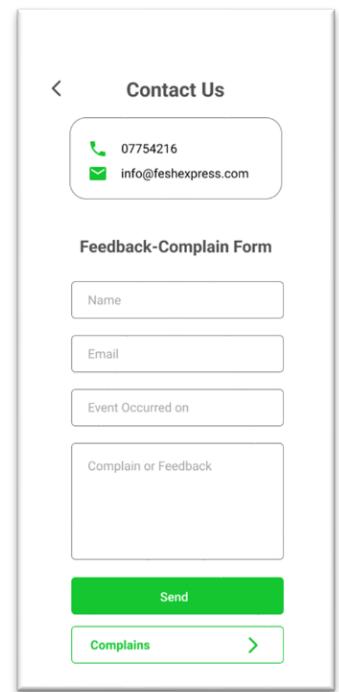
Order History



Order Details



Contact Us



Complains

< Complain Information

Complain

Event Occurred on 20/10/2022

Description Some items are rotten. Please make sure to send good items

Complain Status Resolved

Complain

Event Occurred on 20/10/2022

Description Some items are rotten. Please make sure to send good items

Complain Status Resolved

Settings

< Settings

Dark Mode

Log Out

Enable Location

Promotion

< Your Loyalty Points

1000

Point expiry date: 2023/05/10

Promotions & Discounts

For HNB card **20% Off**
Till End of May

For Seylan card **30% Off**
Till End of May

Favorites

My Favorites

Blueberry - 100g
Rs 250.00 per 100g
Rs 500.00

Blueberry - 100g
Rs 250.00 per 100g
Rs 500.00

Blueberry - 100g
Rs 250.00 per 100g
Rs 500.00

Blueberry - 100g
Rs 250.00 per 100g
Rs 500.00

Favorites

Item Pages

search

Carrot 100g-Rs.250

Pumpkin 100g-Rs.250

Pumpkin 100g-Rs.250

Tomato 100g-Rs.250

Garlic 100g-Rs.250

Garlic 100g-Rs.250

search

Strawberry 100g-Rs.250

Blueberry 100g-Rs.250

Blueberry 100g-Rs.250

Mango 100g-Rs.250

Grapes 100g-Rs.250

Grapes 100g-Rs.250

search

Beef 100g-Rs.250

Chicken 100g-Rs.250

search

Prawns 100g-Rs.250

Oysters 100g-Rs.250

Fish 100g-Rs.250

crabs 100g-Rs.250

The mobile application interface consists of the following screens:

- Cart**: Displays the user's shopping cart with three items: Blueberry - 100g (Rs 500.00), Mango - 100g (Rs 500.00), and another Blueberry - 100g (Rs 500.00). Subtotal: Rs 1000.00, Delivery Fee: Rs 50.00, Total: Rs 1050.00.
- Address Details**: A form for entering delivery address details: Address, City, Zip Code, or Pick Location. Delivery Type options: Urgent Delivery or Normal Delivery.
- Preview**: Shows the order summary with Available Points (1000) and a button to Redeem them. It also includes a section for Coupons.
- Payment**: Accepts various payment methods including VISA, Mastercard, and American Express. Fields for Name On Card, Card Number, Expiry, and CVV are provided.
- Map**: A map showing the delivery location with a red pin. The address listed is 1268, Suncity Township-I, Rohtak, Haryana 124001, India.
- Thank you Page**: Confirms the order placement with a message: "Your order has been placed!" and an illustration of two people on a balcony.
- Chat**: A placeholder screen for grocery chat, featuring a "Back to home" button.

Figure 4 - Mobile App Design

Screen shots of the CRM system

Login Page

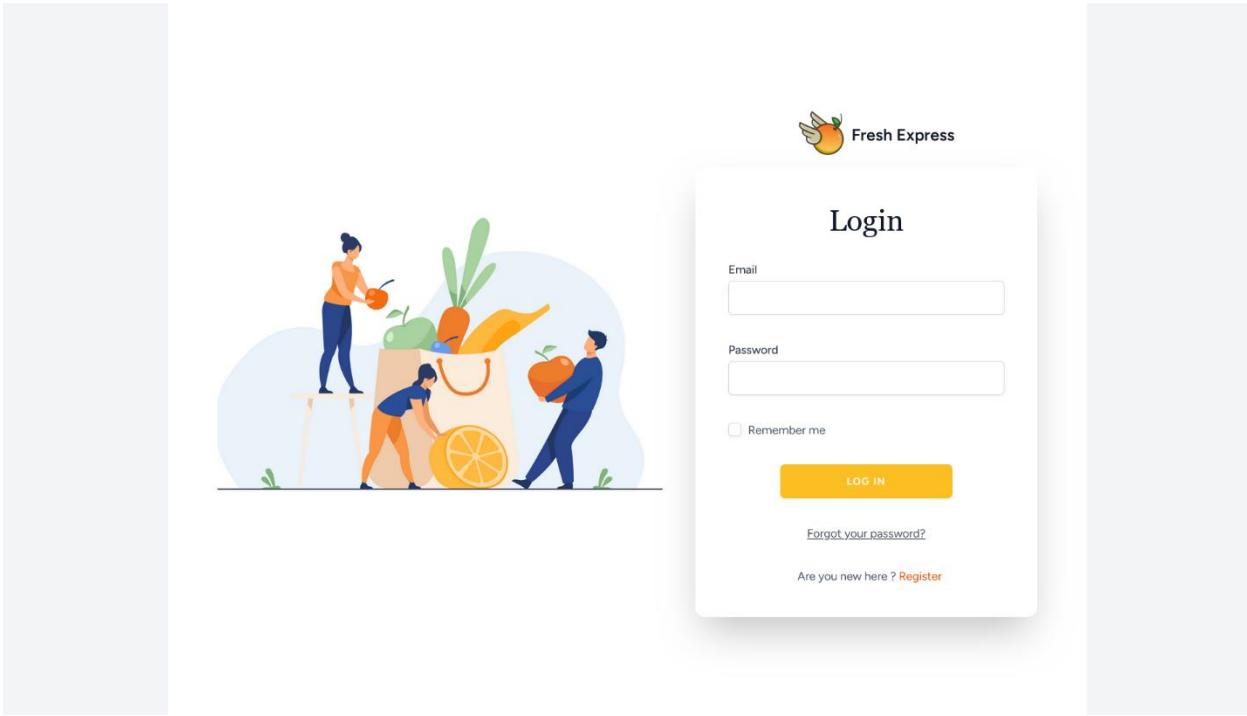


Figure 5-CRM Login page

Registration Page



Register

Name

Register As

Email

Phone

Address

Date of Birth

Password

Confirm Password

I agree to the [Terms of Service](#) and [Privacy Policy](#).

Already have an account? [Login](#)

REGISTER



Figure 6-CRM Registration Page

Dashboard (Admin & Employee)

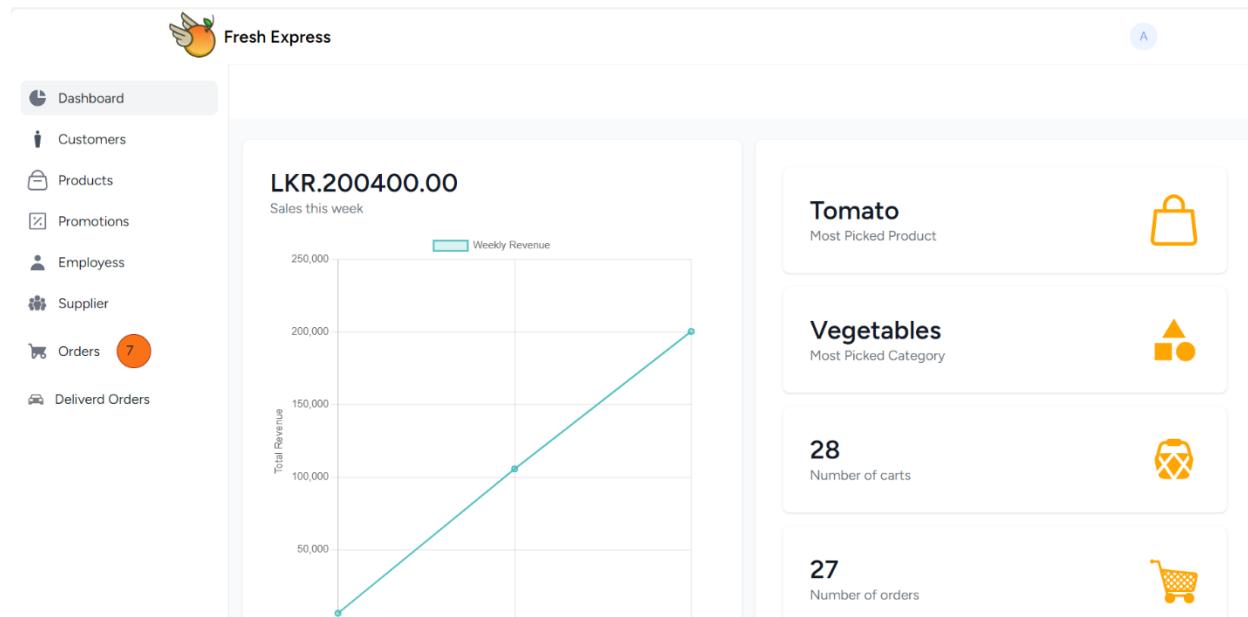


Figure 7 - Admin Employee Dashboard part1

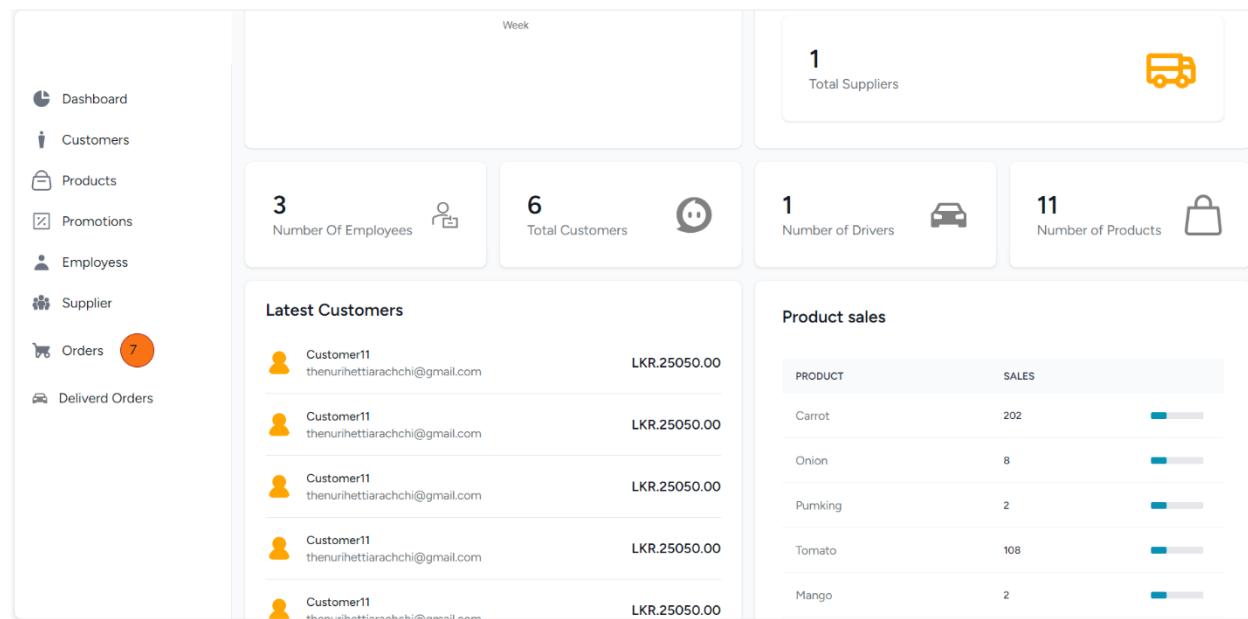


Figure 8 - Admin & Employee Dashboard Part 2

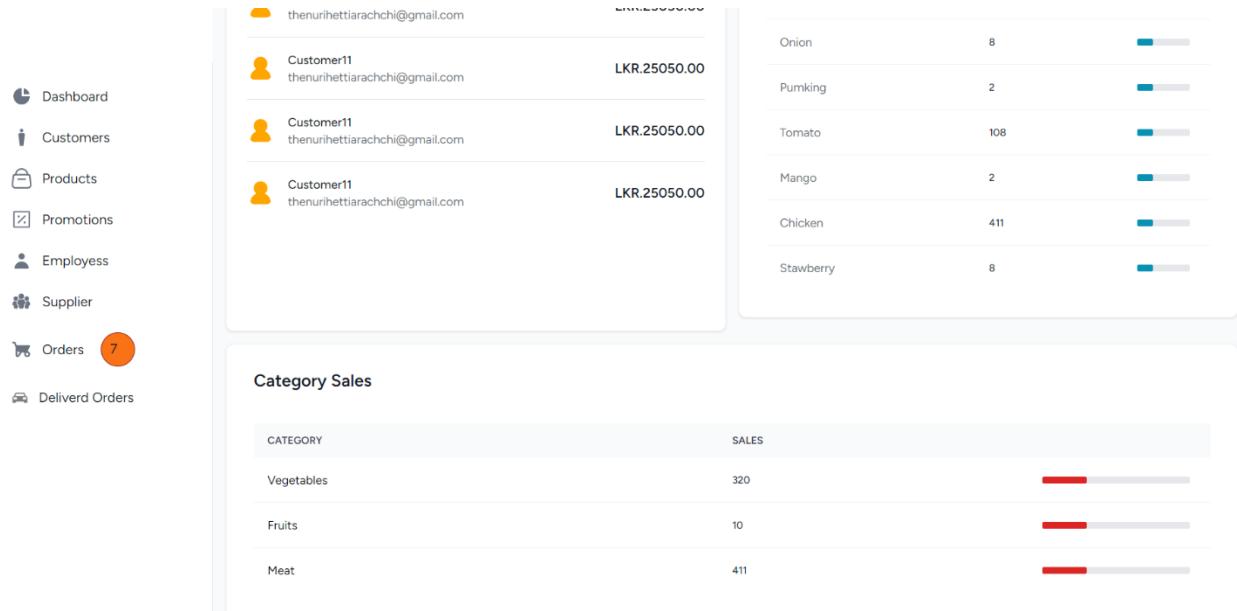


Figure 9 - Admin & Employee part 3

Customer Tab

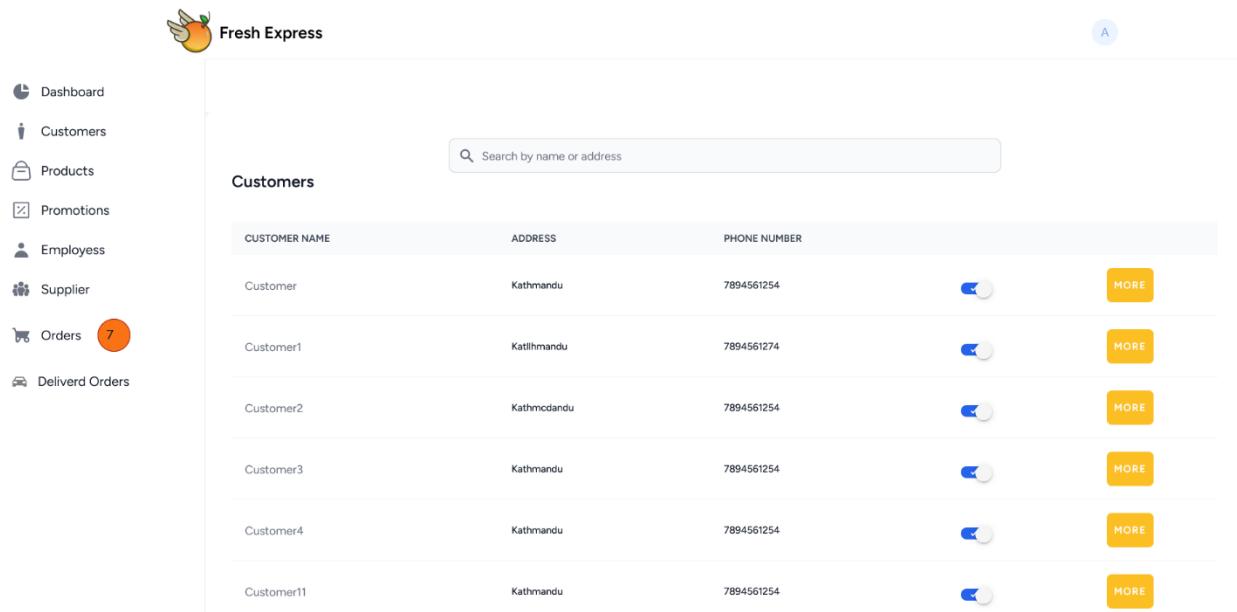


Figure 10 - Customer Tab

Product Page

Fresh Express

Products

PRODUCT ID	PRODUCT NAME	QUANTITY	PRICE	CATEGORY	IMAGE
1	Carrot	9800g	LKR250.00	Vegetables	
2	Onion	9990g	LKR250.00	Vegetables	
3	Pumking	9996g	LKR250.00	Vegetables	
4	Tomato	890g	LKR250.00	Vegetables	
5	Mango	98g	LKR250.00	Fruits	

Showing 1 to 5 of 11 results

Figure 11 - Product Page

Promotions Page

Fresh Express

Promotions

PROMOTION ID	BANK NAME	PERCENTAGE	END MOTH
1	PRO1	50	June
2	sampath	40	may

Figure 12 - Promotions Page

Employees Display page.

The screenshot shows the 'Employees Display' page of the Fresh Express application. The left sidebar contains navigation links: Dashboard, Customers, Products, Promotions, Employees, Supplier, Orders (with 7 notifications), and Delivered Orders. The main content area has two sections: 'Employees(in shop)' and 'Deivers'. The 'Employees(in shop)' section displays three rows of data:

CUSTOMER NAME	ADDRESS	PHONE NUMBER	Switch
Employee	Kathmandu	7894561254	On
Thenuri Hettiarachchi	ffvvfv	0774781949	On
Thenuri Hettiarachchi	reerere	0741741741	On

The 'Deivers' section displays one row of data:

CUSTOMER NAME	ADDRESS	PHONE NUMBER	Switch
sachith	ddccccd	0774147874	On

Figure 13 - Employee Display

Supplier Page

The screenshot shows the 'Supplier' page of the Fresh Express application. The left sidebar contains the same navigation links as Figure 13. The main content area features a search bar with a magnifying glass icon and a yellow 'Search' button. Below the search bar is a table titled 'Supplier' with an 'ADD SUPPLIER' button. The table has columns: SUPPLIER ID, SUPPLIER NAME, ADDRESS, PHONE NUMBER, EMAIL, and PRODUCT. One row of data is listed:

SUPPLIER ID	SUPPLIER NAME	ADDRESS	PHONE NUMBER	EMAIL	PRODUCT
1	sachith	280/900cdrr	0778459874	shacith@fd.com	Mango

At the bottom right of the table are 'DELETE' and 'UPDATE' buttons.

Figure 14 - Supplier Page

Orders Page

The screenshot shows the 'Orders Page' for 'Fresh Express'. The left sidebar includes links for Dashboard, Customers, Products, Promotions, Employees, Supplier, Orders (highlighted with a red circle containing '7'), and Delivered Orders. The main content area is divided into three sections: 'New Orders', 'Previous Orders', and 'Canceled Orders', each with a table and search/filtering options.

New Orders

ORDER ID	ORDER NUMBER	CUSTOMER NAME	CITY	DELIVERY ADDRESS	ZIP CODE	DELIVERY TYPE	SELECT DRIVER	ORDERED PRODU
23	order23	Customer11	Example City	123 Main St	12345	Express	Select ▾	Tomato
24	order24	Customer11	Example City	123 Main St	12345	Express	Select ▾	Tomato
25	order25	Customer11	Example City	123 Main St	12345	Express	Select ▾	Tomato
26	order26	Customer11	Example City	123 Main St	12345	Express	Select ▾	Tomato
27	order27	Customer11	Example City	123 Main St	12345	Express	Select ▾	Tomato

Previous Orders

ORDER ID	ORDER NUMBER	CUSTOMER NAME	CITY	DELIVERY ADDRESS	ZIP CODE	DELIVERY TYPE	ORDERED PRODU
1	ORDER-731e1aa5-fcdd-42fd-9b96-953c7eb89934	Admin	Example City	123 Main St	12345	Express	Carrot
2	ORDER-f637b58b-3810-4c13-999e-39d9be325a08	Admin	Example City	123 Main St	12345	Express	Carrot
4	order2	Admin	Example City	123 Main St	12345	Express	Onion Pumki Tomat
5	order3	Admin	Example City	123 Main St	12345	Express	Tomat
6	order4	Customer11	Example City	123 Main St	12345	Express	Tomat

Showing 1 to 5 of 20 results

Canceled Orders

ORDER ID	ORDER NUMBER	CUSTOMER NAME	CITY	DELIVERY ADDRESS	ZIP CODE	DELIVERY TYPE	ORDERED PRODUCTS	QUANTITY
3	order1	Admin	Example City	123 Main St	12345	Express	Oyster	2
8	order6	Customer11	Example City	123 Main St	12345	Express	Carrot	2

Figure 15 - Order Page

Delivered Orders

The screenshot shows the 'Delivered Orders' section of the Fresh Express application. The left sidebar includes links for Dashboard, Customers, Products, Promotions, Employees, Supplier, Orders (with 7 notifications), and Delivered Orders. The main content area is titled 'Delivered Orders' and displays a table with columns: ORDER ID, CUSTOMER NAME, ADDRESS, ZIP CODE, and PHONE NUMBER. The data in the table is as follows:

ORDER ID	CUSTOMER NAME	ADDRESS	ZIP CODE	PHONE NUMBER
order11	Customer11	123 Main St	12345	7894561254
order15	Customer11	123 Main St	12345	7894561254
order16	Customer11	123 Main St	12345	7894561254
order17	Customer11	123 Main St	12345	7894561254
order19	Customer11	123 Main St	12345	7894561254
order22	Customer11	123 Main St	12345	7894561254

Figure 16 - Delivered Orders

Employee display for the employees

The screenshot shows the 'Employee display for employees' section of the Fresh Express application. The left sidebar includes links for Dashboard, Customers, Products, Promotions, Employees, Supplier, Orders (with 7 notifications), and Delivered Orders. The main content area is titled 'Deivers' and displays a table with columns: CUSTOMER NAME, ADDRESS, and PHONE NUMBER. The data in the table is as follows:

CUSTOMER NAME	ADDRESS	PHONE NUMBER
sachith	ddcccd	0774147874

Figure 17 - Employee display for employees

Driver Dashboard



Fresh Express

S

New Orders

ORDER NUMBER	CUSTOMER NAME	ADDRESS	ZIP CODE	PHONE NUMBER	DELIVERD
order11	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>
order15	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>
order16	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>
order17	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>
order19	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>
order22	Customer11	123 Main St	12345	7894561254	<button>DELIVERD</button>

Previous Orders

ORDER ID	CUSTOMER NAME	ADDRESS	ZIP CODE	PHONE NUMBER
order11	Customer11	123 Main St	12345	7894561254
order15	Customer11	123 Main St	12345	7894561254
order16	Customer11	123 Main St	12345	7894561254
order17	Customer11	123 Main St	12345	7894561254
order19	Customer11	123 Main St	12345	7894561254
order22	Customer11	123 Main St	12345	7894561254

Figure 18 - Driver Dashboard

I have used mailtrap to send email to the customer when the order has been dispatched.

The screenshot shows the Mailtrap web interface. On the left, there's a sidebar with navigation links: Home, Email Testing, Inboxes (selected), Email Sending, Email Marketing (with a 'soon' badge), Billing, Settings, and Help. The main area displays an email message titled "Dispatched Order". The message details are: From: Example <hello@example.com>, To: <thenurihettiarachchi@gmail.com>. The email was sent a few seconds ago. The message content is as follows:

Dispatched Order

From: Example <hello@example.com>
To: <thenurihettiarachchi@gmail.com>

Show Headers

HTML HTML Source Text Raw Spam Analysis HTML Check (red) Tech Info

Product Name | Quantity
Tomato | Quantity: 100

View Order

Thank you for using our application!
Regards,
FreshExpress

If you're having trouble clicking the "View Order" button, copy and paste the URL below into your web browser: <http://127.0.0.1:8000>

Figure 19 – mailtrap

The screenshot shows a customer-facing email from FreshExpress. The subject is "Hello!". The email body contains the following information:

Your order has been dispatched. Here are the details:

Order Number: order29
Total Price: LKR25050
Delivery Fee: LKR50
Purchased Products:

Product Name | Quantity
Tomato | Quantity: 100

View Order

Thank you for using our application!
Regards,
FreshExpress

If you're having trouble clicking the "View Order" button, copy and paste the URL below into your web browser: <http://127.0.0.1:8000>

Figure 20 - Mail that the customer can see

I used pusher for real time notification to notify the users of the CRM when a new order has arrived.

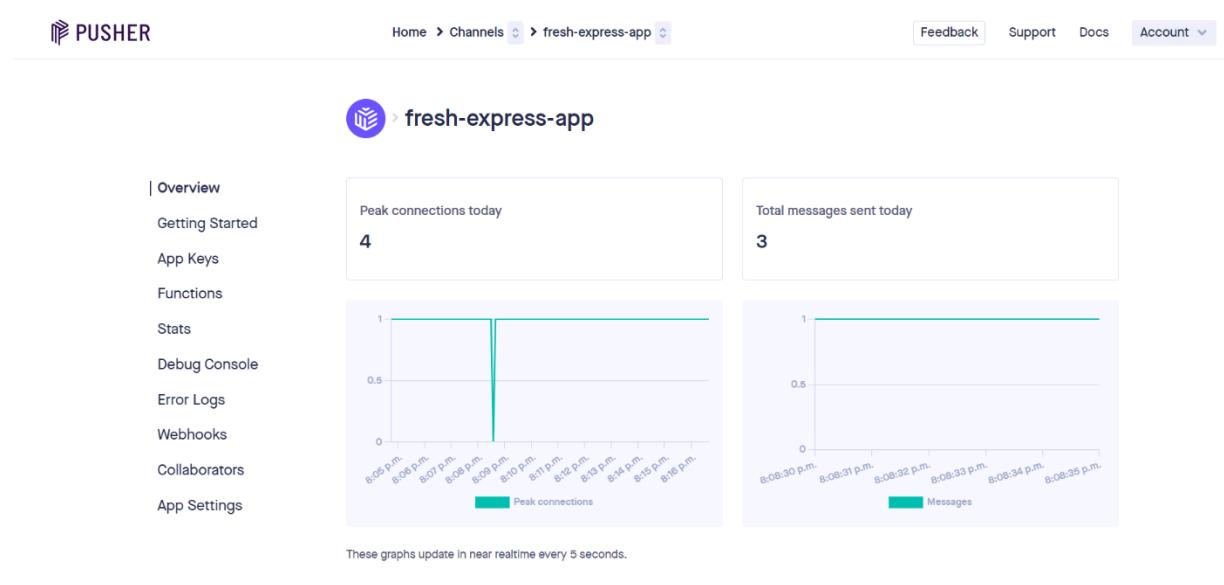
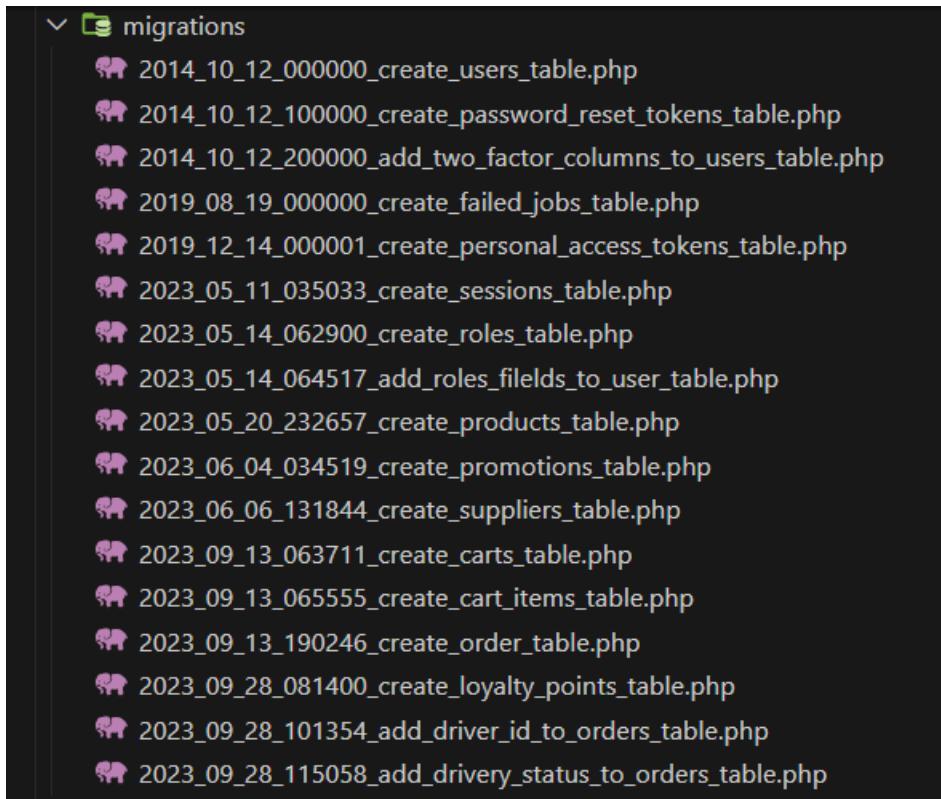


Figure 21 – pusher

Screenshots of the code & postman requests

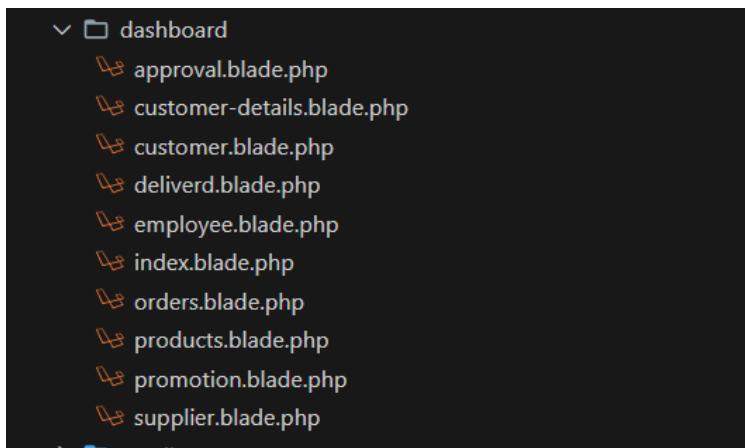
Migration files



```
✓ migrations
  ↗ 2014_10_12_000000_create_users_table.php
  ↗ 2014_10_12_100000_create_password_reset_tokens_table.php
  ↗ 2014_10_12_200000_add_two_factor_columns_to_users_table.php
  ↗ 2019_08_19_000000_create_failed_jobs_table.php
  ↗ 2019_12_14_000001_create_personal_access_tokens_table.php
  ↗ 2023_05_11_035033_create_sessions_table.php
  ↗ 2023_05_14_062900_create_roles_table.php
  ↗ 2023_05_14_064517_add_roles_filelds_to_user_table.php
  ↗ 2023_05_20_232657_create_products_table.php
  ↗ 2023_06_04_034519_create_promotions_table.php
  ↗ 2023_06_06_131844_create_suppliers_table.php
  ↗ 2023_09_13_063711_create_carts_table.php
  ↗ 2023_09_13_065555_create_cart_items_table.php
  ↗ 2023_09_13_190246_create_order_table.php
  ↗ 2023_09_28_081400_create_loyalty_points_table.php
  ↗ 2023_09_28_101354_add_driver_id_to_orders_table.php
  ↗ 2023_09_28_115058_add_drivery_status_to_orders_table.php
```

Figure 22 - Migration Files

Livewire and view files



```
✓ dashboard
  ↗ approval.blade.php
  ↗ customer-details.blade.php
  ↗ customer.blade.php
  ↗ deliverd.blade.php
  ↗ employee.blade.php
  ↗ index.blade.php
  ↗ orders.blade.php
  ↗ products.blade.php
  ↗ promotion.blade.php
  ↗ supplier.blade.php
```

Figure 23 - Blade files

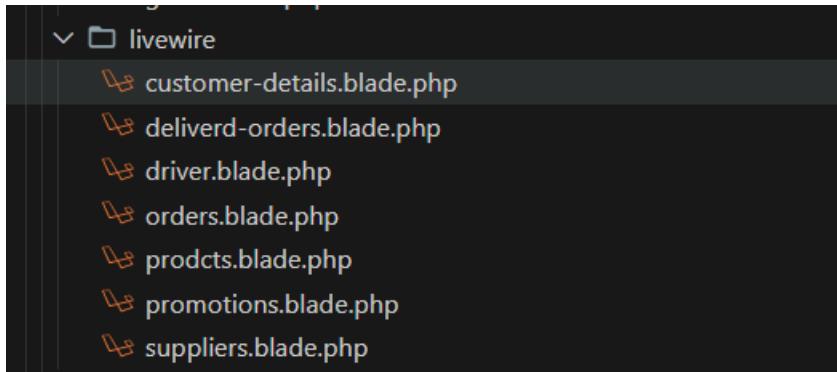


Figure 24 - Livewire files

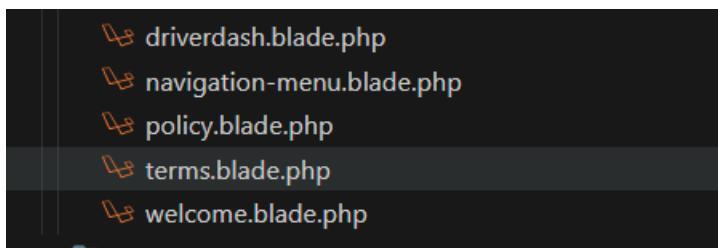


Figure 25 - Blade Files

Controllers

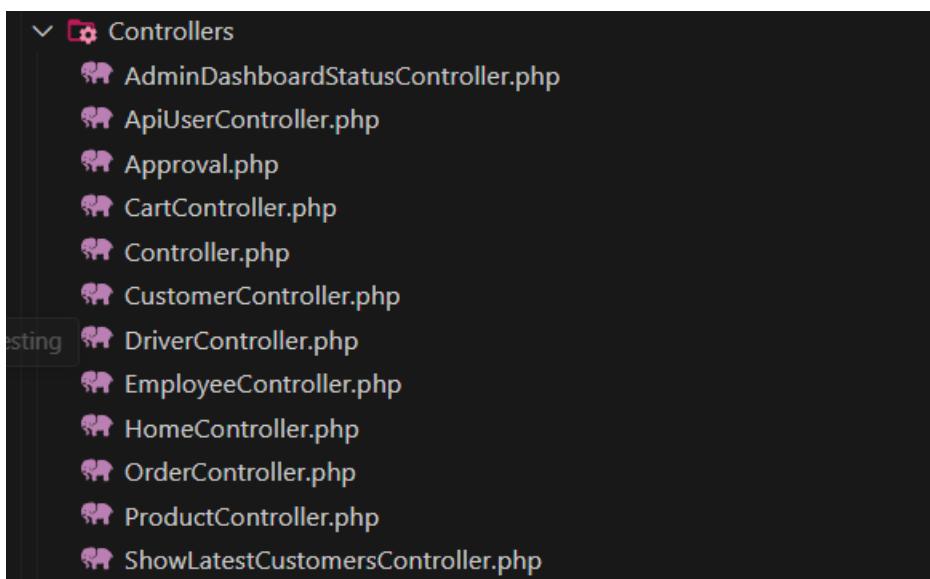


Figure 26 – Controllers

Livewire Controllers

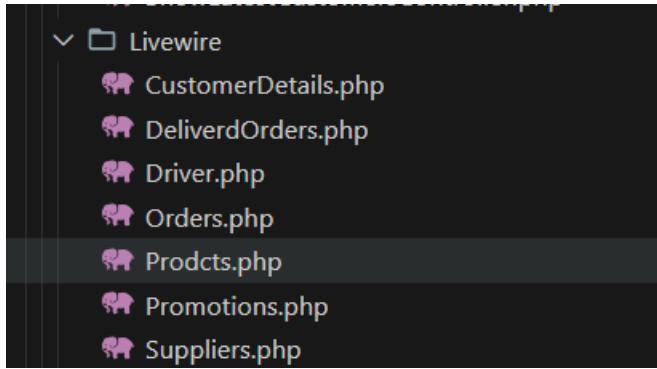


Figure 27 - Livewire Controllers

Models

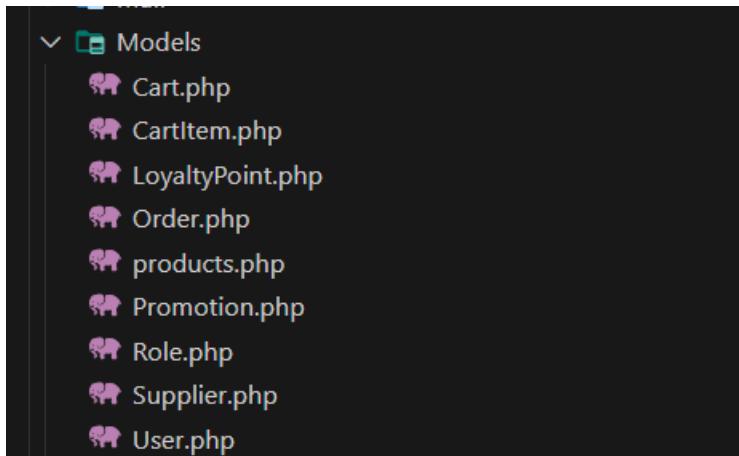


Figure 28 – Models

Cart Controller



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\products;
6 use Illuminate\Http\Request;
7
8 use App\Models\Cart;
9
10 class CartController extends Controller
11 {
12     public function addtoCart(Request $request)
13     {
14
15         $user = $request->user();
16
17         // $product=$request->product_id;
18         // return $product;
19
20         // $quantity=$request->quantity;
21
22         // extract the product id and quantity from the request's json body
23         // $product_id = $request->product_id;
24         // $quantity = $request->quantity;
25
26         // return $product_id;
27
28         //check if the user has a cart
29         $cart = Cart::where('user_id', $user->id)->where('is_paid', false)->first();
30
31
32         //if not create a new cart
33         if (!$cart) {
34             $cart = Cart::create([
35                 'user_id' => $user->id,
36                 'is_paid' => false,
37                 'total' => 0,
38             ]);
39         }
40
41         //check products already in cart
42         $cartItem = $cart->items()->where('product_id', $request->product_id)->first();
43         $product = products::find($request->product_id);
44         $price = $product->price;
45
46         //if product is in cart, increase quantity
47         if ($cartItem) {
48
49             $newTotal = $price * ($cartItem->quantity + $request->quantity);
50             $cartItem->update([
51                 'quantity' => $cartItem->quantity + $request->quantity,
52                 'total' => $newTotal,
53             ]);
54
55         } else {
56
57             // get the product price
58             // $product = products::find($request->product_id);
59             // $price = $product->price;
60
61             //if the product is not in te cart add it
62             $cartItem = $cart->items()->create([
63                 'product_id' => $request->product_id,
64                 'quantity' => $request->quantity,
65                 'total' => $price * $request->quantity,
66             ]);
67
68         }
69         $cart->refresh();
70         //calculate the cart total
71
72         $cartTotal = $cart->items()->sum('total');
73
74
75         //update the cart total
76
77         $cart->update([
78             'total' => $cartTotal,
79         ]);
80
81         return response()->json([
82             'message' => 'Product added to cart',
83             'cart' => $cart,
84         ]);
85
86
87     }
88 }
89 }
```

Figure 29 - Cart Controller

This is the file that I used to get the items from the mobile to the CRM cart when the user add a item it will get added to the users cart , first it checks if there is a existing cart to that user if so then the item get added to the cart , if not it will create new cart and add the items to the cart. And I'm sending a response saying tat the product got added to cart.

The below I have sent API request form postman and it shows the output.

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, and Explore. The main workspace is titled "My Workspace" and contains a collection named "FreshExpress API". A POST request is selected with the URL `http://127.0.0.1:8000/api/add-to-cart`. The "Body" tab is active, showing a JSON payload:

```
1 {"product_id": "4",  
2 "quantity": "100"}  
3  
4
```

The "Test Results" tab shows the response details: Status: 200 OK, Time: 2.66 s, Size: 483 B. The response body is displayed in JSON format:

```
1 {  
2   "message": "Product added to cart",  
3   "cart": {  
4     "id": 29,  
5     "user_id": 8,  
6     "is_paid": 0,  
7     "total": 25000,  
8     "created_at": "2023-10-01T04:32:54.000000Z",  
9     "updated_at": "2023-10-01T04:32:54.000000Z"  
10   }  
11 }
```

Figure 30 - API Add to cart

ApiController



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Promotion;
6 use App\Models\User;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Facades\Hash;
9 use Illuminate\Support\Facades\Hash;
10 use Illuminate\Validation\ValidationException;
11 use Illuminate\Http\Response;
12 use Laravel\Sanctum\Sanctum;
13
14 class ApiController extends Controller
15 {
16     /**
17      *
18     * @param Request $request
19     * @return JsonResponse
20     */
21     public function requestToken(Request $request): JsonResponse
22     {
23         $request->validate([
24             'email' => 'required|email',
25             'password' => 'required',
26             'device_name' => 'required',
27         ]);
28
29         $user = User::where('email', $request->email)->first();
30
31         if (! $user || ! Hash::check($request->password, $user->password)) {
32             throw ValidationException::withMessages([
33                 'email' => ['The provided credentials are incorrect.'],
34             ]);
35         }
36
37         $token = $user->createToken($request->device_name);
38
39         return response()->json([
40             'token' => $token->plainTextToken,
41             'username' => $user->name, // Add the username to the response
42         ]);
43     }
44
45     public function revokeToken(Request $request): string
46     {
47         $request->user()->currentAccessToken()->delete();
48
49         return response()->json(['message' => 'Token revoked']);
50     }
51
52     public function revokeAllTokens(Request $request): string
53     {
54         $request->user()->tokens()->delete();
55
56         return response()->json(['message' => 'Tokens revoked']);
57     }
58
59     public function register(Request $request): JsonResponse
60     {
61         // return response()->json($request);
62
63         echo '';
64
65         $request->validate([
66             'name' => 'required|unique:users,email',
67             'password' => 'required',
68             'confirmPassword' => 'required|same:password',
69             'phone' => 'required|string|min:10|max:10|regex:/^\d{10}/',
70             'address' => 'required|string|max:255',
71             'dob' => 'required',
72             'device_name' => 'required',
73             'date' => [
74                 function ($attribute, $value, $fail) {
75                     if (! DateTime::createFromFormat('Y-m-d', $value)) {
76                         $fail('Invalid date format');
77                     } else {
78                         $dob = Carbon::parse($value);
79                         $now = Carbon::now();
80                         if ($dob->diffInYears($now) <= 18) {
81                             $fail('You must be older than 18 years old to register.');
82                         }
83                     }
84                 },
85             ],
86             'device_name' => 'required',
87         ]);
88
89         $user = User::where('email', $request->email)->first();
90
91         if ($user) {
92             return response()->json(['message' => 'You are already a registered user'], 409);
93         }
94
95         $newUser = new User();
96         $newUser->name = $request->name;
97         $newUser->email = $request->email;
98         $newUser->role_id = 1;
99         $newUser->password = Hash::make($request->password);
100        $newUser->phone = $request->phone;
101        $newUser->address = $request->address;
102        $newUser->dob = $request->dob;
103        $newUser->status = 1;
104
105        $newUser->save();
106
107        $token = $newUser->createToken($request->device_name);
108
109        return response()->json([
110            'token' => $token->plainTextToken,
111            'username' => $newUser->name,
112            // Add the username to the response
113        ]);
114
115        // return response()->json(['message' => 'Registration successful', 'user' => $newUser], 200);
116    }
117}
```

Figure 31 - API API Controller

```

1  public function getCustomerDetails(Request $request): JsonResponse
2  {
3      // Retrieve the customer details based on the API token
4      $user = Sanctum::actingAs($request->user(), ['*']);
5
6      if ($user) {
7          // Return the customer details as JSON response
8          return response()->json([
9              'name' => $user->name,
10             'contactNumber' => $user->phone,
11             'address' => $user->address,
12             'dateOfBirth' => $user->dob,
13             'emailAddress' => $user->email,
14         ]);
15     } else {
16         // Handle the case when customer details are not found
17         return response()->json(['error' => 'Customer details not found'], 404);
18     }
19 }
20
21 public function getPromotions(Request $request): JsonResponse
22 {
23     // Retrieve promotion details from the database
24     $promotions = Promotion::all(); // This fetches all promotions; you can customize the query as needed
25
26     // Check if any promotions were found
27     if ($promotions->isEmpty()) {
28         return response()->json(['message' => 'No promotions found'], 404);
29     }
30
31     // Return the promotion details as a JSON response
32     return response()->json(['promotions' => $promotions]);
33 }
34
35 public function getUserLoyaltyPoints(Request $request): JsonResponse
36 {
37     // Retrieve the authenticated user based on the API token
38     $authenticatedUser = Auth::user();
39
40     // Check if the user is authenticated
41     if (!$authenticatedUser) {
42         return response()->json(['error' => 'User not authenticated'], 401);
43     }
44
45     // Load the authenticated user's loyalty points if it's a relationship
46     $authenticatedUser->load('loyaltyPoints');
47
48     // Calculate the total loyalty points for the authenticated user
49     $totalLoyaltyPoints = $authenticatedUser->loyaltyPoints->sum('points');
50
51     // Create a response with the authenticated user's loyalty points
52     $responseData = [
53         'user_id' => $authenticatedUser->id,
54         'name' => $authenticatedUser->name,
55         'total_loyalty_points' => $totalLoyaltyPoints,
56     ];
57
58     return response()->json($responseData);
59 }
60 }
61

```

Figure 32 - API Controller

In this file I am requesting the Api token to registration and login getting customer details of the user to display in the mobile, get the promotions to, ang get the user loyalty points to display in the mobile as well.

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Environment:** Fresh Express
- Request Type:** POST
- URL:** http://127.0.0.1:8000/api/register
- Body (form-data):**

Key	Value	Description
email	the@gm.com	
password	147147147	
ConfirmPassword	147147147	
phone	074785478	
address	thalawakale	
dob	1997/8/5	
device_name	nokia	
name	thththh	
- Response Status:** 200 OK
- Response Body (Pretty):**

```

1 "token": "66|SGpAKUNrH6Vf0A16Fb3WzZvv1p45sR130qs1gfSy",
2 "username": "thththh"
3
4

```

Figure 33 - API Register

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Header:** Home, Workspaces, API Network, Explore, Search Postman, Invite, Upgrade, etc.
- Request URL:** http://127.0.0.1:8000/api/login
- Method:** POST
- Body (form-data):**

email	thenurihettiarachchi@gmail.com
password	customerpassword
device_name	nokia
- Response:** Status: 200 OK, Time: 1489 ms, Size: 386 B
- Body (Pretty):**

```

1
2   "token": "6514A0W1Bk2SwiSkU1nWnIRuN0m1RwIRv2VNJLInSNB",
3   "username": "Customer11"
4

```

Figure 34 - API Login

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Header:** Home, Workspaces, API Network, Explore, Search Postman, Invite, Upgrade, etc.
- Request URL:** http://127.0.0.1:8000/api/customer/details
- Method:** GET
- Headers:** Authorization (green dot), Headers (7)
- Query Params:**

Key	Value	Description
Key	Value	Description
- Response:** Status: 200 OK, Time: 2.17 s, Size: 454 B
- Body (Pretty):**

```

1
2   "name": "Customer11",
3   "contactNumber": "7894561254",
4   "address": "Kathmandu",
5   "dateOfBirth": "1999-01-02",
6   "emailAddress": "thenurihettiarachchi@gmail.com"
7

```

Figure 35 - API Getting Customer Details

The screenshot shows the Postman interface with a collection named "FreshExpress API". A specific request titled "Get Products" is selected. The URL is set to `http://127.0.0.1:8000/api/user-loyalty-points`. The "Body" tab is active, displaying a JSON response:

```
1
2   "user_id": 8,
3   "name": "Customer11",
4   "total_loyalty_points": 200.40000000000003
```

Figure 36 - API Loyalty Points

The screenshot shows the Postman interface with a collection named "FreshExpress API". A specific request titled "Get Products" is selected. The URL is set to `http://127.0.0.1:8000/api/promotions`. The "Body" tab is active, displaying a JSON response:

```
1
2   "promotions": [
3     {
4       "id": 1,
5       "Name": "PR01",
6       "Percentage": 50,
7       "EndMonth": "June",
8       "created_at": "2023-09-27T04:51:05.000000Z",
9       "updated_at": "2023-09-27T04:51:05.000000Z"
10    },
11    {
12      "id": 2,
13      "Name": "sampa",
14      "Percentage": 48,
15      "EndMonth": "may",
16      "created_at": "2023-09-28T07:44:40.000000Z",
17      "updated_at": "2023-09-28T07:44:40.000000Z"
18    }
19  ]
```

Figure 37 - API Promotions

Order Controller



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Cart;
6 use App\Models\Order;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Str;
9 use App\Events\NewOrderReceivedEvent;
10
11 class OrderController extends Controller
12 {
13     public function completeCart(Request $request, Cart $cart)
14     {
15
16         // dd($cart);
17
18
19         //check if the cart exists and is not already paid
20         if (!$cart || $cart->is_paid) {
21             return response()->json(['message' => 'Cart not found or already completed'], 404);
22         }
23
24         // $address = $request->delivery_address;
25         // $ZIP_code = $request->ZIP_code;
26         // $city = $request->city;
27         // $delivery_type = $request->delivery_type;
28         // return $address;
29         //Make the cart as completed (is_paid = true)
30
31         // Find the highest order number in the database and increment it by 1
32         $latestOrderNumber = Order::max('id');
33         if ($latestOrderNumber) {
34             $orderNumber = 'order' . $latestOrderNumber + 1;
35         } else {
36             $orderNumber = 'order1';
37         }
38         // return ($latestOrderNumber);
39
40
41         $deliveryFee = 50;
42         $totalAmount = $cart->total + $deliveryFee;
43         //Create an order
44         $order = Order::create([
45             'user_id' => $cart->user_id,
46             'cart_id' => $cart->id,
47             'total' => $totalAmount,
48             'status' => 'pending',
49             'delivery_status' => 'pending',
50             'delivery_address' => $request->delivery_address,
51             'ZIP_code' => $request->ZIP_code,
52             'city' => $request->city,
53             'delivery_type' => $request->delivery_type,
54             'order_number' => $orderNumber,
55         ]);
56         //calculate loyalty points based on the total amount
57         $loyaltyPointsEarned = $totalAmount*0.001;
58         //Add the loyalty points to the user
59         $user = $cart->user;
60         $user->loyaltyPoints()->create([
61             'points' => $loyaltyPointsEarned,
62             'order_id' => $order->id,
63         ]);
64         $cart->update([
65             'is_paid' => true,
66         ]);
67
68         $response = [
69             'message' => 'cart marked as completed',
70             'order_number' => $orderNumber,
71             'cart_total' => $cart->total,
72             'delivery_fee' => $deliveryFee,
73             'total_amount' => $totalAmount,
74             'order_created_date' => $order->created_at->toDateString(),
75             'loyalty_points_earned' => $loyaltyPointsEarned,
76         ];
77
78         event((new NewOrderReceivedEvent()));
79
80         return response()->json($response, 200);
81
82
83
84     }
85 }
```

Figure 38 - Order Controller

In the order controller I am making the cart as an order and saving it in the database , and also I am calculating the loyalty points as well ,after that I am sending the details as a response.

The below are the Api request that sent through the postman and the outputs

POST {{base_url}}/api/complete-cart/29

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Token

61|e1zW2J3mmM7f5BybuavmIgdXwzhlpC

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```

1 "message": "cart marked as completed",
2 "order_number": "order28",
3 "cart_total": 25000,
4 "delivery_fee": 50,
5 "total_amount": 25050,
6 "order_created_date": "2023-10-01",
7 "loyalty_points_earned": 25.00
8
9

```

Status: 200 OK Time: 4.79 s Size: 492 B Save as Example

Figure 39 – Order

And there are two more API requests that I am sending to the Api.

```
// Get product categories
Route::get('/products/categories', [ProductController::class, 'getProductCategories']);
```

GET {{base_url}}/api/products/categories

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Token

({{auth_token}})

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```

1 [
2   "categories": [
3     "vegetables",
4     "fruits",
5     "meat",
6     "fish"
7   ]
8

```

Status: 200 OK Time: 2.37 s Size: 350 B Save as Example

Figure 40 - Getting Product categories.

```
// Get products by category name
Route::get('/products/category/{categoryName}', [ProductController::class, 'getProductsByCategoryName']);
```

The screenshot shows the Postman application interface. At the top, there is a code editor window containing the provided PHP code. Below it, the Postman interface displays a collection named "FreshExpress API" with a single endpoint named "Get Products". The endpoint is set to a GET method with the URL `((base.url))/api/products/category/Meat`. The "Authorization" tab is selected, showing a "Bearer Token" type with the token value `((auth_token))`. The "Body" tab is selected, showing a JSON response with the following structure:

```
1  "products": [
2    {
3      "id": 8,
4      "name": "Chicken",
5      "price": 250,
6      "image": "products/YPNY5669Gc6BaR01rAnxF9fbagSLxMyhxNASM6jw.png",
7      "category": "Meat"
8    }
9  ]
10 ]
11 ]
```

The status bar at the bottom indicates a successful response with a status of 200 OK, a time of 646 ms, and a size of 442 B.

Figure 41 - Getting Products by category name.

Route Files (API.PHP & WEB.PHP)

```
1 <?php
2
3 use App\Events\PusherSample;
4 use App\Http\Controllers\Approval;
5 use App\Http\Controllers\AdminDashboardStatusController;
6 use App\Http\Controllers\CustomerApproval;
7 use App\Http\Controllers\DashboardController;
8 use App\Http\Controllers\EmployeeController;
9 use App\Http\Controllers\HomeController;
10 use Illuminate\Routing\Controllers\Middleware;
11 use Illuminate\Support\Facades\Route;
12 use App\Http\Controllers\HomeController;
13
14 /*
15 | -----
16 | Web routes
17 |
18 |
19 | Here is where you can register web routes for your application. These
20 | routes are loaded by the RouteServiceProvider and all of them will
21 | be assigned to the "web" middleware group. Make something great!
22 |
23 */
24
25 // Route::get('dashboard', function () {
26 //     return view('welcome');
27 // });
28 // Route::get('/admin/approval', [AdminController::class, 'approval'])->name('admin.approval');
29
30 Route::get('/dev', function () {
31     event(new NewOrderReceivedEvent(100));
32 });
33
34
35 Route::get('/', [HomeController::class, 'checkRoleId']);
36
37 Route::post('/admin/change-status/{user}', [Approval::class, 'switchStatus'])->
38     ->name('admin.approve')->middleware('adminOrEmployee');
39
40 // Route::post('/admin/change-status/{user}', [CustomerApproval::class, 'switch'])-
41 //     ->name('admin.customerApprove')->middleware('adminOrEmployee');
42
43 Route::middleware(['auth'])->group(function () {
44     Route::resource('dashboard/customer', CustomerController::class)->name('index', 'customer');
45     Route::resource('dashboard/employee', EmployeeController::class)->name('index', 'employee');
46     // Route::resource('dashboard/customer/approval', CustomerApproval::class)->name('index', 'customerAproval');
47     Route::view('dashboard/orders', 'dashboard.orders')->name('orders');
48     Route::view('dashboard/products', 'dashboard.products')->name('products');
49 });
50
51 // Route::get('customers/search', [CustomerController@search])->name('customers.search');
52 // Route::resource('customers', 'CustomerController');
53
54 // Route::get('/admin', function () {
55 //     return view('dashboard.index');
56 // })->name('admin.dashboard')
57 // ->middleware(['adminOrEmployee', 'auth']);
58
59 Route::get('/admin', [AdminDashboardStatusController::class, 'index'])
60     ->name('admin.dashboard')
61     ->middleware(['adminOrEmployee', 'auth']);
62
63 // Route::get('/latest-customers', [ShowLatestCustomersController::class, 'showLatestCustomers'])
64 //     ->name('latest-customers')
65 // ->middleware(['adminOrEmployee', 'auth']);
66
67 Route::get('/employee', [AdminDashboardStatusController::class, 'index'])
68     ->name('employee.dashboard')
69     ->middleware(['adminOrEmployee', 'userApproved']);
70
71 // Route::get('/employee', function () {
72 //     return view('dashboard.index');
73 // })->name('employee.dashboard')
74 // ->middleware(['adminOrEmployee', 'userApproved']);
75
76 Route::get('/drive', function () {
77     return view('drivelist');
78 })->name('drive.dashboard')
79     ->middleware(['userApproved']);
80
81 Route::get('/customer', function () {
82     return view('customerdash');
83 })->name('customer.dashboard');
84
85
86 Route::middleware([
87     'auth:sanctum',
88     config('jetstream.auth_session'),
89     'verified'
90 ])->group(function () {
91     Route::get('/', [HomeController::class, 'checkRoleId']);
92 });
93
94 Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
95     Route::get('/dashboard/products', function () {
96         return view('dashboard.products');
97     })->name('products');
98 });
99
100 Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
101     Route::get('/dashboard/promotion', function () {
102         return view('dashboard.promotion');
103     })->name('promotion');
104 });
105
106 Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
107     Route::get('/dashboard/supplier', function () {
108         return view('dashboard.supplier');
109     })->name('supplier');
110 });
111
112 //view customer details
113 Route::get('/customerdetails/{user_id}', function ($userId) {
114     return view('dashboard.customer-details', compact('userId'));
115 })->name('customerdetails');
116
117 Route::get('/deliverorders', function () {
118     return view('dashboard.deliverd');
119 })->name('deliverorders');
```

Figure 42 – WEB.php

```
1 <?php
2
3 use App\Http\Controllers\OrderController;
4 use Illuminate\Http\Request;
5 use Illuminate\Support\Facades\Route;
6 use App\Http\Controllers\ApiUserController;
7 use App\Http\Controllers\ProductController;
8 use App\Http\Controllers\ApiCartController;
9 use App\Http\Controllers\CartController;
10
11
12 /*
13 | -----
14 | API Routes
15 | -----
16 |
17 |
18 | Here is where you can register API routes for your application. These
19 | routes are loaded by the RouteServiceProvider and all of them will
20 | be assigned to the "api" middleware group. Make something great!
21 |
22 */
23
24 Route::middleware('auth:sanctum')->post('/user', function (Request $request) {
25     return $request->user();
26 });
27
28
29 Route::post('login', [ApiController::class, 'requestToken']);
30
31 Route::post('register', [ApiController::class, 'register']);
32
33 // Route::get('getCustomerDetails', [ApiController::class, 'getCustomerDetails']);
34 Route::get('/products', [ProductController::class, 'index']);
35 // Get product categories
36 Route::get('/products/categories', [ProductController::class, 'getProductCategories']);
37
38
39 // Get products by category name
40 Route::get('/products/category/{categoryName}', [ProductController::class, 'getProductsByCategoryName']);
41
42 //Cart
43 // Route::post('/add-to-cart', 'CartController@addToCart')->middleware('api.token');
44
45
46 // Route::group(['middleware' => 'api.token'], function () {
47 //     Route::post('/add-to-cart', 'CartController@addToCart');
48
49 // });
50
51 // Route::middleware('auth:sanctum')->post('/add-to-cart', function (Request $request) {
52 //     return $request->user();
53
54 // });
55
56 //adding the items to cart
57 Route::middleware('auth:sanctum')->post('/add-to-cart', [CartController::class, 'addCart']);
58
59 //creating an order and gettin the order details
60 Route::middleware('auth:sanctum')->post('/complete-cart/{cart}', [OrderController::class, 'completeCart']);
61
62
63
64 // Route::post('/complete-cart/{cart}', 'OrderController@completeCart');
65
66
67
68
69 // http://127.0.0.1:8000/api/products/categories/Fruit/
70
71
72
73
74
75 // send hello
76 Route::get('hello', function () {
77     return 'Hello World';
78 });
79 //getting customer details to display in the profile page
80 Route::middleware(['auth:sanctum'])->group(function () {
81     Route::get('/customer/details', [ApiController::class, 'getCustomerDetails']);
82     Route::get('user-loyalty-points', [ApiController::class, 'getUserLoyaltyPoints']);
83
84 });
85
86 //getting all the promotions
87 Route::get('/promotions', [ApiController::class, 'getPromotions']);
88
89
```

Figure 43 - API.php

Assumptions

To develop this website, I have assumed that there is only one location since this is an online shop.

Future Upgrade plane

I have identified several key areas that can be upgraded and enhanced to make the CRM more competitive and adaptable in the long term. Those are as follows:

Personalized Email and Messages:

Our long-term strategy's main concept is improving consumer involvement. Personalized email and message campaigns are one of the significant changes I have coming up. With this improvement, I have made huge progress toward giving the consumers a personalized and interesting shopping experience.

Personalization is more than just a mere slogan; it's a dynamic strategy including a greater understanding of the clients. The system may provide personalized product offers and promotions that appeal to everyone by examining client preferences and purchasing behavior. The customers feel cherished and understood thanks to this level of customization, which goes beyond basic marketing tactics.

Think about a consumer who regularly purchases organic produce. We can send them individualized messages with special offers and discounts on their preferred goods.

Enhanced Reporting and Analytics

I intend to integrate advanced reporting and analytics capabilities into the CRM system to fully realize its potential. The staff will benefit from greater understanding of customer behavior, inventory control, and sales trends thanks to this improvement.

Making data-driven decisions is essential for staying ahead in the competitive world of supermarket retail. We will offer real-time dashboards that depict crucial indicators thanks to our advanced reporting capabilities. For instance, we can keep tabs on which products are popular, comprehend seasonal shopping trends, and spot inventory bottlenecks. We can improve the product offerings and inventory management techniques using these findings.

the analytics capabilities will also include the ability to comprehend customer behavior. Within the CRM, we can track customer journeys and identify user drop-off points.

Customer Feedback and Review System

Creating a thorough mechanism for receiving client reviews and comments. Customers will have a platform thanks to this modification to express their opinions on goods, shipping experiences, and general order satisfaction.

It is impossible to exaggerate the value of feedback. Beyond just star ratings, customer evaluations offer qualitative input that allows us to improve the quality of our services and product offers.

Adding a review system also shows the clients that the organization is trustworthy and transparent. They can view the unvarnished, truthful thoughts of other customers, which may positively affect their purchasing decisions. Reviews serve as social proof and bolster the authority of our brand.

Additionally, this approach will let the staff quickly respond to any problems or issues that clients may have.

Expanding as a SaaS solution

Branding and customization: The SaaS system I was considering offers the client branding and customization choices, giving them to brand the CRM to fit their own identity.

Scalable infrastructure: As more users sign up for the system, I thought of on load balancing and a scalable infrastructure to guarantee excellent performance and availability.

Marketing and sales strategy: I thought of developing a marketing strategy for the users who has grocery shops, supermarkets, and other similar businesses. This includes online advertising as well as industry events.

Test Cases

Test case ID	Description	Input	Expected Output	Actual Output	Result
T01	Adding Driver to a deliver an order	Select a driver	Order has goes to the driver dashboard	Order went to the driver dashboard	pass
T02	Clicking on the dispatch button in the new order table in the order tab without selecting a driver	Select a driver “ ”	A validation error should display saying. “The selected driver field is required.”	A validation appeared. “The selected driver field is required.”	pass
T03	Sending the order to previous order table after dispatching	Click on the dispatch button	The order should go to the previous order table.	The order went to the previous order table.	pass
T04	Canceling an order	Click on the cancel button	The order should go to the cancel order table	The order went to the cancel order table	pass
T05	Displaying the order as delivered in	Click on the deliver button,	The order should go to the previous order table in	The order got displayed as delivered by going to the	pass

	the driver dashboard		the driver dashboard	previous order table	
T06	Displaying the delivered orders in the delivered order tab	Click in the “derived orders” tab	The order that has been delivered should display	The orders that have been delivered got displayed.	pass
T07	After an order is dispatched, that order will get displayed in the customer purchase history	Click on the dispatch button	The order should go to the customers purchase history	The order displays as a purchase history	pass
T08	When the driver tries to access the admin dashboard by URL	/Dashboard	Redirect to the login page	Got redirect to the login page.	pass
T09	Search order by order ID	Order1	Display the order	Display the order	pass
T10	Search customer by customer name	Sampath	Display the customer	Display the customer	pass
T11	Seach customer by customer address	thalawathugoda	Display the customer	Display the customer	pass

T12	Search order by ZIP CODE	1457	Display the order	Display the order	pass
T13	If the customer has not purchased any items	-	In the address Book the delivery address should be displayed as “No delivery address found”	In the address Book the delivery address should be displayed as “No delivery address found”	pass
T15	If the customer has not purchased any items	-	The loyalty point should be displayed as 0	The loyalty point should be displayed as 0	pass
T16	Displaying total number of employees	Registering to the system	The number count should be increase by 1 in the dashboard	The number count should be increase by 1 in the dashboard	pass
T17	Displaying total number of divers	Registering to the system	The number count should be increase by 1 in the dashboard	The number count should be increase by 1 in the dashboard	pass

Performance Testing

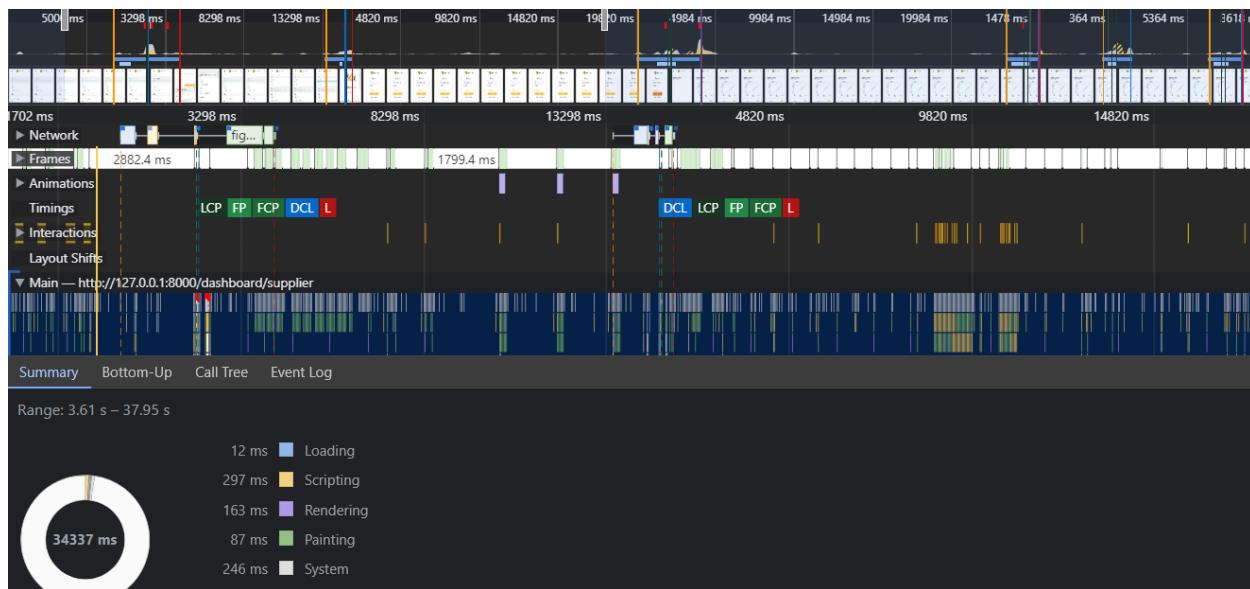


Figure 44 - Performance testing

LightHouse Testing

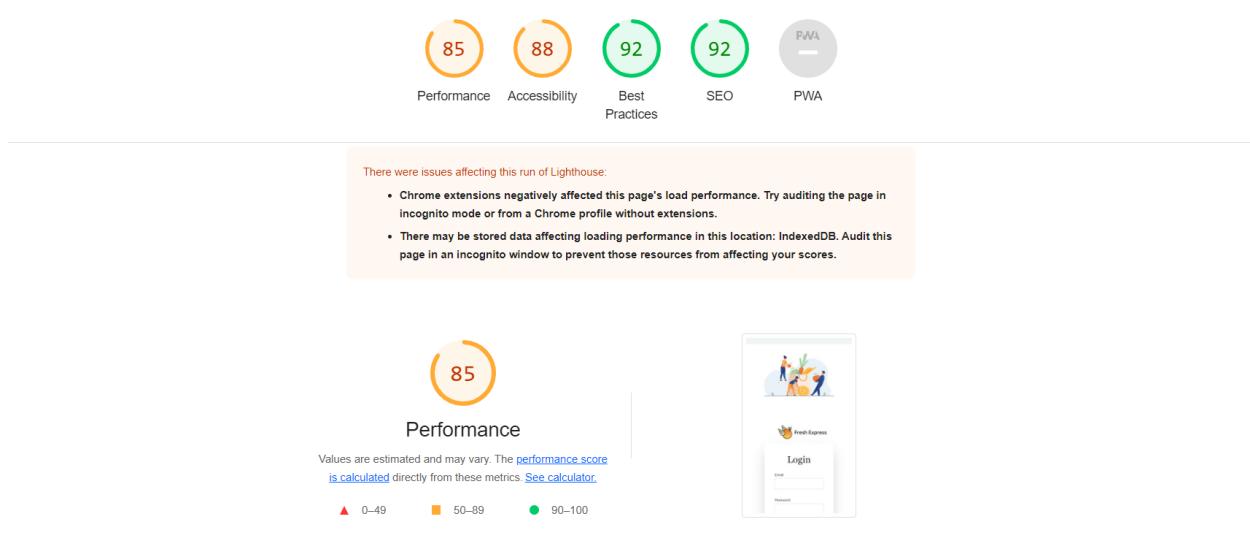


Figure 45 - Lighthouse testing

Conclusion

Displayed website helps the company to manage inventory and track their weekly sales based on the customer purchases. In an online platform efficient management of resources would only achieve business success. Proposed system helps the business to improve their sales, enhance customer satisfaction as well as proper management of internal possess. I have taken all possible steps in the system to exclude system flaws as well as to make the system user friendly.

The screenshots of the system as well as the postman API request to get the customer information and promotions get customer order etc, are included. Also, I have finally, I have included the mind map and GitHub link to my project.