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

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 out 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.

Alphine 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

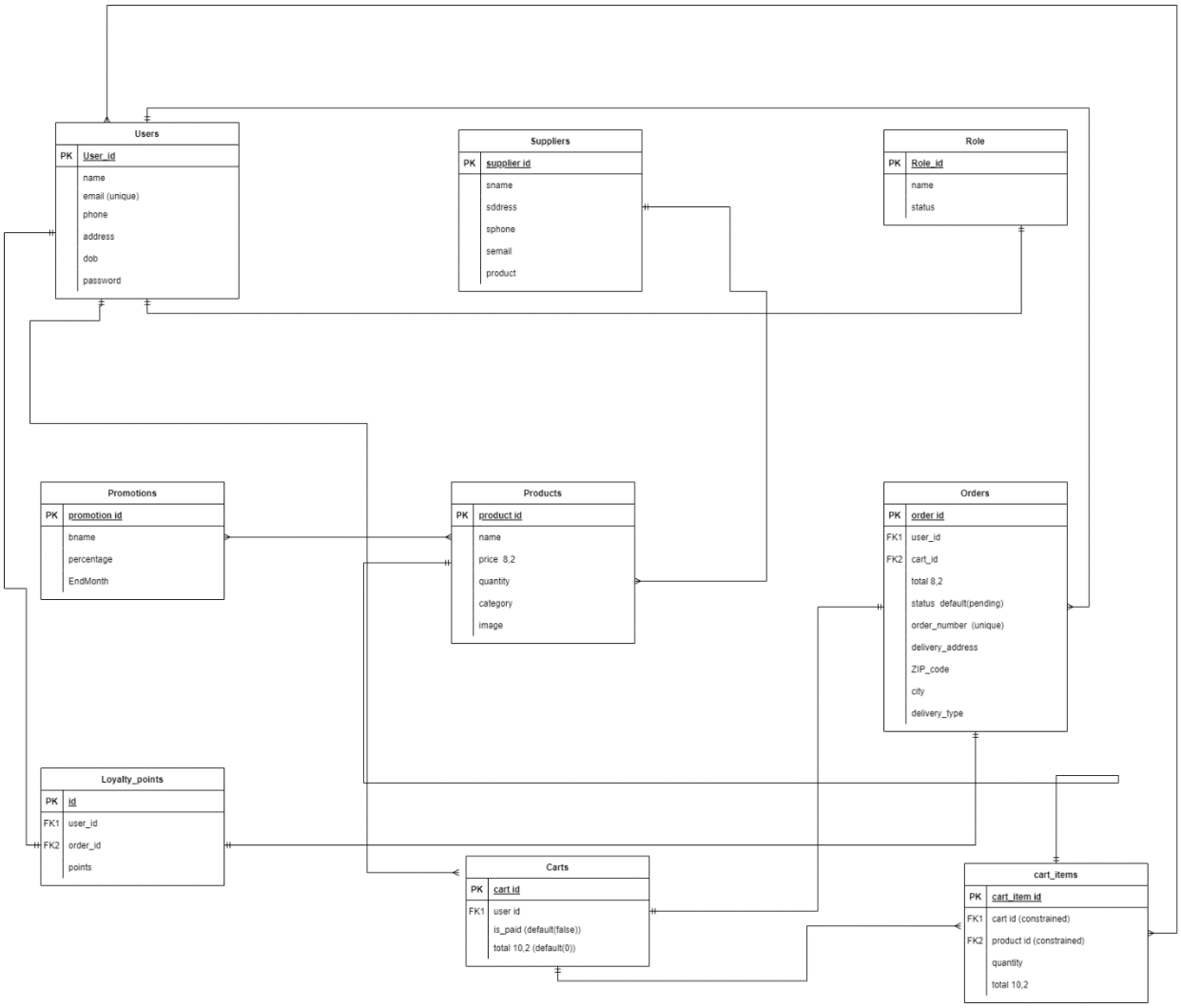
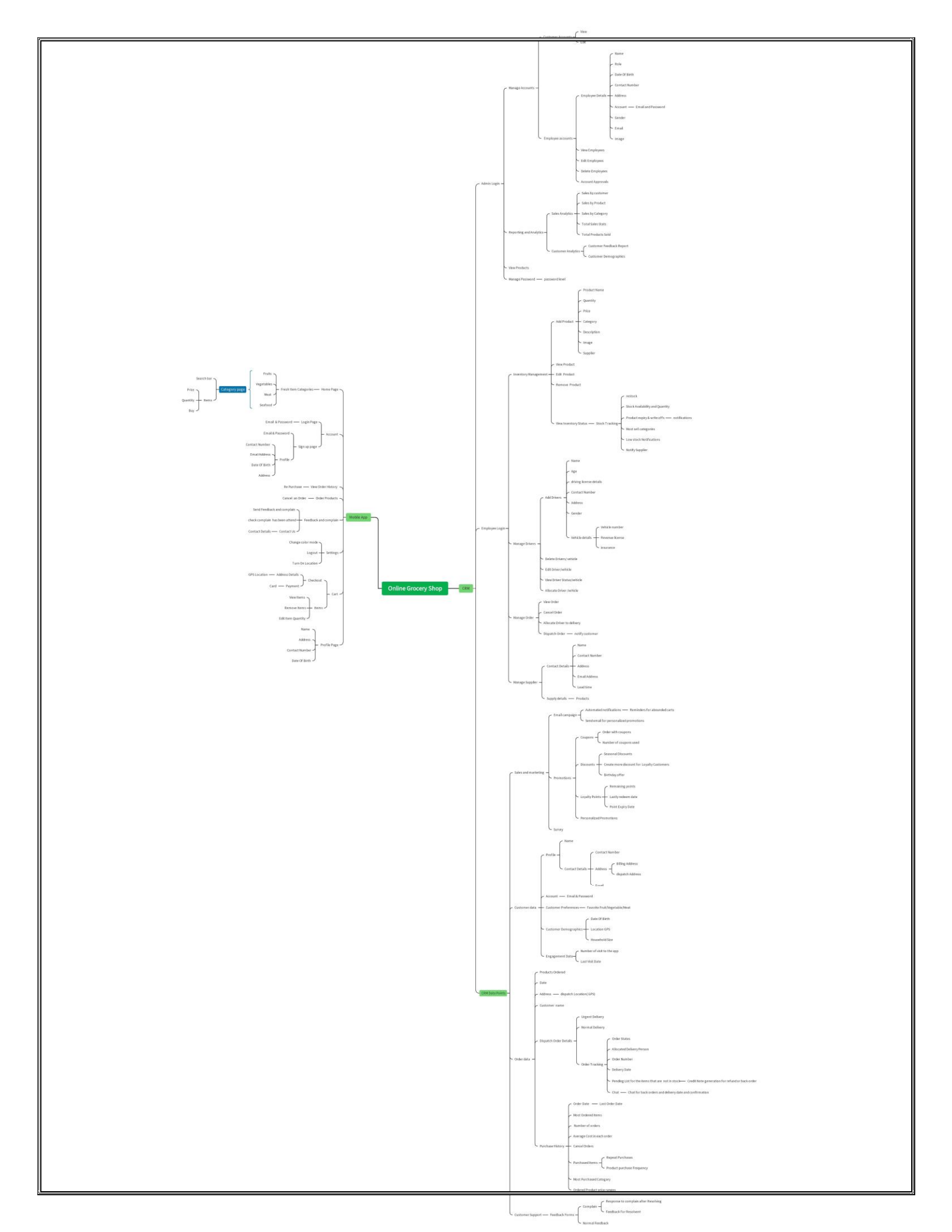
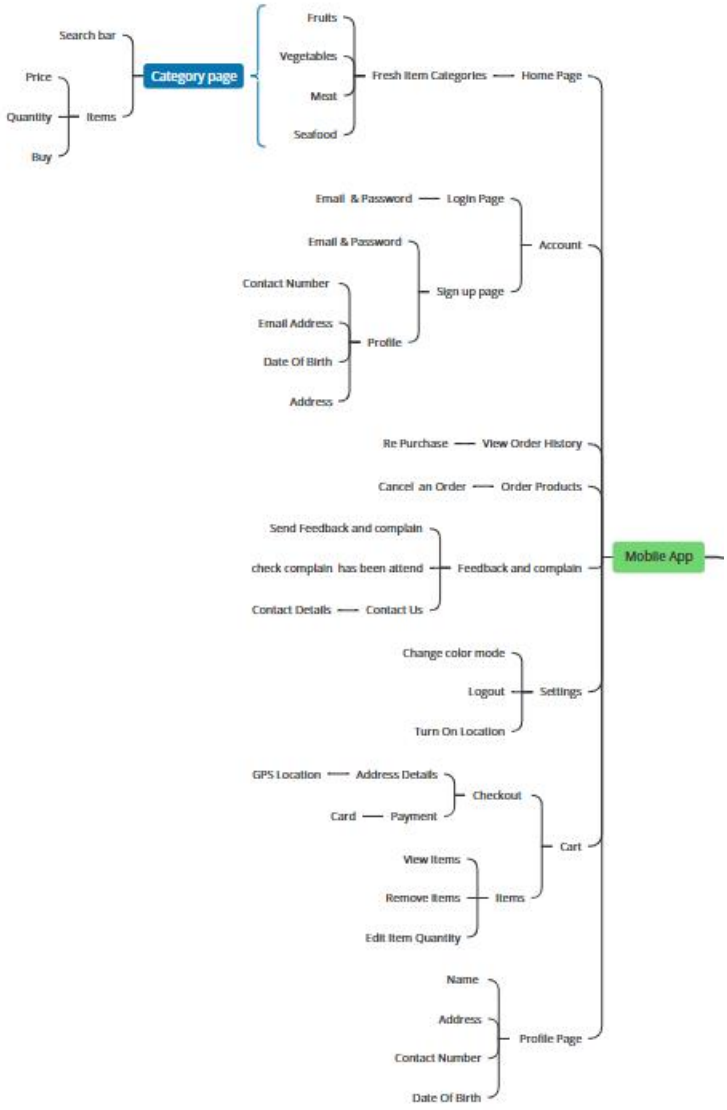


Figure 2 – ER Diagram

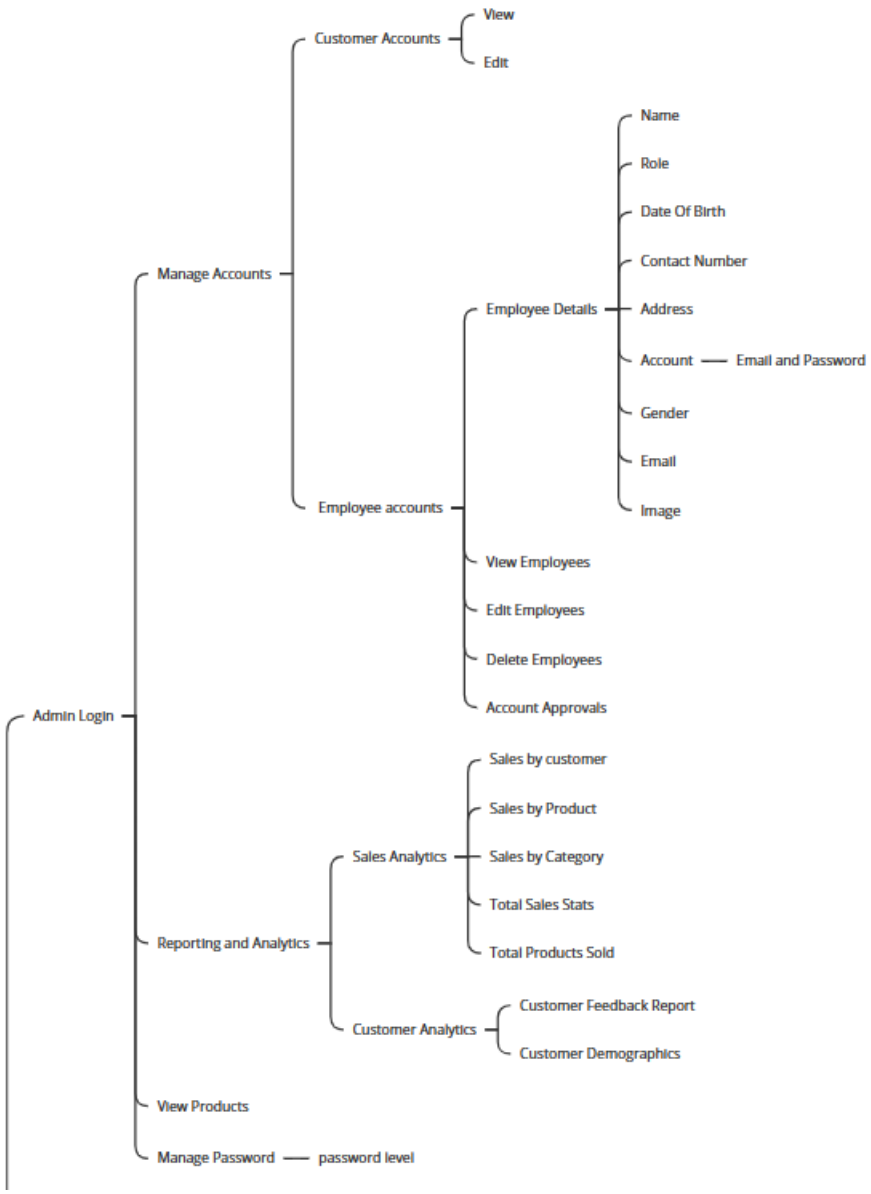
Mind Map


Online Grocery Shop
Mobile App
Search bar
Price
Quantity
Buy
Items
Category page
Fruits
Vegetables
Meat
Seafood
Fresh Item Categories
Home Page
Email & Password
Login Page
Account
Email & Password
Sign up page
Contact Number
Email Address
Date Of Birth
Address
Profile
Re Purchase
View Order History
Cancel on Order
Order Products
Send Feedback and complain
check complain has been attend
Contact Details
Contact Us
Feedback and complain
Change color mode
Logout
Turn On Location
Settings
GPS Location
Address Details
Checkout
Card
Payment
View Items
Remove Items
Edit Item Quantity
Items
Cart
Name
Address
Contact Number
Date Of Birth
Profile Page
CRM
Admin Login
Manage Accounts
View
Edit
Name
Role
Date Of Birth
Contact Number
Address
Account
Email and Password
Gender
Email
Image
Employee Details
Employee accounts
View Employees
Edit Employees
Delete Employees
Account Approvals
Sales by customer
Sales by Product
Sales by Category
Total Sales Stats
Total Products Sold
Sales Analytics
Customer Feedback Report
Customer Demographics
Customer Analytics
Reporting and Analytics
View Products
Manage Password
password level
Product Name
Quantity
Price
Category
Description
Image
Supplier
Add Product
View Product
Edit Product
Remove Product
restock
Stock Availability and Quantity
Product expiry & write offs
notifications
Most sell categories
Low stock Notifications
Notify Supplier
View Inventory Status
Stock Tracking
Inventory Management
Employee Login
Name
Age
driving license details
Contact Number
Address
Gender
Add Drivers
Vehicle number
Revenue license
Insurance
Vehicle details
Manage Drivers
Delete Drivers/vehicle
Edit Driver/vehicle
View Driver Status/vehicle
Allocate Driver /vehicle
View Order
Cancel Order
Allocate Driver to delivery
Dispatch Order
notify customer
Manage Order
Name
Contact Number
Address
Email Address
Lead time
Contact Details
Supply details
Products
Manage Supplier
Automated notifications
Reminders for abounded carts
Send email for personalized promotions
Email campaign
Order with coupons
Number of coupons used
Coupons
Seasonal Discounts
Create more discount for Loyalty Customers
Birthday offer
Discounts
Remaining points
Lastly redeem date
Point Expiry Date
Loyalty Points
Promotions
Personalized Promotions
Survey
Sales and marketing
Name
Contact Number
Billing Address
dispatch Address
Address
Email
Contact Details
Profile
Account
Email & Password
Favorite Fruit/Vegetable/Meat
Customer Preferences
Date Of Birth
Location GPS
Household Size
Customer Demographics
Number of visit to the app
Last Visit Date
Engagement Data
Customer data
Products Ordered
Date
Address
dispatch Location(GPS)
Customer name
Urgent Delivery
Normal Delivery
Dispatch Order Details
Order States
Allocated Delivery Person
Order Number
Delivery Date
Pending List for the items that are not in stock
Credit Note generation for refund or back order
Chat
Chat for back orders and delivery date and confirmation
Order Tracking
Order Data
Order Date
Last Order Date
Most Ordered Items
Number of orders
Average Cost in each order
Cancel Orders
Repeat Purchases
Product purchase Frequency
Purchased Items
Most Purchased Category
Ordered Product price ranges
Purchase History
CRM Data Points
Response to complain after Resolving
Feedback For Resolvent
Normal Feedback
Complain
Feedback Forms
Customer Support

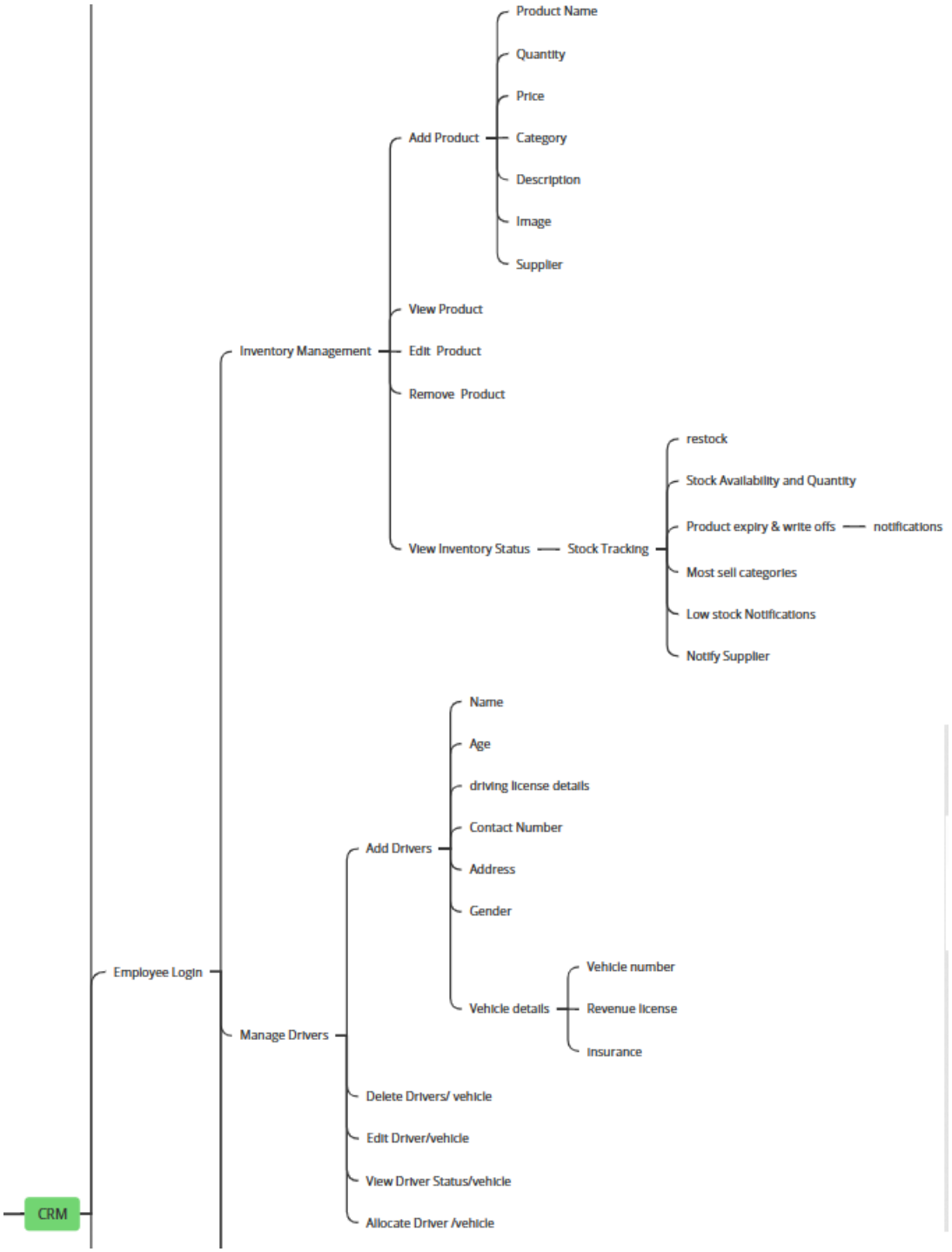
Mobile App

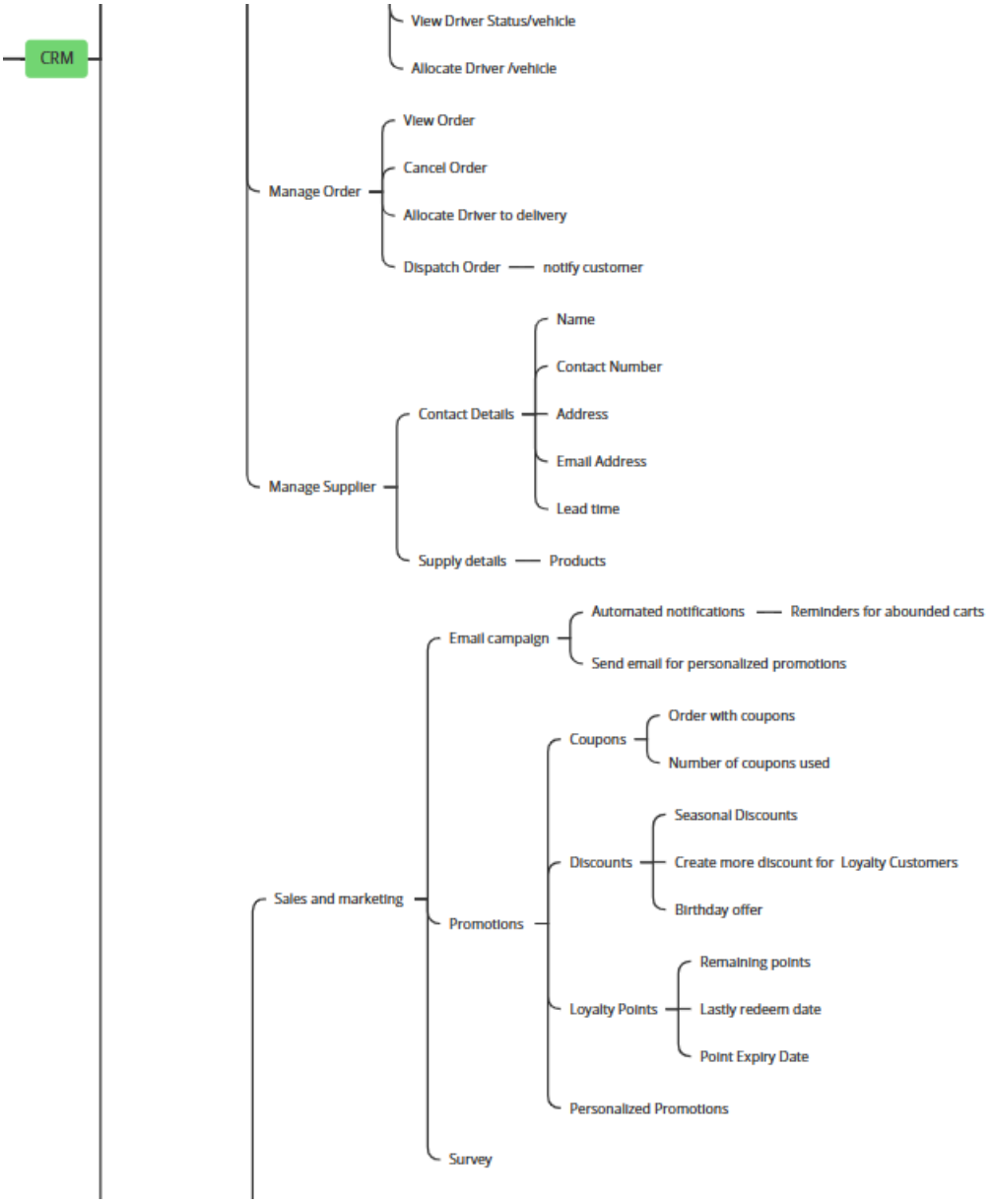

Mobile App
Home Page
Fresh Item Categories
Fruits
Vegetables
Meat
Seafood
Category page
Search bar
Items
Price
Quantity
Buy
Account
Login Page
Email & Password
Sign up page
Email & Password
Profile
Contact Number
Email Address
Date Of Birth
Address
Order Products
View Order History
Re Purchase
Cancel an Order
Feedback and complain
Send Feedback and complain
check complain has been attend
Contact Us
Contact Details
Settings
Change color mode
Logout
Turn On Location
Cart
Checkout
Address Details
GPS Location
Payment
Card
Items
View Items
Remove Items
Edit Item Quantity
Profile Page
Name
Address
Contact Number
Date Of Birth

CRM

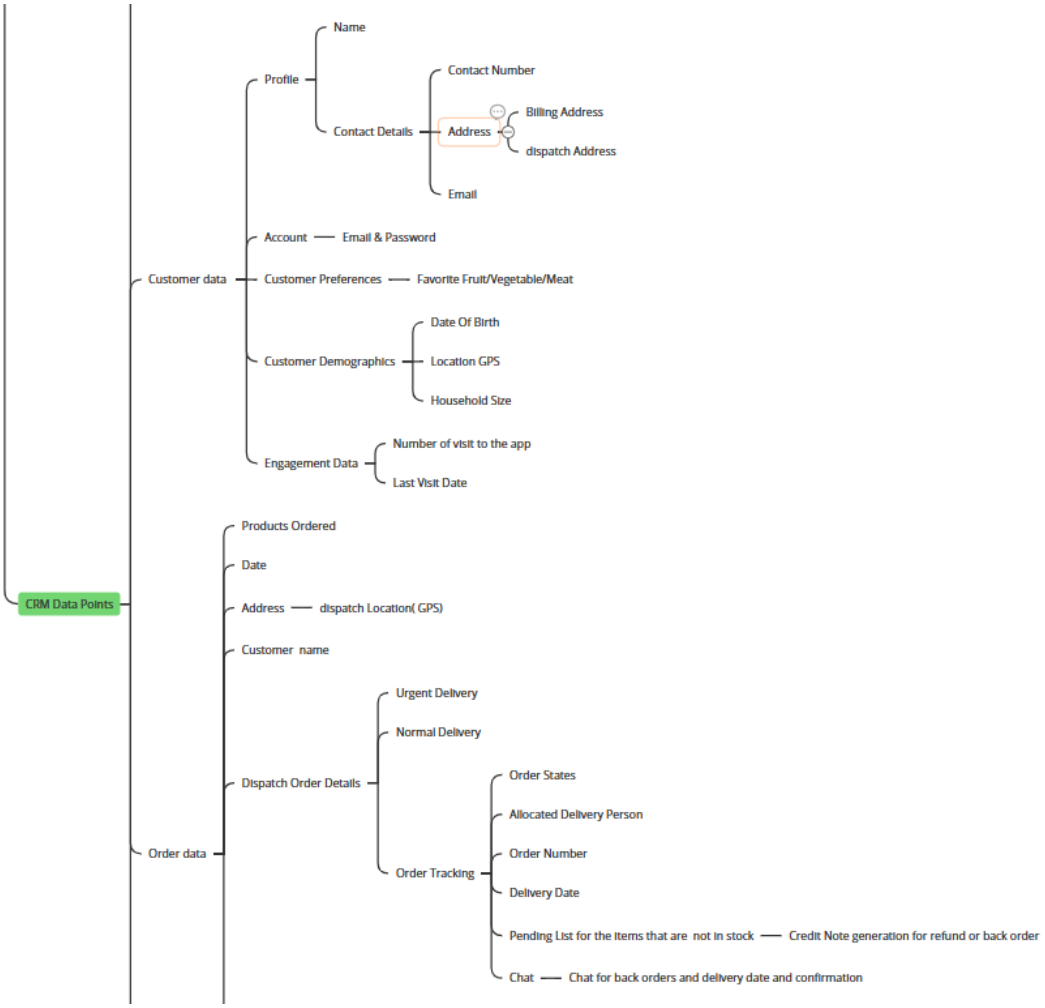

Admin Login
Manage Accounts
Customer Accounts
View
Edit
Employee accounts
Employee Details
Name
Role
Date Of Birth
Contact Number
Address
Account
Email and Password
Gender
Email
Image
View Employees
Edit Employees
Delete Employees
Account Approvals
Reporting and Analytics
Sales Analytics
Sales by customer
Sales by Product
Sales by Category
Total Sales Stats
Total Products Sold
Customer Analytics
Customer Feedback Report
Customer Demographics
View Products
Manage Password
password level

CRM


CRM
Employee Login
Inventory Management
Add Product
Product Name
Quantity
Price
Category
Description
Image
Supplier
View Product
Edit Product
Remove Product
View Inventory Status
Stock Tracking
restock
Stock Availability and Quantity
Product expiry & write offs
notifications
Most sell categories
Low stock Notifications
Notify Supplier
Manage Drivers
Add Drivers
Name
Age
driving license details
Contact Number
Address
Gender
Vehicle details
Vehicle number
Revenue license
Insurance
Delete Drivers/ vehicle
Edit Driver/vehicle
View Driver Status/vehicle
Allocate Driver /vehicle


CRM
View Driver Status/vehicle
Allocate Driver /vehicle
Manage Order
View Order
Cancel Order
Allocate Driver to delivery
Dispatch Order
notify customer
Manage Supplier
Contact Details
Name
Contact Number
Address
Email Address
Lead time
Supply details
Products
Sales and marketing
Email campaign
Automated notifications
Reminders for abounded carts
Send email for personalized promotions
Promotions
Coupons
Order with coupons
Number of coupons used
Discounts
Seasonal Discounts
Create more discount for Loyalty Customers
Birthday offer
Loyalty Points
Remaining points
Lastly redeem date
Point Expiry Date
Personalized Promotions
Survey

CRM Data Points



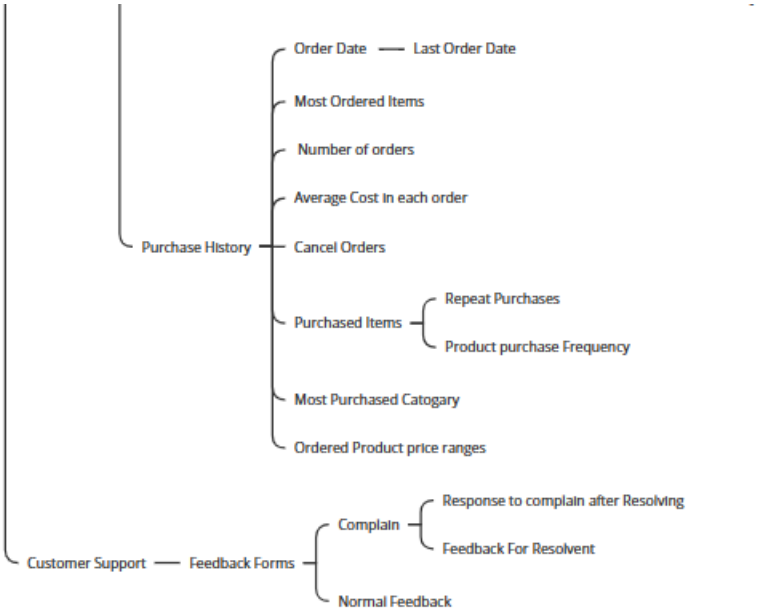


Figure 3 - Mind Map

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

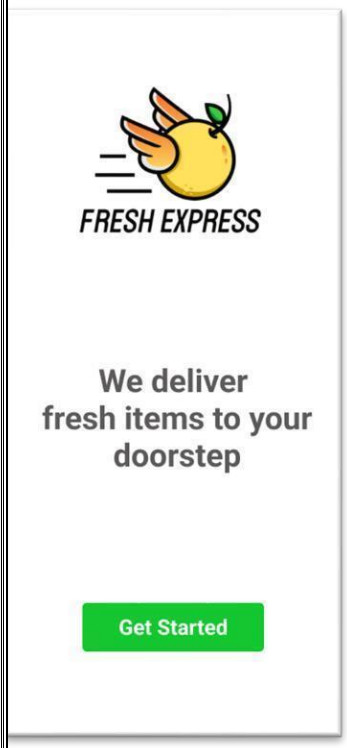
- Login
- Register
- Can see customer details
- Can see the drivers and other employees in the system
- Add Promotions
- Add Products
- Dispatch an order

Driver

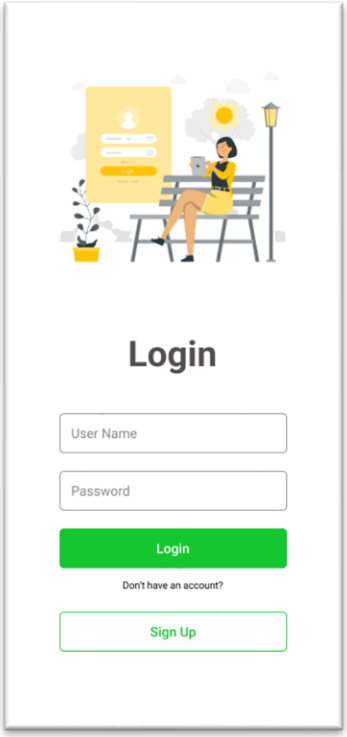
- Can see the customer orders that should be delivered

Screen shots of the mobile app

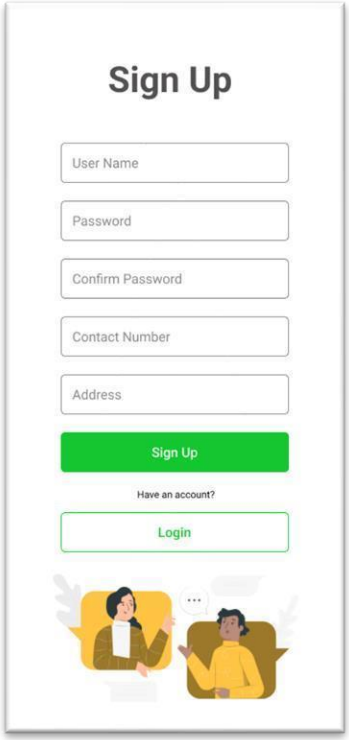
Main Page



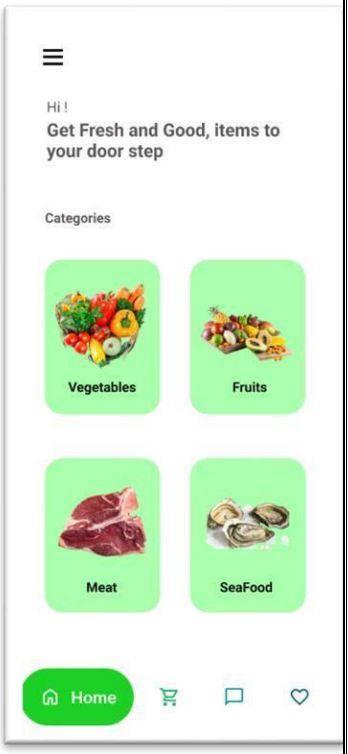
Login Page



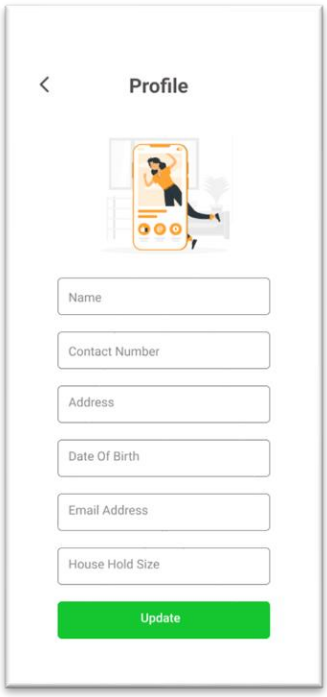
Sign Up Page



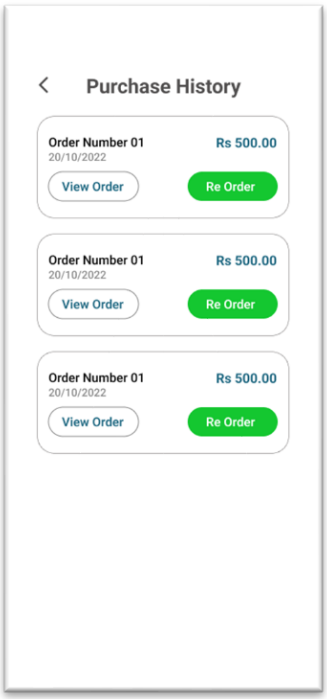
Home Page



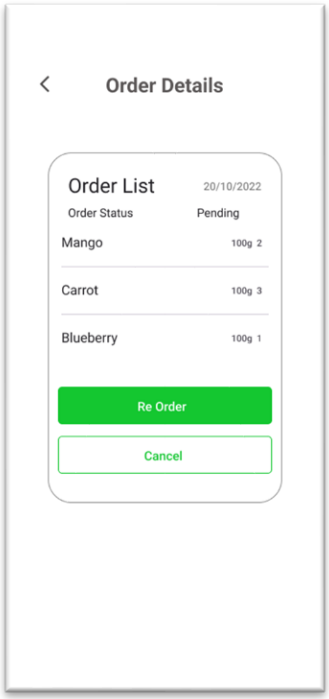
Profile



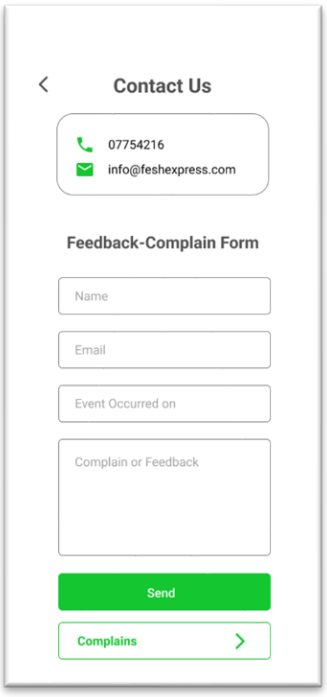
Order History



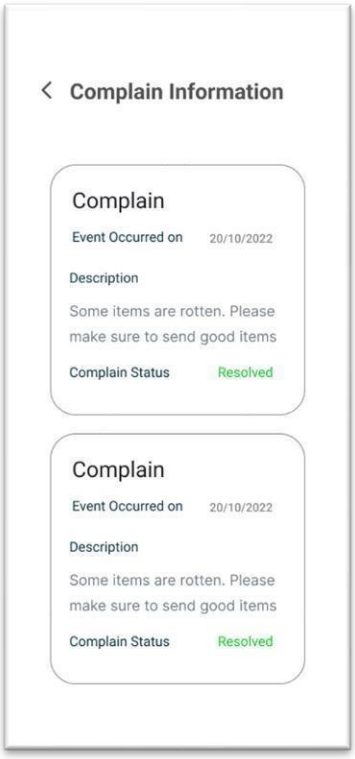
Order Details



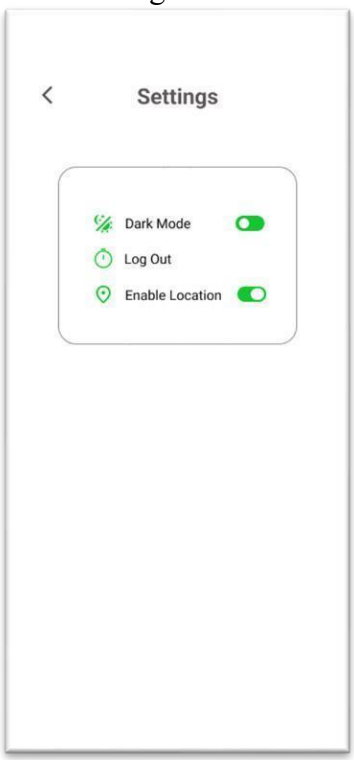
Contact Us



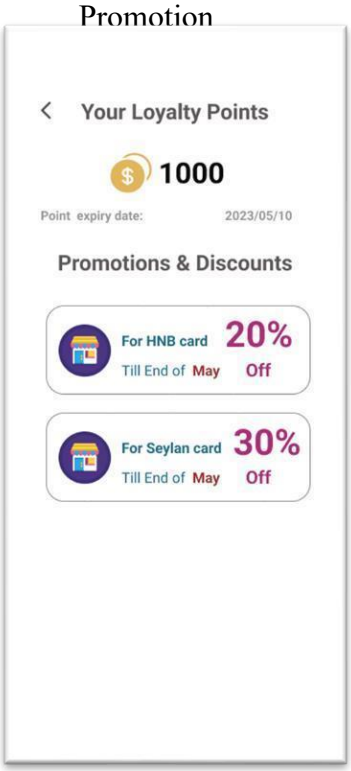
Complains



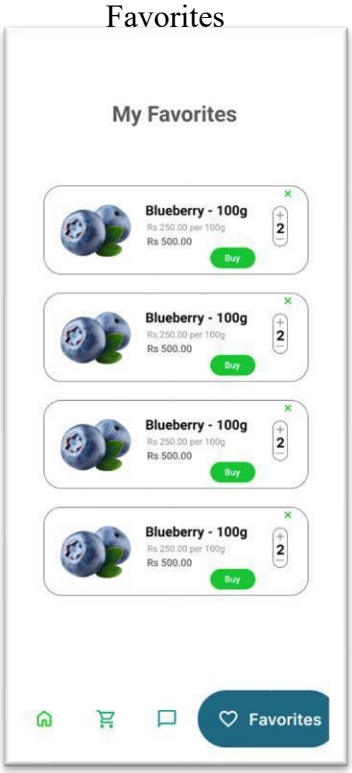
Settings



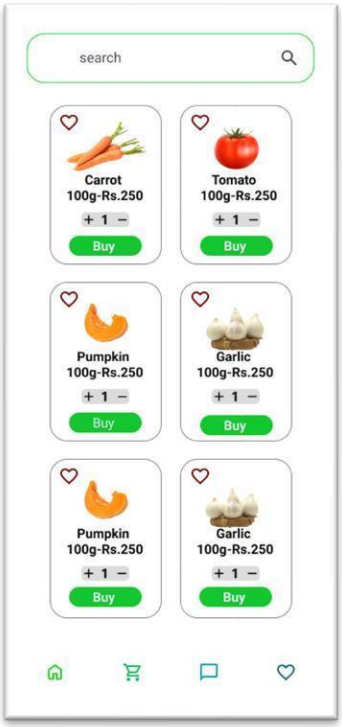
Promotion

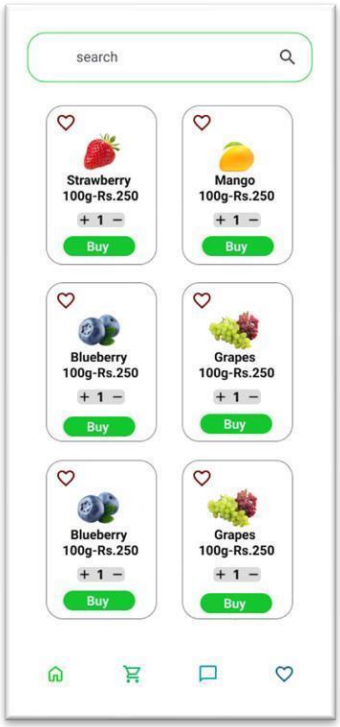


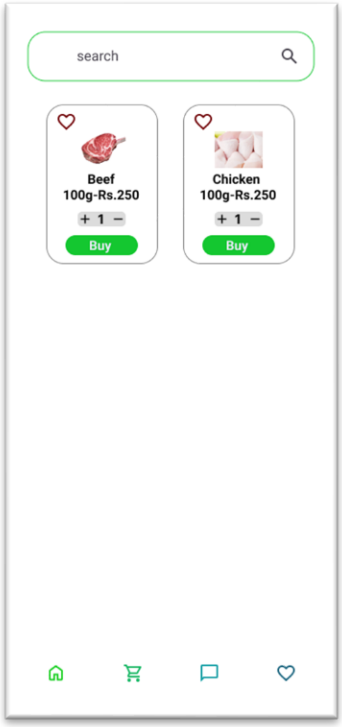
Favorites

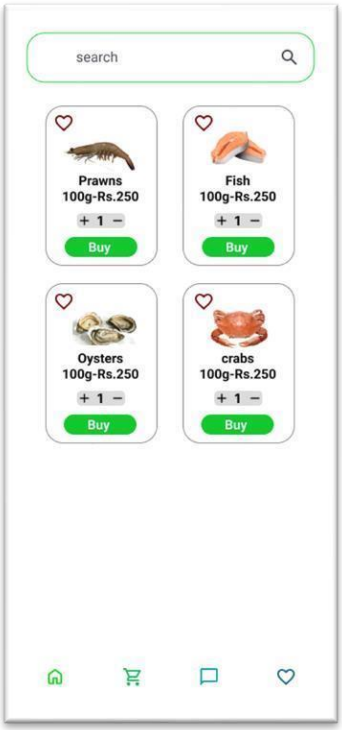


Item Pages

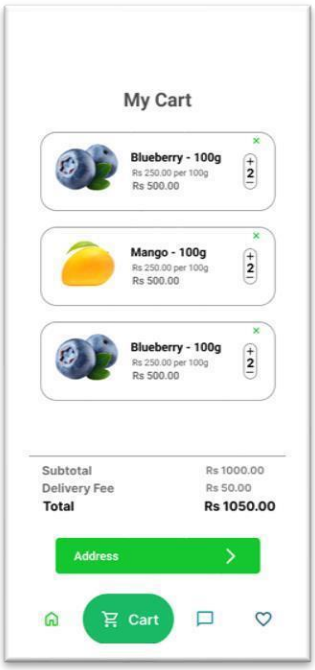




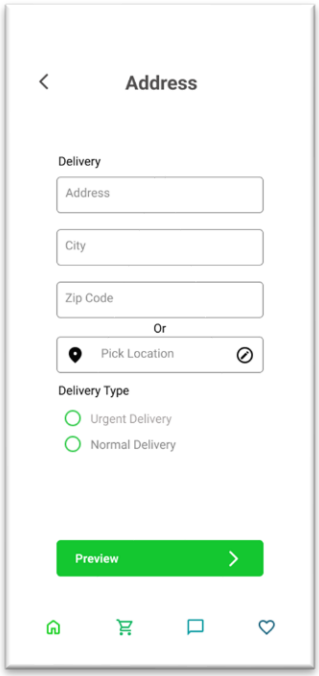




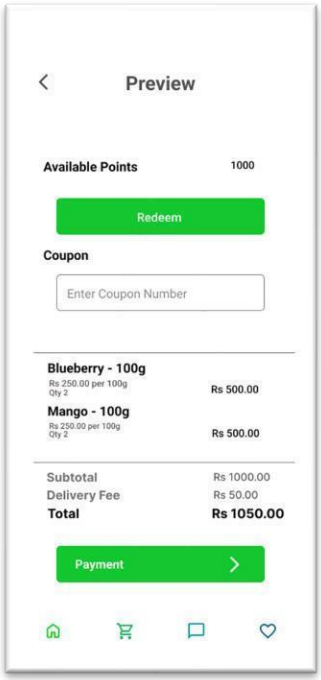
Cart



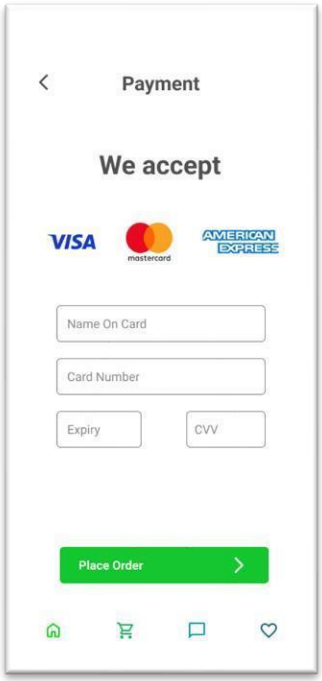
Address Details



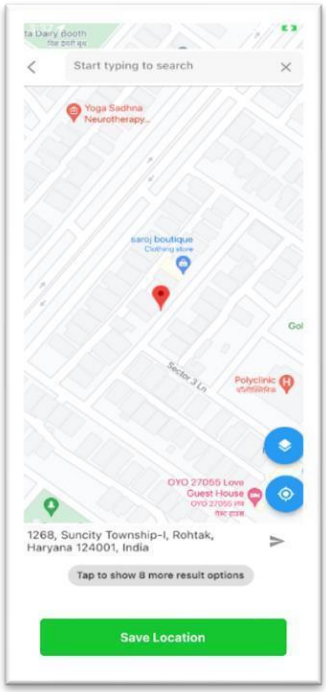
Preview



Payment



Map



Thank you Page

Chat

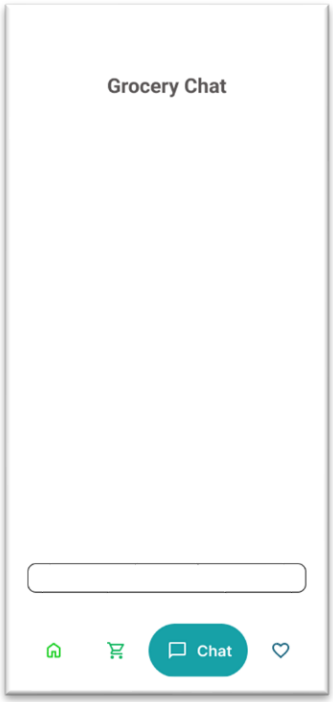


Figure 4 - Mobile App Design

Screen shots of the CRM system

Login Page

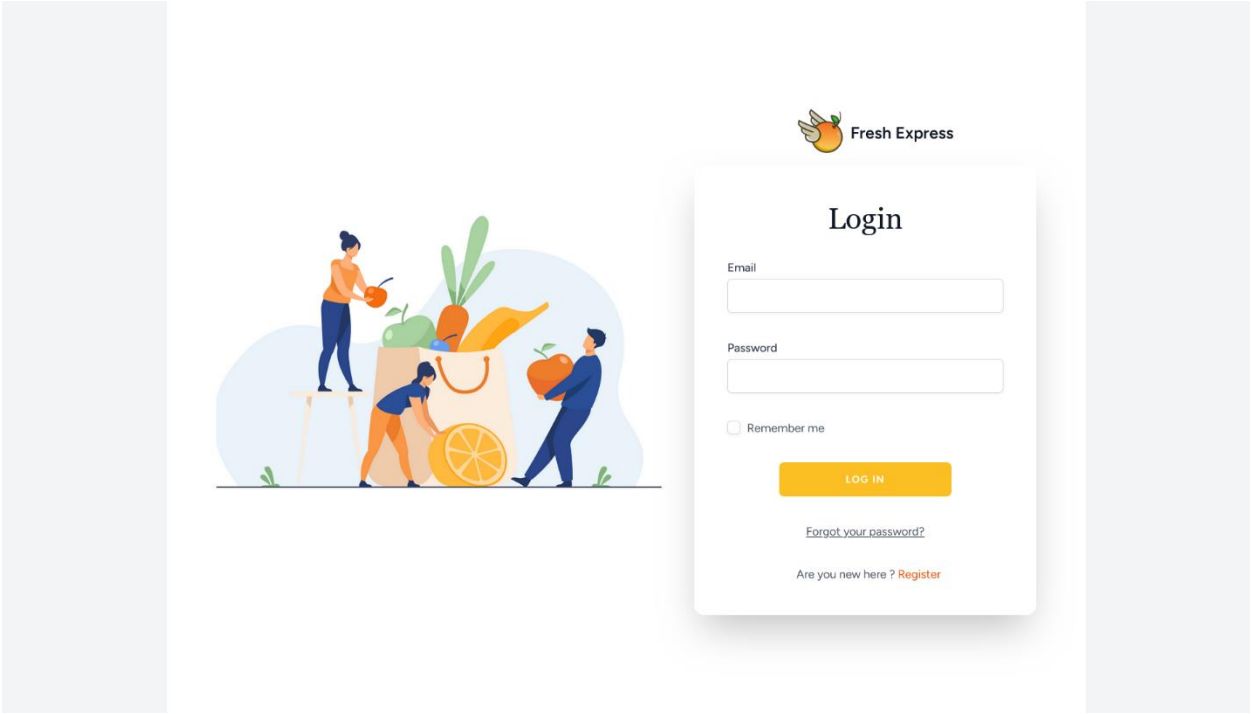


Figure 5-CRM Login page

Registration Page

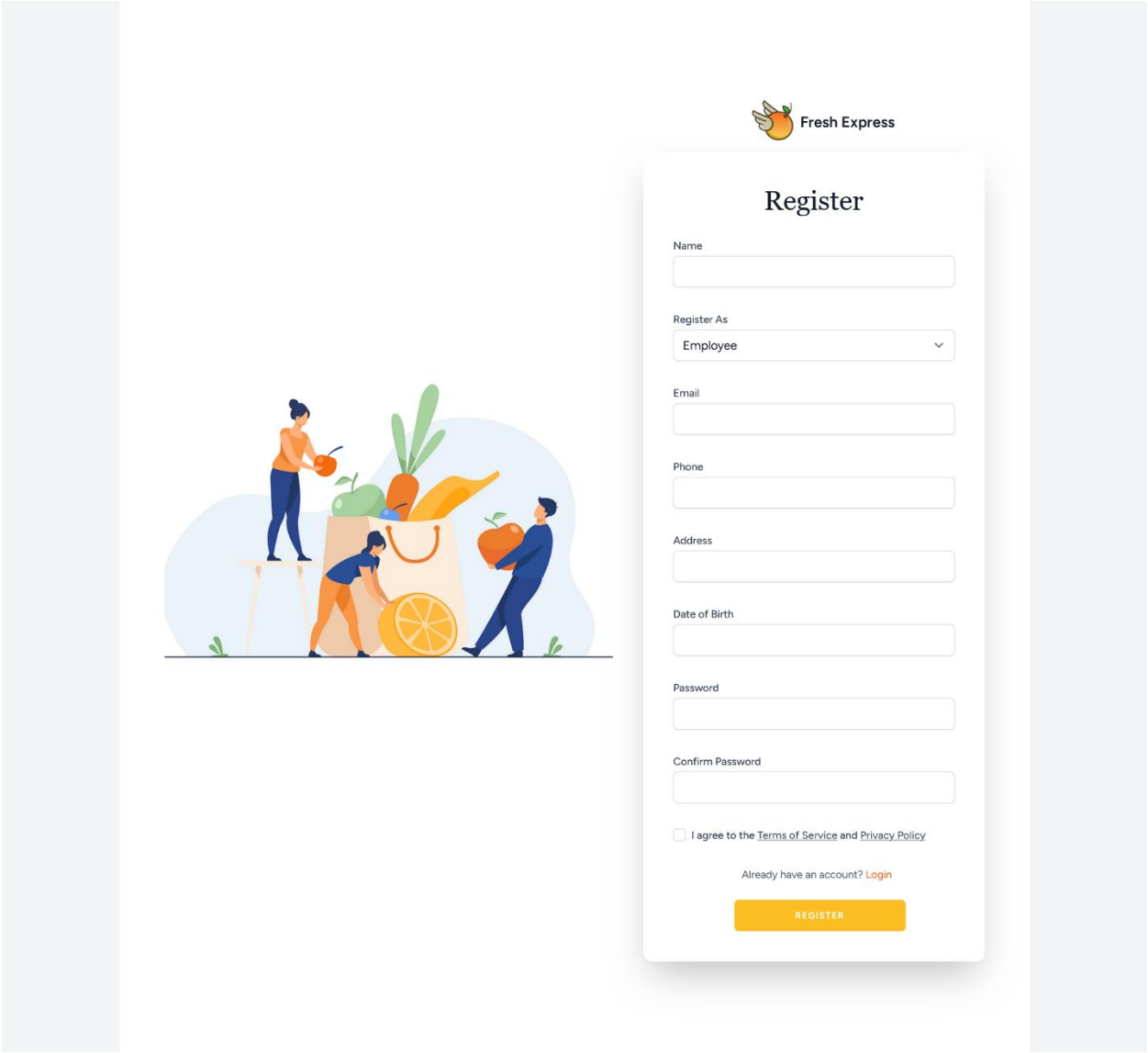


Figure 6-CRM Registration Page

Dashboard (Admin & Employee)

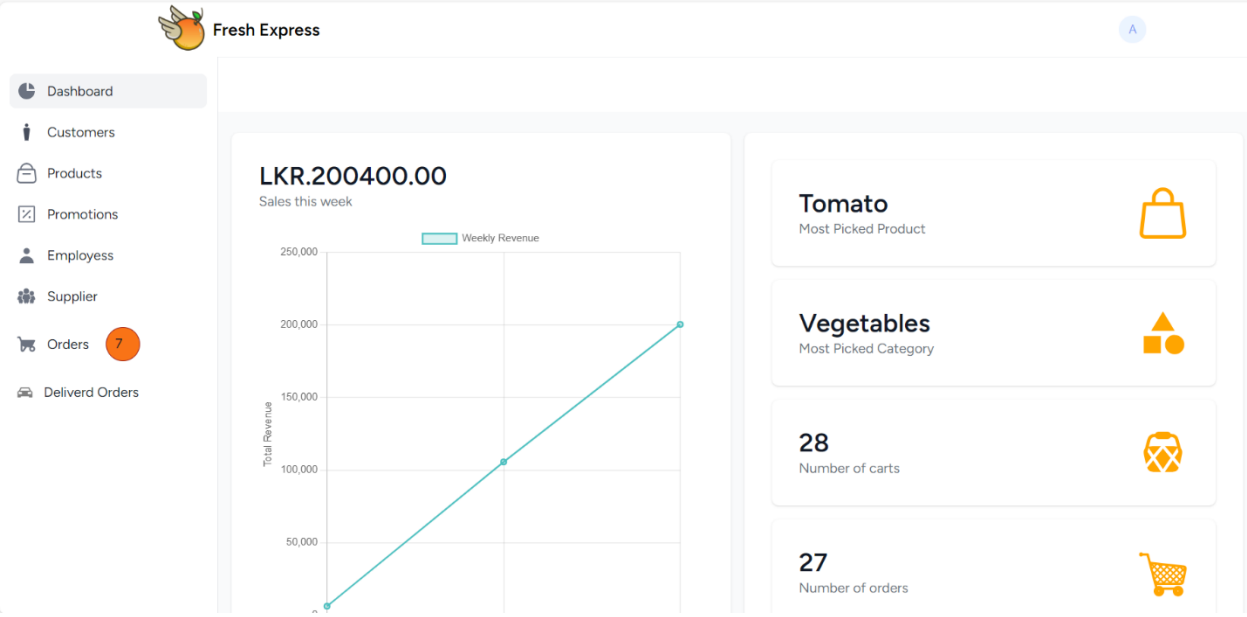


Figure 7 - Admin Employee Dashboard part1

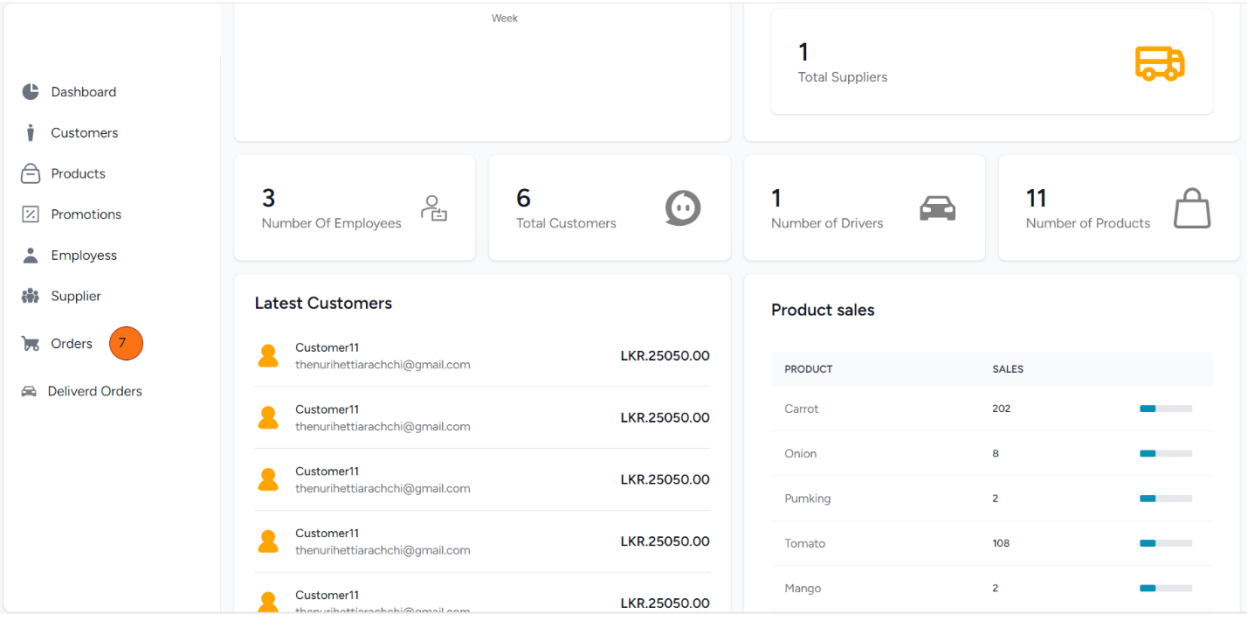


Figure 8 - Admin & Employee Dashboard Part 2

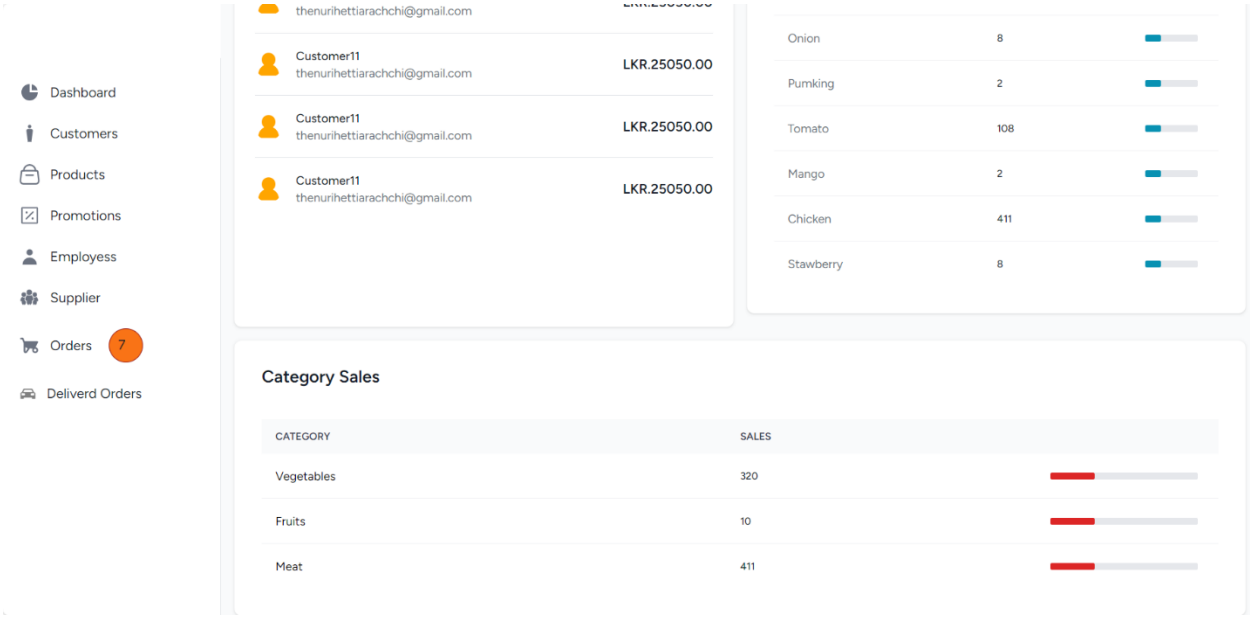


Figure 9 - Admin & Employee part 3

Customer Tab

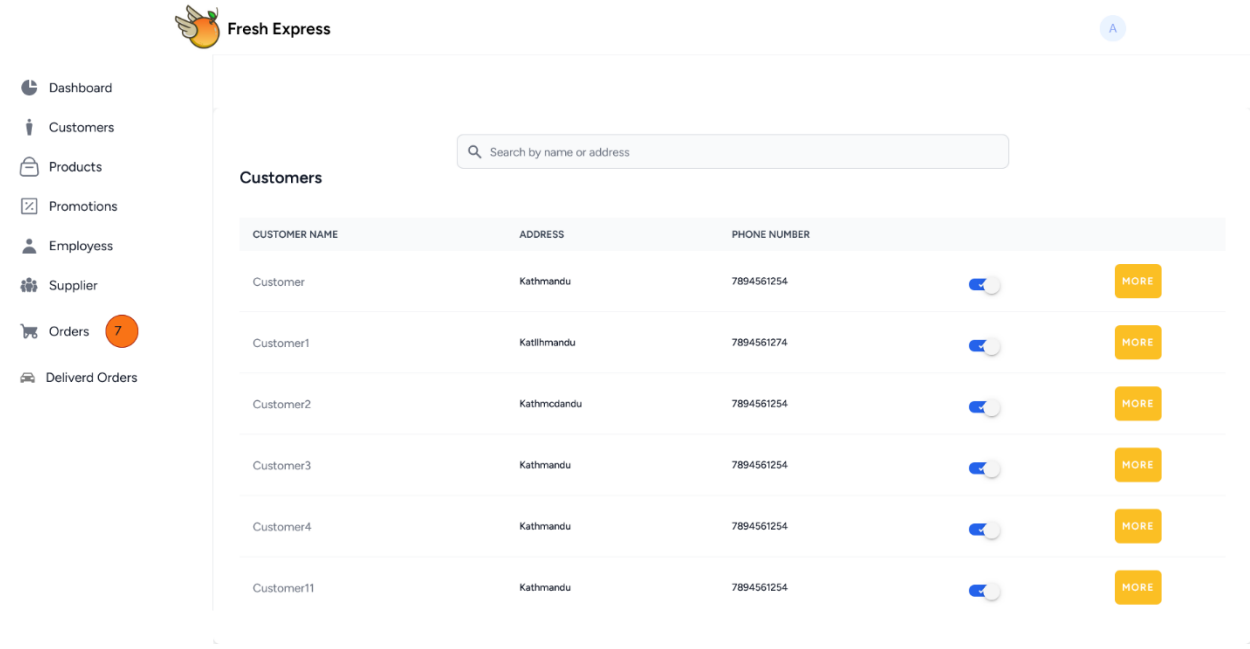


Figure 10 - Customer Tab

Product Page

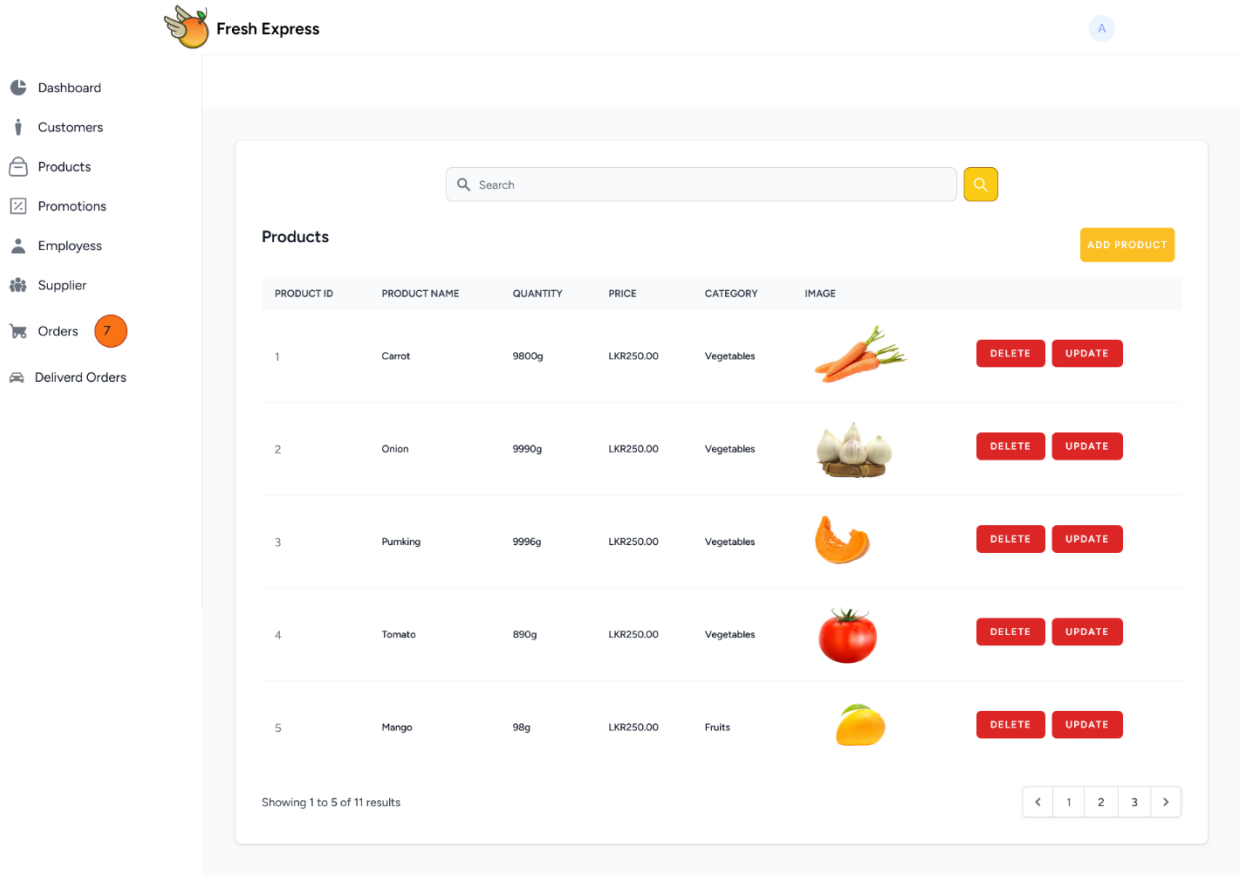


Figure 11 - Product Page

Promotions Page

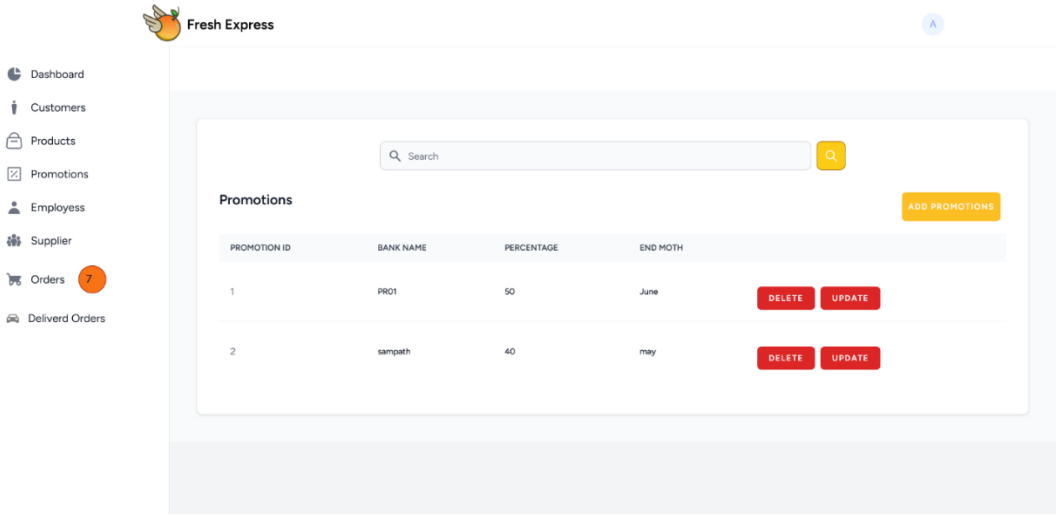


Figure 12 - Promotions Page

Employees Display page.

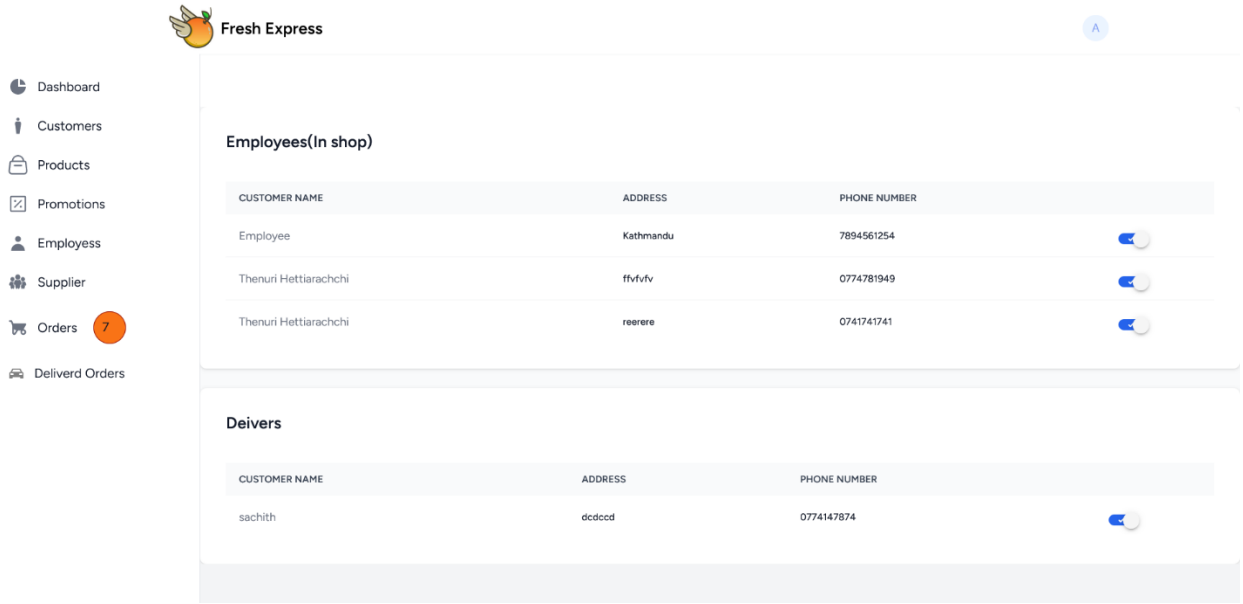


Figure 13 - Employee Display

Supplier Page

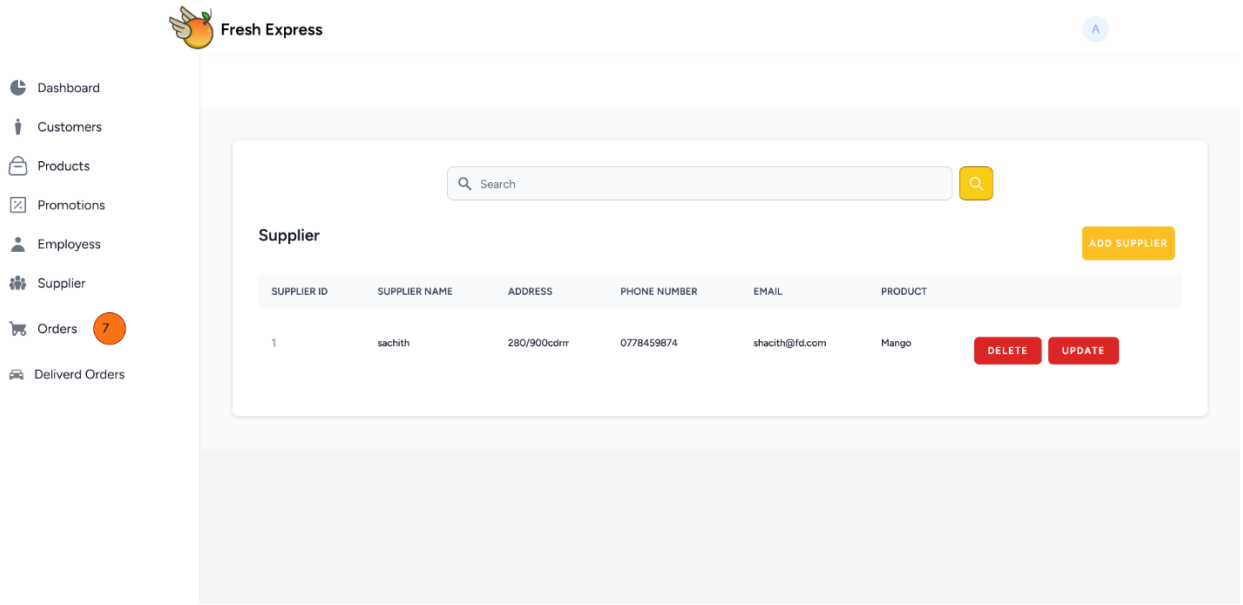


Figure 14 - Supplier Page

Orders Page

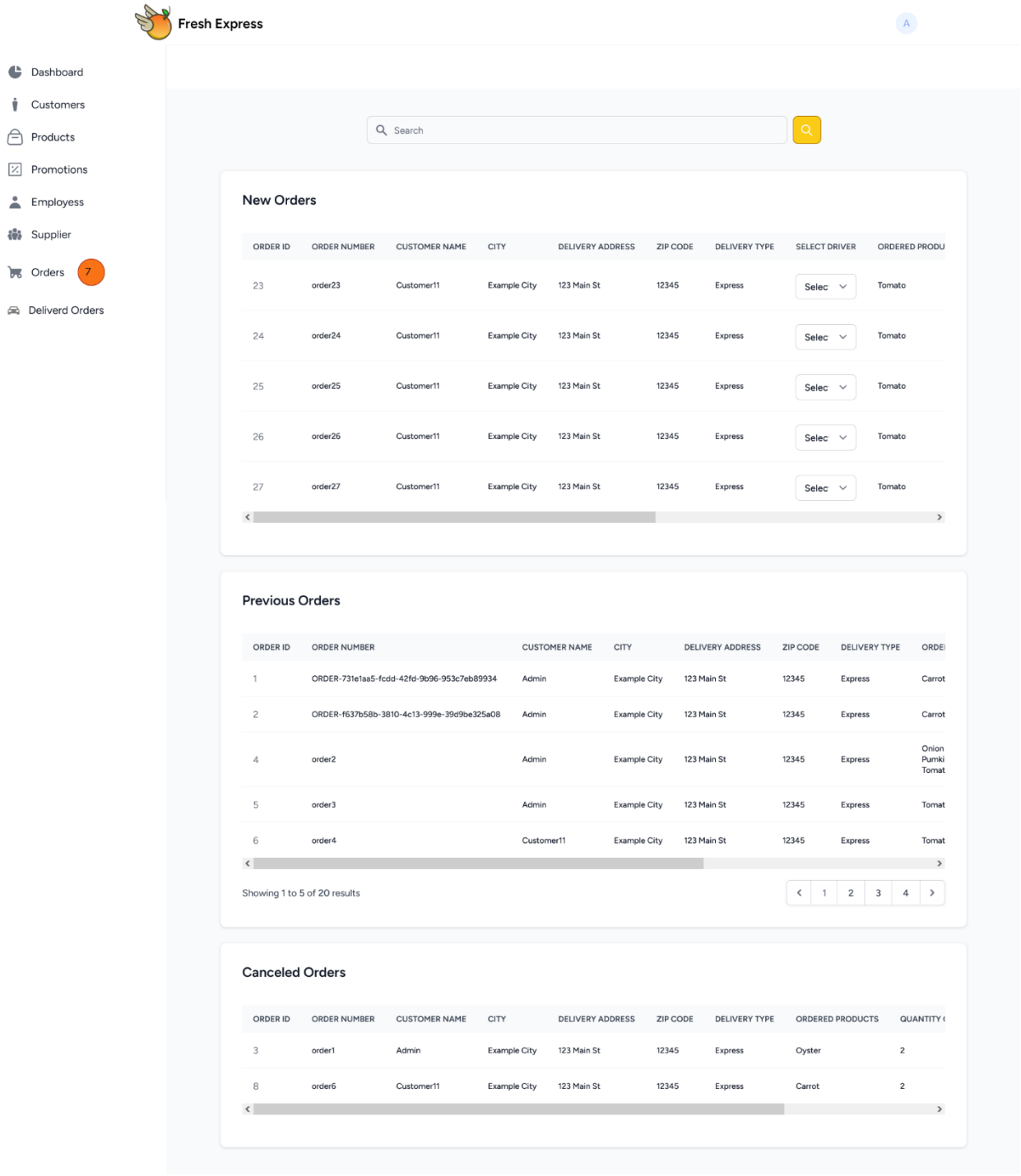


Figure 15 - Order Page

Delivered Orders

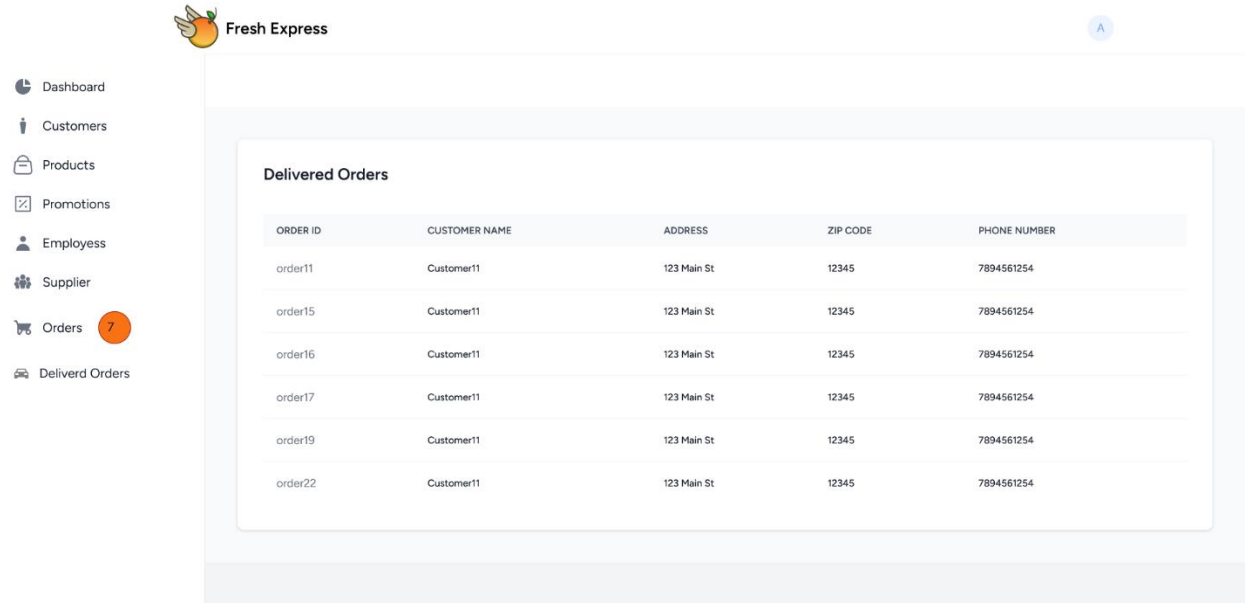


Figure 16 - Delivered Orders

Employee display for the employees

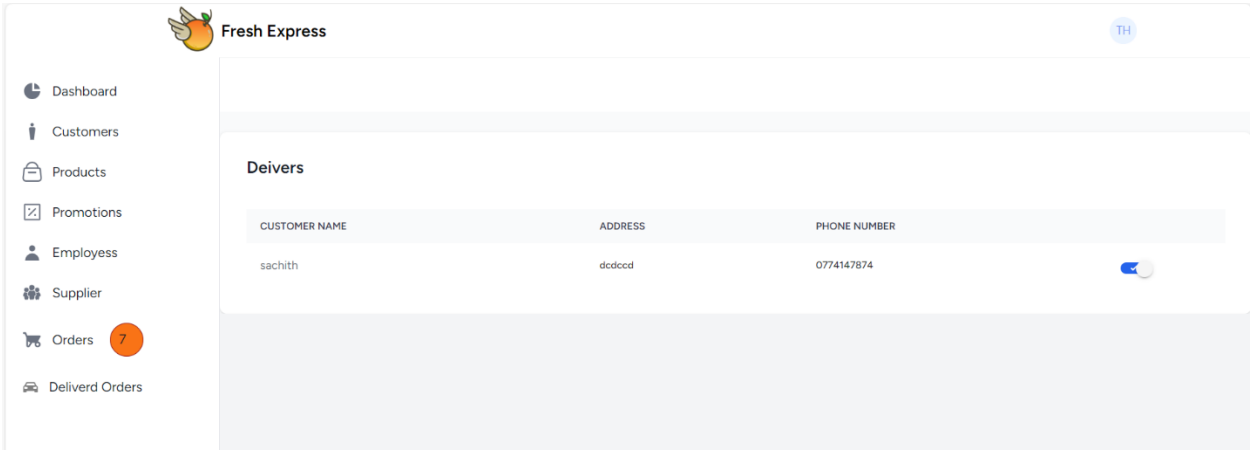


Figure 17 - Employee display for employees

Driver Dashboard

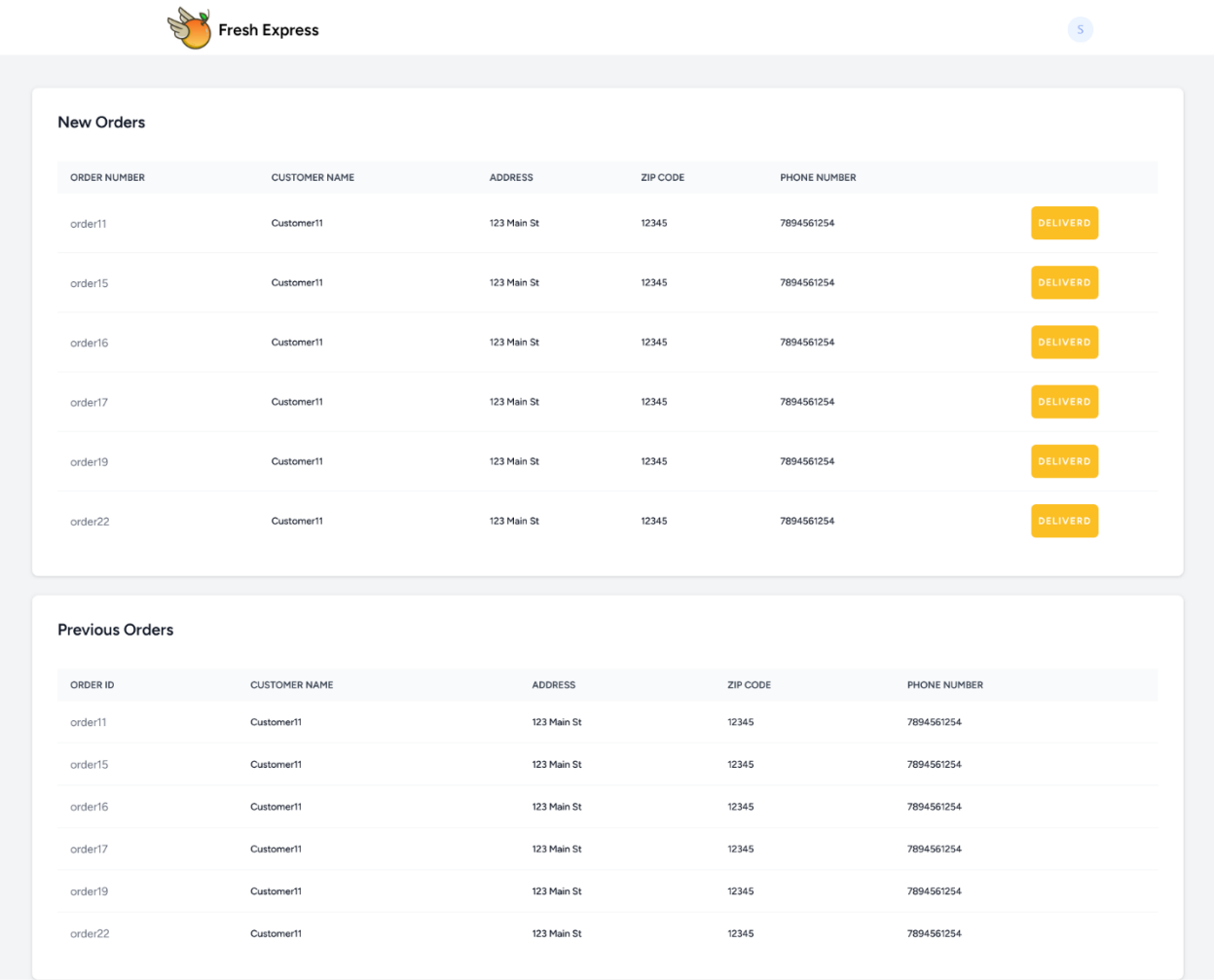


Figure 18 - Driver Dashboard

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

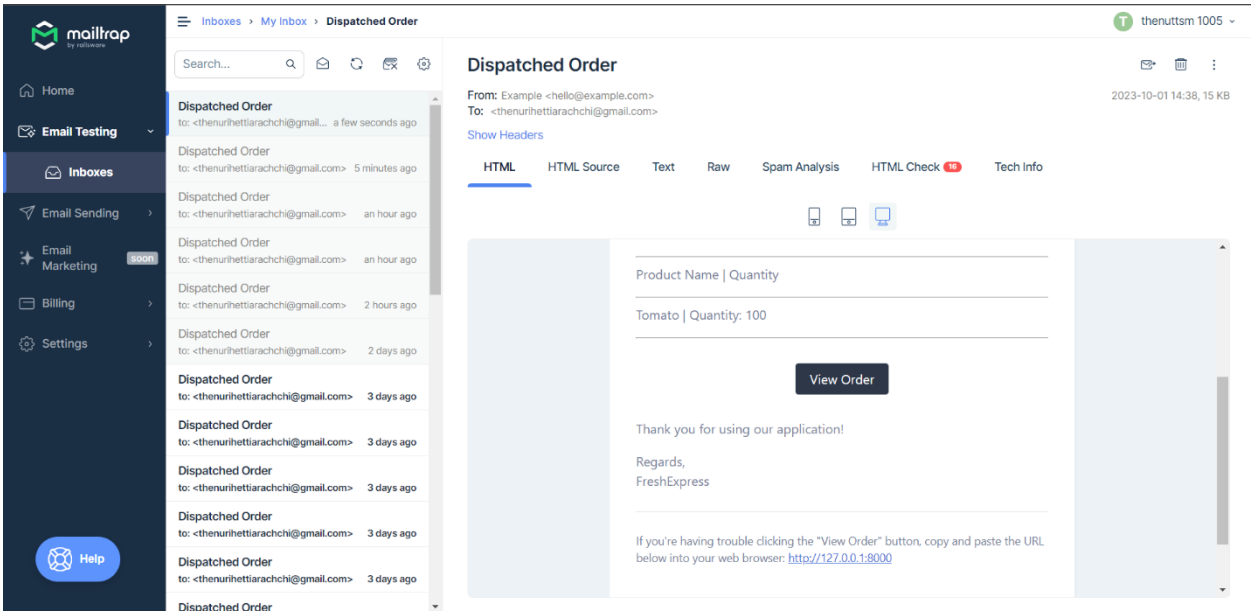


Figure 19 – mailtrap

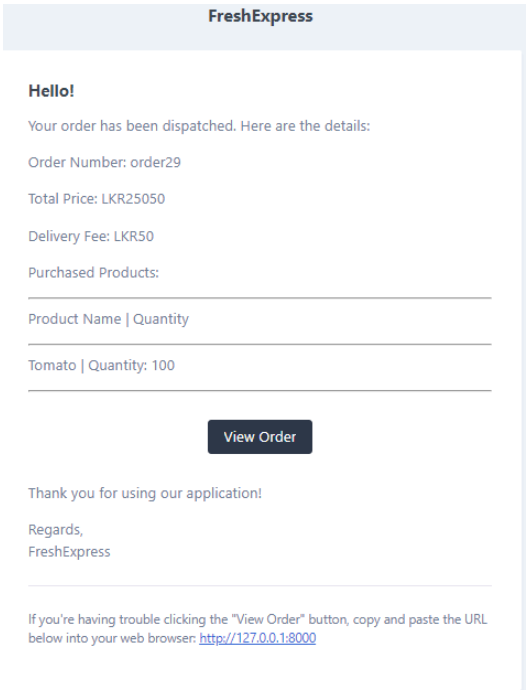


Figure 20 - Mail that the customer can see

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

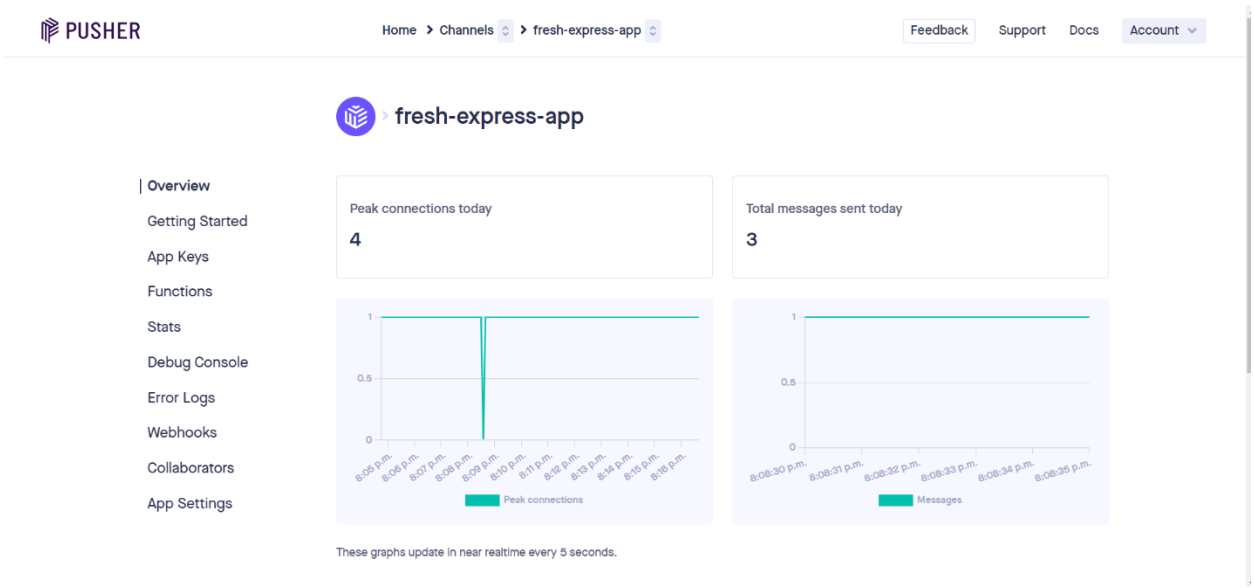


Figure 21 – pusher

Screenshots of the code & postman requests

Migration files

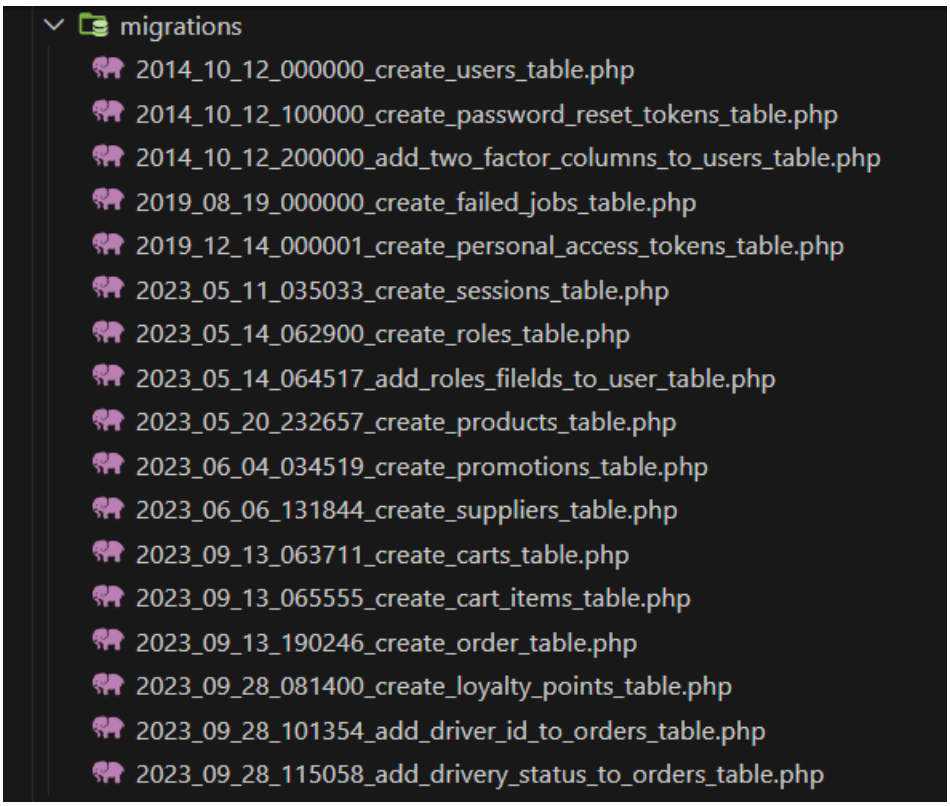


Figure 22 - Migration Files

Livewire and view files

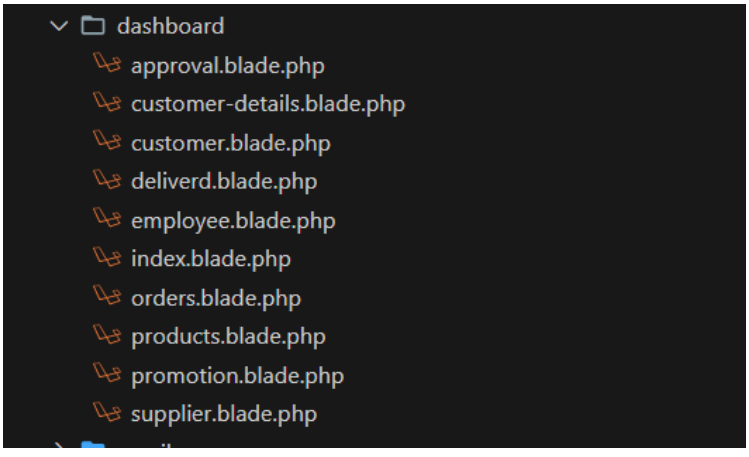


Figure 23 - Blade files

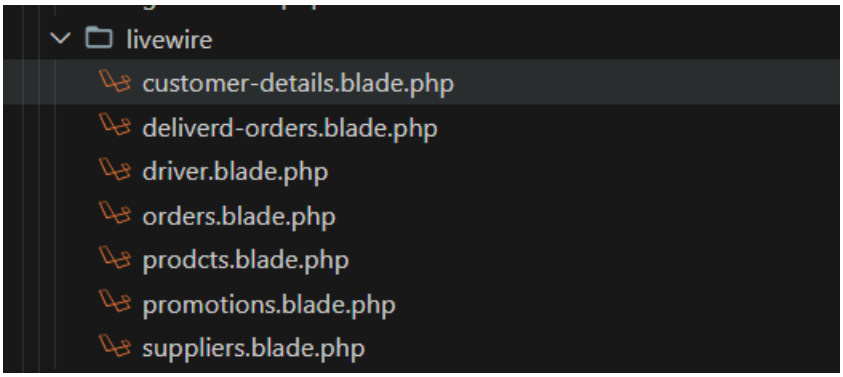


Figure 24 - Livewire files

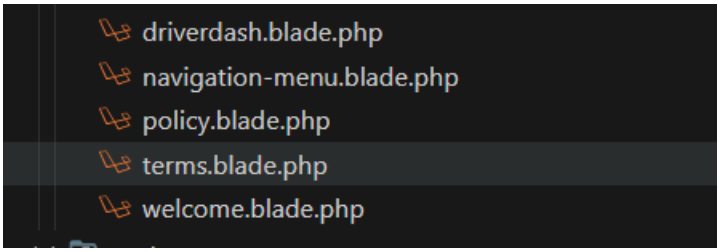


Figure 25 - Blade Files

Controllers

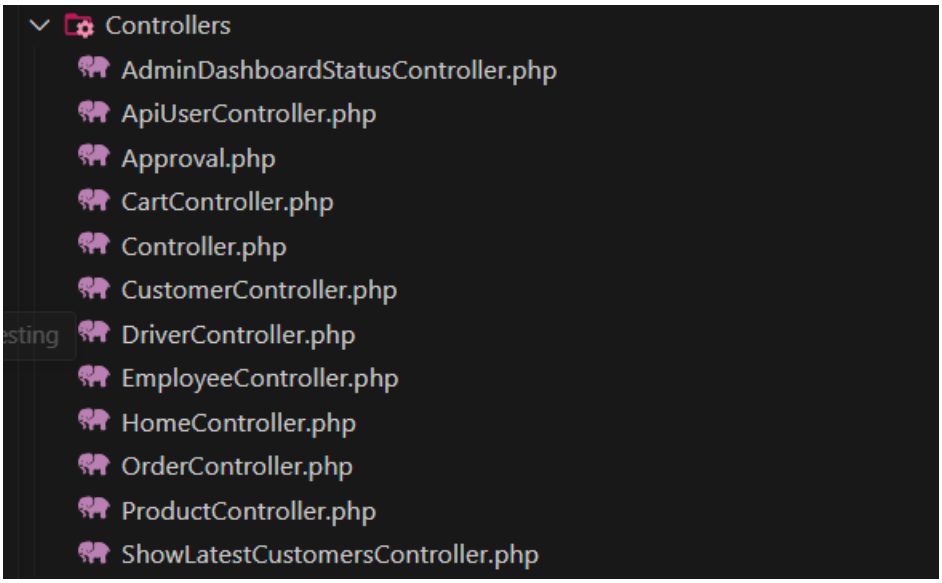


Figure 26 – Controllers

Livewire Controllers

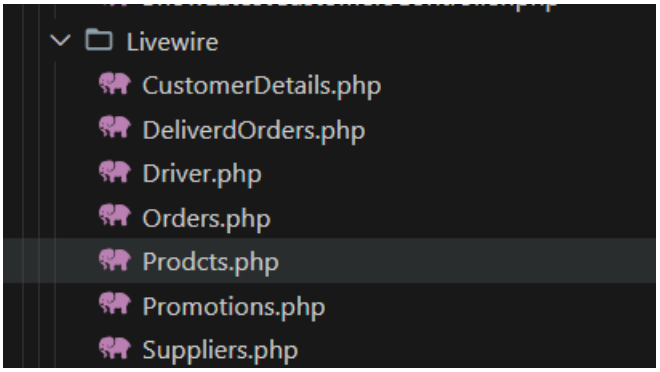


Figure 27 - Livewire Controllers

Models

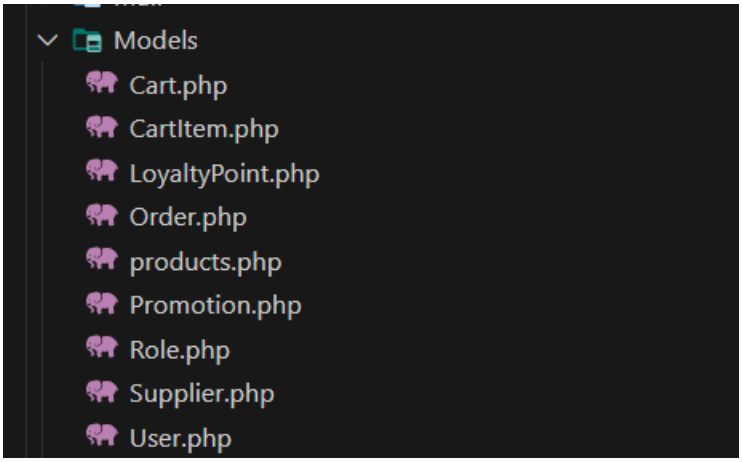


Figure 28 – Models

Cart Controller

```php
<?php

namespace App\Http\Controllers;

use App\Models\products;
use Illuminate\Http\Request;

use App\Models\Cart;

class CartController extends Controller
{
    public function addToCart(Request $request)
    {

        $user = $request->user();

        // $product=$request->product_id;
        // return $product;

        // $quantity=$request->quantity;

        // extract the product id and quantity from the request's json body
        // $product_id = $request->product_id;
        // $quantity = $request->quantity;

        // return $product_id;

        //check if the user has a cart
        $cart = Cart::where('user_id', $user->id)->where('is_paid', false)->first();


        //if not create a new cart
        if (!$cart) {
            $cart = Cart::create([
                'user_id' => $user->id,
                'is_paid' => false,
                'total' => 0,
            ]);
        }

        //check products alredy in cart
        $cartItem = $cart->items()->where('product_id', $request->product_id)->first();
        $product = products::find($request->product_id);
        $price = $product->price;

        //if product is in cart, increase quantity
        if ($cartItem) {


            $newTotal = $price *($cartItem->quantity + $request->quantity);
            $cartItem->update([
                'quantity' => $cartItem->quantity + $request->quantity,
                'total' => $newTotal,
            ]);

        } else {

            // get the product price
            // $product = products::find($request->product_id);
            // $price = $product->price;

            //if the product is not in te cart add it
            $cartItem = $cart->items()->create([
                'product_id' => $request->product_id,
                'quantity' => $request->quantity,
                'total' => $price * $request->quantity,
            ]);
        }
        $cart->refresh();
        //calculate the cart total

         $cartTotal = $cart->items()->sum('total');


        //update the cart total

        $cart->update([
            'total' => $cartTotal,
        ]);

        return response()->json([
            'message' => 'Product added to cart',
            'cart' => $cart,
        ]);



    }
}
```

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.

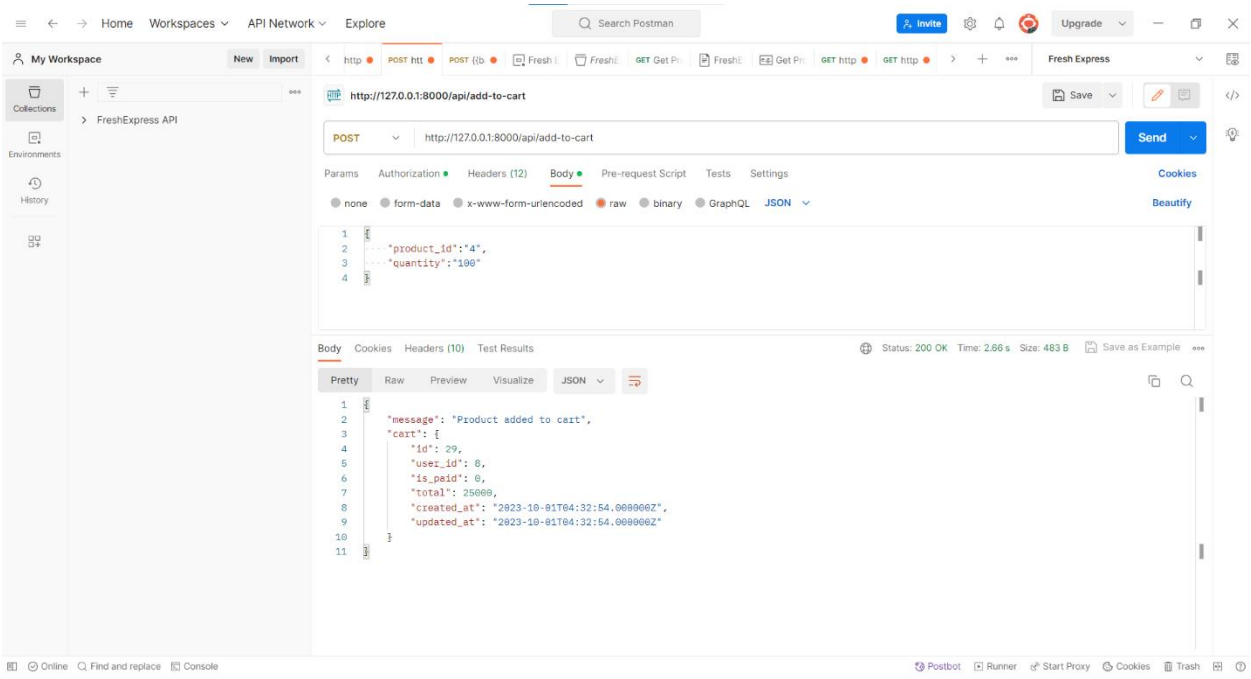
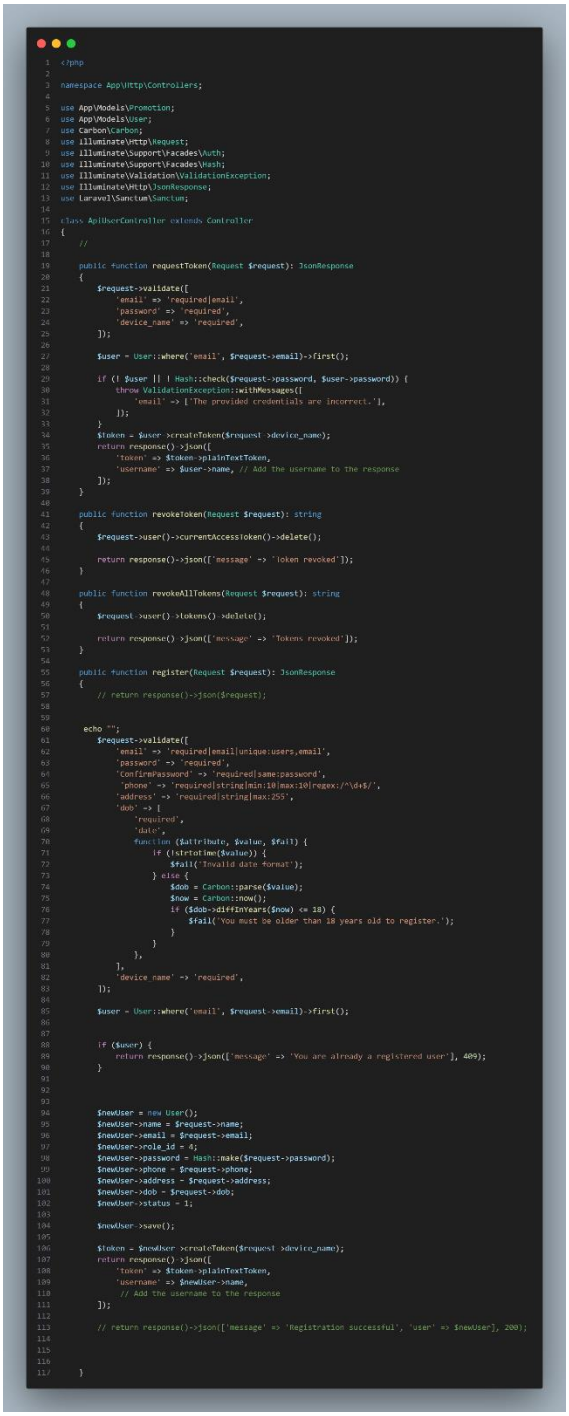


Figure 30 - API Add to cart

ApiUserController



```php
<?php

namespace App\Http\Controllers;

use App\Models\Promotion;
use App\Models\User;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\ValidationException;
use Illuminate\Http\JsonResponse;
use Laravel\Sanctum\Sanctum;

class ApiUserController extends Controller
{
    //

    public function requestToken(Request $request): JsonResponse
    {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required',
            'device_name' => 'required',
        ]);

        $user = User::where('email', $request->email)->first();

        if (! $user || ! Hash::check($request->password, $user->password)) {
            throw ValidationException::withMessages([
                'email' => ['The provided credentials are incorrect.'],
            ]);
        }
        $token = $user ->createToken($request ->device_name);
        return response() ->json([
            'token' => $token->plainTextToken,
            'username' => $user->name, // Add the username to the response
        ]);
    }

    public function revokeToken(Request $request): string
    {
        $request->user()->currentAccessToken()->delete();

        return response()->json(['message' => 'Token revoked']);
    }

    public function revokeAllTokens(Request $request): string
    {
        $request ->user() ->tokens() ->delete();

        return response() ->json(['message' -> 'Tokens revoked']);
    }

    public function register(Request $request): JsonResponse
    {
        // return response()->json($request);


     echo "";
        $request->validate([
            'email' => 'required|email|unique:users,email',
            'password' -> 'required',
            'ConfirmPassword' -> 'required|same:password',
            'phone' -> 'required|string|min:10|max:10|regex:/^\d+$/',
            'address' -> 'required|string|max:255',
            'dob' -> [
                'required',
                'date',
                function ($attribute, $value, $fail) {
                    if (!strtotime($value)) {
                        $fail('Invalid date format');
                    } else {
                        $dob = Carbon::parse($value);
                        $now = Carbon::now();
                        if ($dob->diffInYears($now) <= 18) {
                            $fail('You must be older than 18 years old to register.');
                        }
                    }
                },
            ],
            'device name' -> 'required',
        ]);

        $user = User::where('email', $request->email)->first();


        if ($user) {
            return response()->json(['message' => 'You are already a registered user'], 409);
        }



        $newUser = new User();
        $newUser->name = $request->name;
        $newUser->email = $request->email;
        $newUser->role_id = 4;
        $newUser->password = Hash::make($request->password);
        $newUser->phone = $request->phone;
        $newUser->address = $request->address;
        $newUser->dob = $request->dob;
        $newUser->status = 1;

        $newUser->save();

        $token = $newUser ->createToken($request ->device_name);
        return response()->json([
            'token' => $token->plainTextToken,
            'username' => $newUser->name,
            // Add the username to the response
        ]);

        // return response()->json(['message' => 'Registration successful', 'user' => $newUser], 200);



    }
```

Figure 31 - API API Controller

```php
    public function getCustomerDetails(Request $request): JsonResponse
    {
        // Retrieve the customer details based on the API token
        $user = Sanctum::actingAs($request->user(), ['*']);

        if ($user) {
            // Return the customer details as JSON response
            return response()->json([
                'name' => $user->name,
                'contactNumber' => $user->phone,
                'address' => $user->address,
                'dateOfBirth' => $user->dob,
                'emailAddress' => $user->email,
            ]);
        } else {
            // Handle the case when customer details are not found
            return response()->json(['error' => 'Customer details not found'], 404);
        }
    }

    public function getPromotions(Request $request): JsonResponse
    {
        // Retrieve promotion details from the database
        $promotions = Promotion::all(); // This fetches all promotions; you can customize the query as needed

        // Check if any promotions were found
        if ($promotions->isEmpty()) {
            return response()->json(['message' => 'No promotions found'], 404);
        }

        // Return the promotion details as a JSON response
        return response()->json(['promotions' => $promotions]);
    }

    public function getUserLoyaltyPoints(Request $request): JsonResponse{
        // Retrieve the authenticated user based on the API token
        $authenticatedUser = Auth::user();

        // Check if the user is authenticated
        if (!$authenticatedUser) {
            return response()->json(['error' => 'User not authenticated'], 401);
        }

        // Load the authenticated user's loyalty points if it's a relationship
        $authenticatedUser->load('loyaltyPoints');

        // Calculate the total loyalty points for the authenticated user
        $totalLoyaltyPoints = $authenticatedUser->loyaltyPoints->sum('points');

        // Create a response with the authenticated user's loyalty points
        $responseData = [
            'user_id' => $authenticatedUser->id,
            'name' => $authenticatedUser->name,
            'total_loyalty_points' => $totalLoyaltyPoints,
        ];

        return response()->json($responseData);
    }

}

```

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.

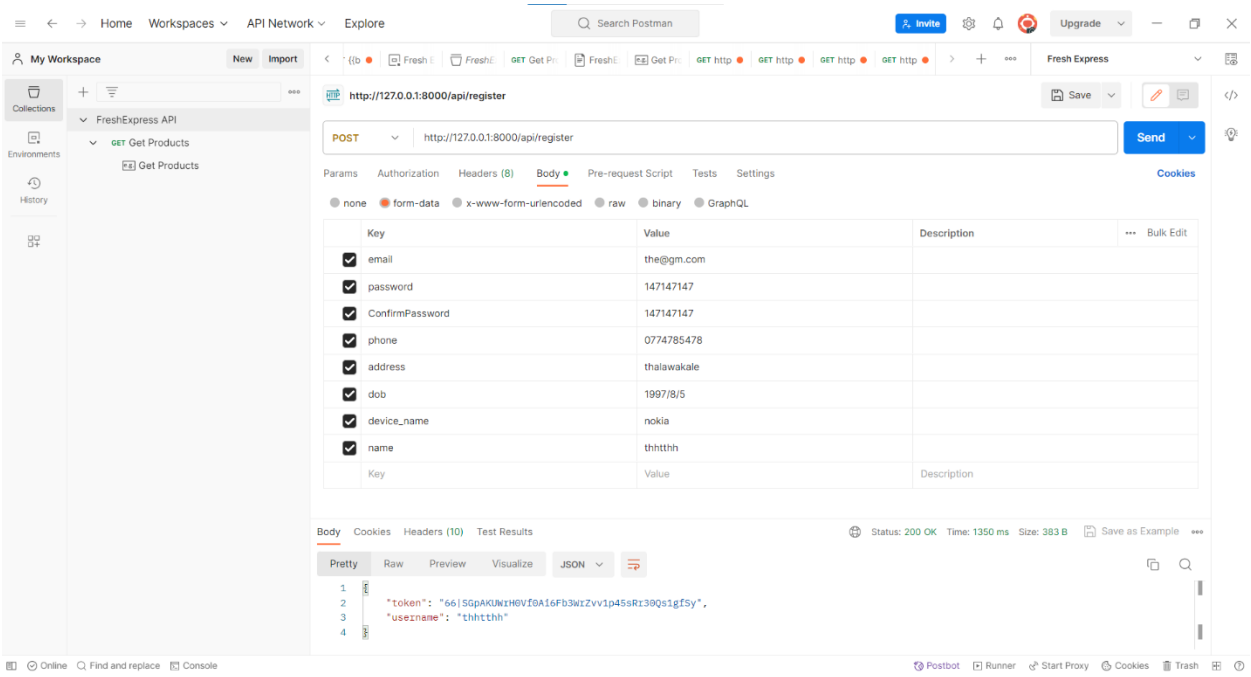


Figure 33 - API Register

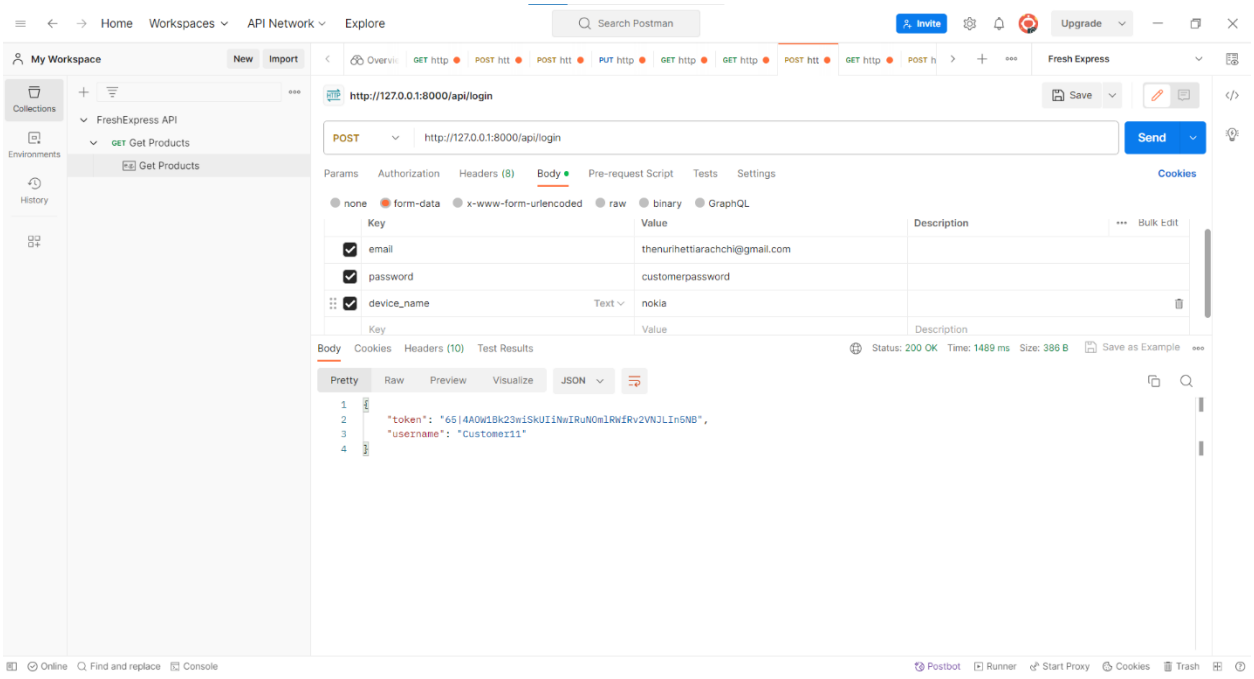


Figure 34 - API Login

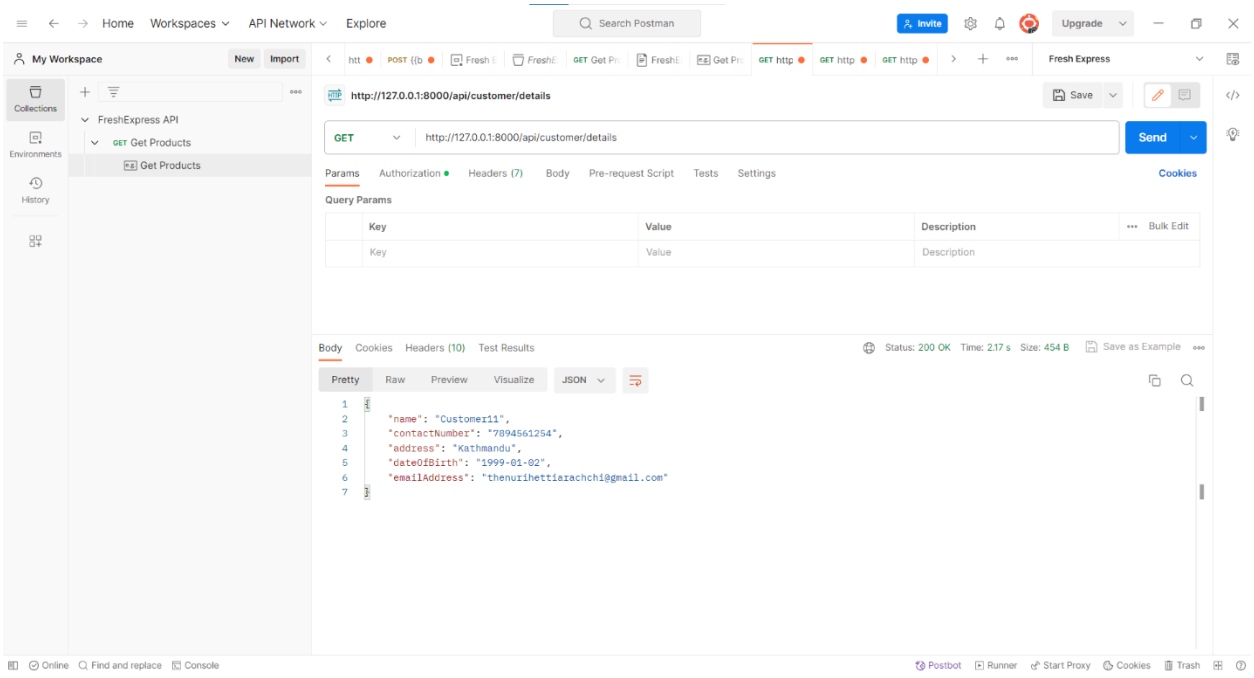


Figure 35 -API Getting Customer Details

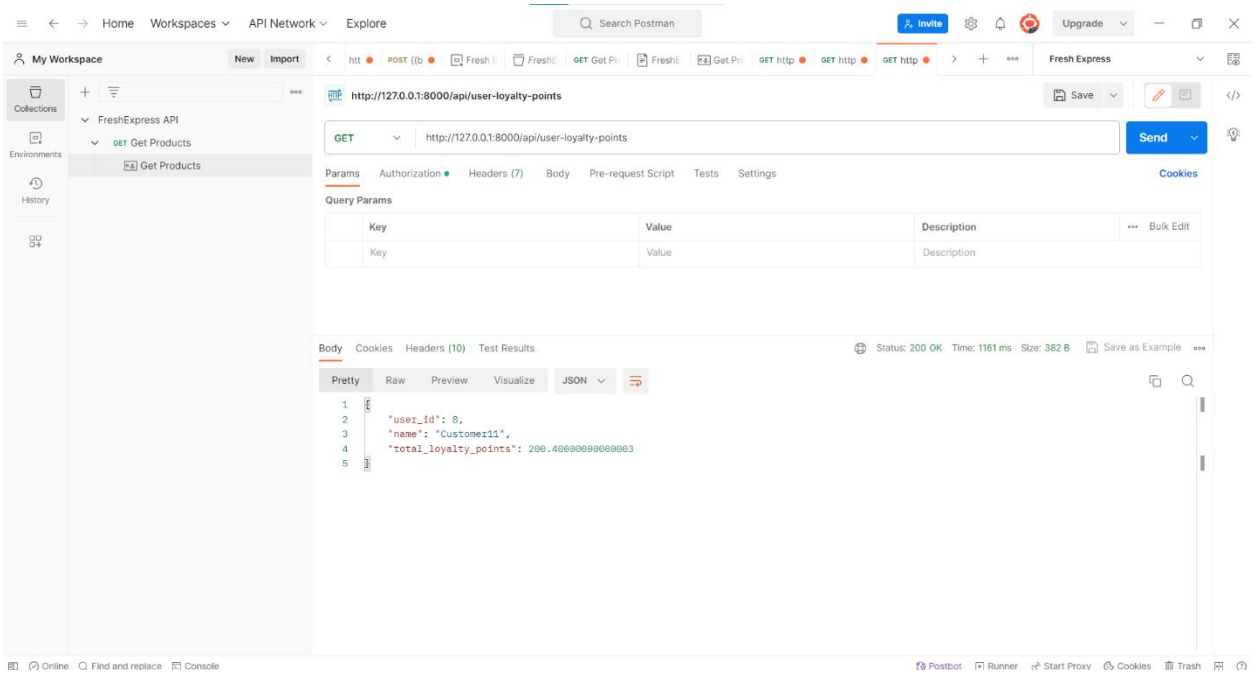


Figure 36 - API Loyalty Points

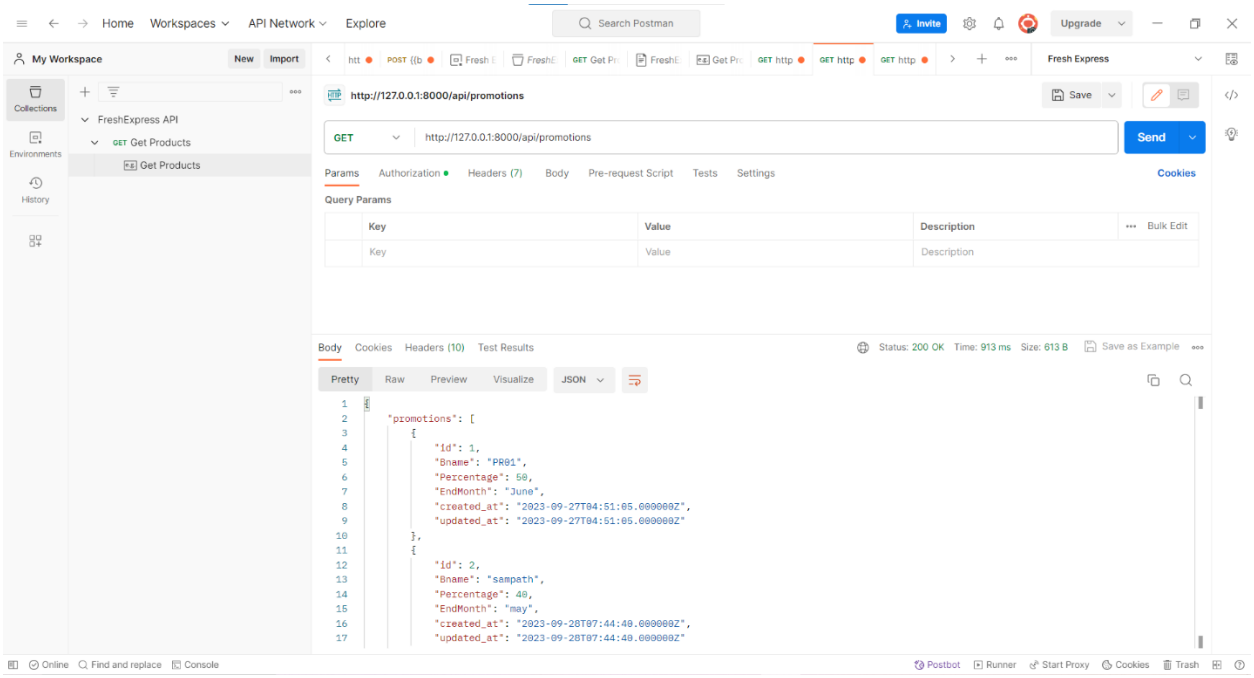


Figure 37 - API Promotions

Order Controller

```php
<?php

namespace App\Http\Controllers;

use App\Models\Cart;
use App\Models\Order;
use Illuminate\Http\Request;
use Illuminate\Support\Str;
use App\Events\NewOrderReceivedEvent;

class OrderController extends Controller
{
    public function completeCart(Request $request, Cart $cart)
    {

        // dd($cart);


        //check if the cart exists and is not already paid
        if (!$cart || $cart->is_paid) {
            return response()->json(['message' => 'Cart not found or already completed'], 404);
        }

        // $address = $request->delivery_address;
        // $ZIP_code = $request->ZIP_code;
        // $city = $request->city;
        // $delivery_type = $request->delivery_type;
        // return $address;
        //Make the cart as completed (is_paid = true)

        // Find the highest order number in the database and increment it by 1
        $latestOrderNumber = Order::max('id');
        if ($latestOrderNumber) {
            $orderNumber = 'order' . $latestOrderNumber + 1;
        } else {
            $orderNumber = 'order1';
        }
        // return ($latestOrderNumber);


        $deliveryFee = 50;
        $totalAmount = $cart->total + $deliveryFee;
        //Create an order
        $order = Order::create([
            'user_id' => $cart->user_id,
            'cart_id' => $cart->id,
            'total' => $totalAmount,
            'status' => 'pending',
            'delivery_status' => 'pending',
            'delivery_address' => $request->delivery_address,
            'ZIP_code' => $request->ZIP_code,
            'city' => $request->city,
            'delivery_type' => $request->delivery_type,
            'order_number' => $orderNumber,
        ]);
        //calculate loyalty points based on the total amount
        $loyaltyPointsEarned = $totalAmount*0.001;
        //Add the loyalty points to the user
        $user = $cart->user;
        $user->loyaltyPoints()->create([
            'points' => $loyaltyPointsEarned,
            'order_id' => $order->id,
        ]);
        $cart->update([
            'is_paid' => true,
        ]);

        $response = [
            'message' => 'cart marked as completed',
            'order_number' => $orderNumber,
            'cart_total' => $cart->total,
            'delivery_fee' => $deliveryFee,
            'total_amount' => $totalAmount,
            'order_created_date' => $order->created_at->toDateString(),
            'loyalty_points_earned' => $loyaltyPointsEarned,
        ];


        event((new NewOrderReceivedEvent()));

        return response()->json($response, 200);


    }
}
```

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

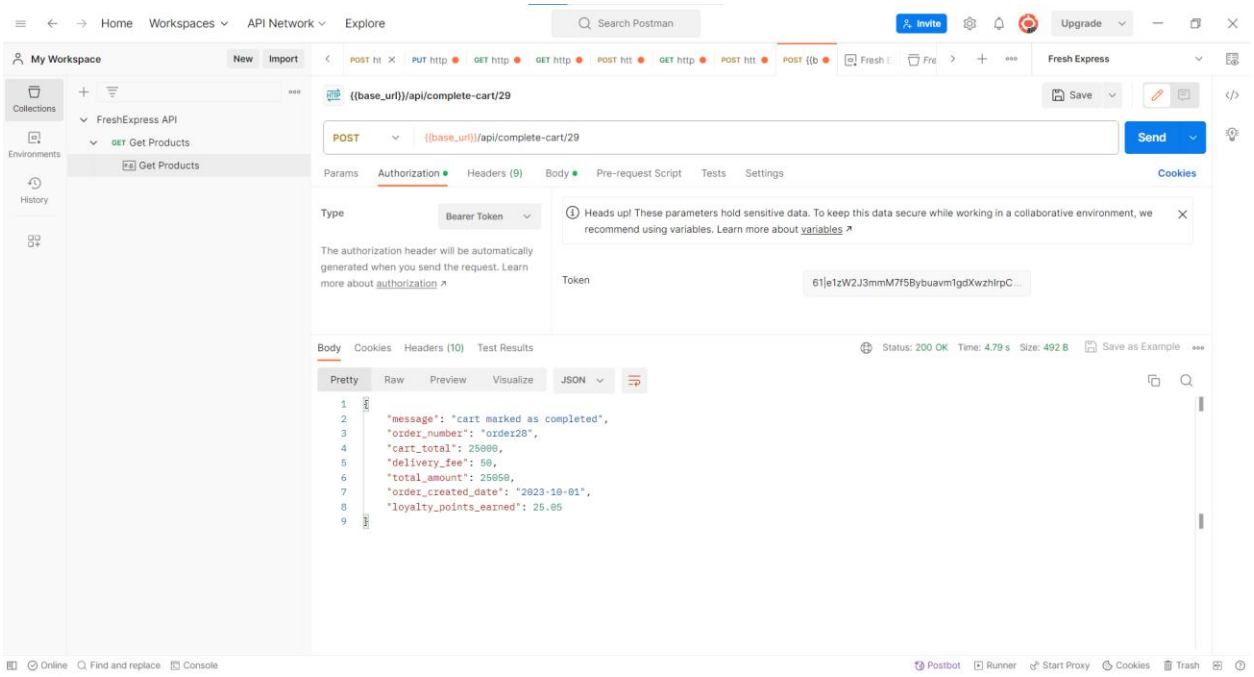


Figure 39 – Order

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

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

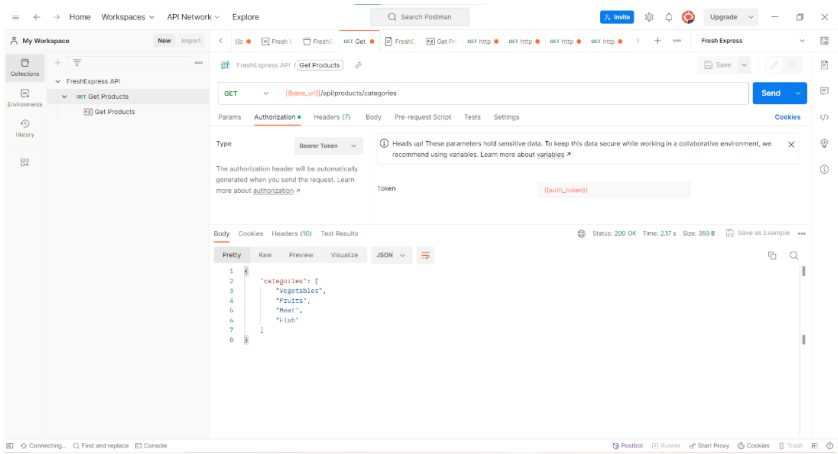


Figure 40 - Getting Product categories.

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

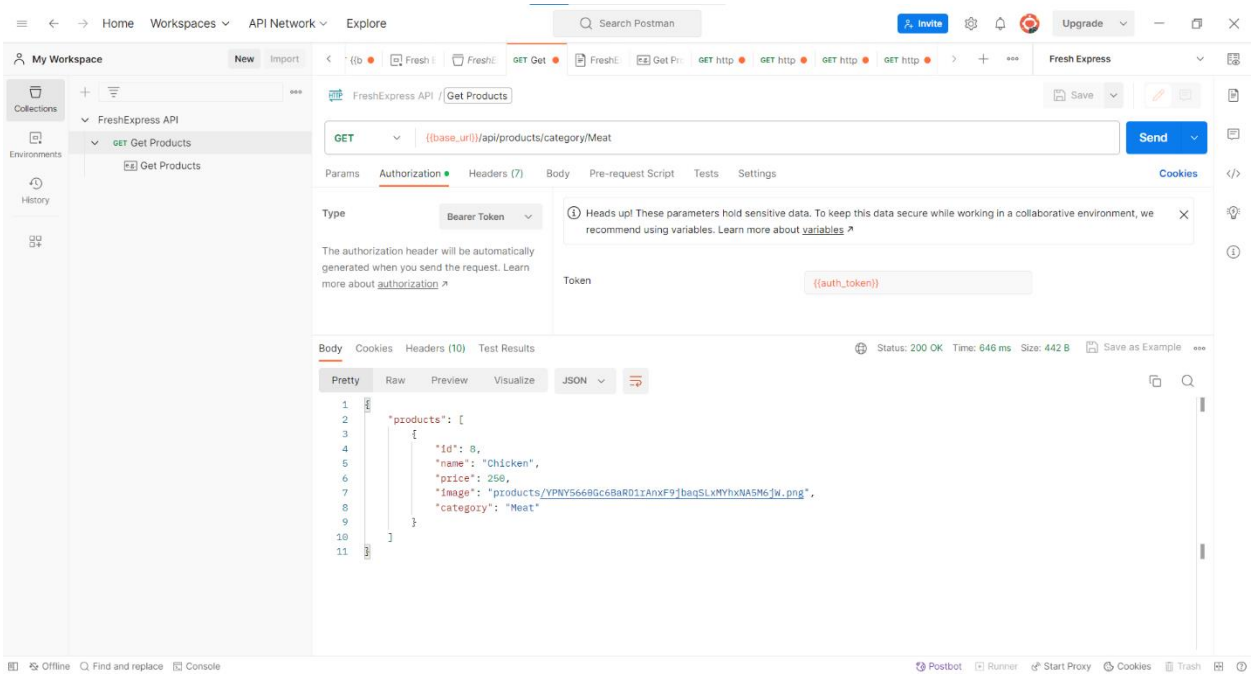


Figure 41 - Getting Products by category name.

Route Files (API.PHP & WEB.PHP)

```php
<?php

use App\Events\PusherSample;
use App\Http\Controllers\Approval;
use App\Http\Controllers\AdminDashboardStatusController;
use App\Http\Controllers\customerApproval;
use App\Http\Controllers\CustomerController;
use App\Http\Controllers\EmployeeController;
use App\Http\Controllers\ShowLatestCustomersController;
use Illuminate\Routing\Controllers\Middleware;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

// Route::get('dashboard', function () {
//     return view('welcome');
// });
// Route::get('/admin/approval', [AdminController::class, 'approval'])->name('admin.approval');

Route::get('/dev', function () {
    event((new NewOrderReceivedEvent(100)));
});


Route::get('/', [HomeController::class, 'checkRoleid']);

Route::post('/admin/change-status/{user}', [Approval::class, 'switchStatus'])
    ->name('admin.approve')->middleware('adminOrEmployee');

// Route::post('/admin/change-status/{user}', [customerApproval::class, 'switch'])
//     ->name('admin.customerapprove')->middleware('adminOrEmployee');

Route::middleware(['auth'])->group(function () {
    Route::resource('dashboard/customer', CustomerController::class)->name('index', 'customer');
    Route::resource('dashboard/employee', EmployeeController::class)->name('index', 'employee');
    // Route::resource('dashboard/customerApproval', customerApproval::class)->name('index', 'customerApproval');
    Route::resource('dashboard/approval', Approval::class)->name('index', 'approval');
    Route::view('dashboard/orders', 'dashboard.orders')->name('orders');
    //Route::view('dashboard/products', 'dashboard.products')->name('products');
});
// Route::get('customers/search', 'CustomerController@search')->name('customers.search');
// Route::resource('customers', 'CustomerController');

// Route::get('/admin', function () {
//     return view('dashboard.index');
// })->name('admin.dashboard')
// ->middleware(['adminOrEmployee','auth']);

Route::get('/admin', [AdminDashboardStatusController::class, 'index'])
    ->name('admin.dashboard')
    ->middleware(['adminOrEmployee', 'auth']);

// Route::get('/latest-customers', [ShowLatestCustomersController::class, 'showLatestCustomers'])
// ->name('latest-customers')
// ->middleware(['adminOrEmployee', 'auth']);

Route::get('/employee', [AdminDashboardStatusController::class, 'index'])
    ->name('employee.dashboard')
    ->middleware(['adminOrEmployee', 'userApproved',]);

// Route::get('/employee', function () {
//     return view('dashboard.index');
// })->name('employee.dashboard')
// ->middleware(['adminOrEmployee','userApproved']);

Route::get('/driver', function () {
    return view('driverdash');
})->name('driver.dashboard')
    ->middleware(['userApproved']);

Route::get('/customer-', function () {
    return view('customerdash');
})->name('customer.dashboard');


Route::middleware([
    'auth:sanctum',
    config('jetstream.auth_session'),
    'verified'
])->group(function () {
    Route::get('/', [HomeController::class, 'checkRoleid']);
});

Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
    Route::get('/dashboard/products', function () {
        return view('dashboard.products');
    })->name('products');
});

Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
    Route::get('/dashboard/promotion', function () {
        return view('dashboard.promotion');
    })->name('promotion');
});

Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified'])->group(function () {
    Route::get('/dashboard/supplier', function () {
        return view('dashboard.supplier');
    })->name('supplier');
});

//view customer details
Route::get('/customerdetails/{user_id}', function ($userId) {
    return view('dashboard.customer-details', compact('userId'));
})->name('customerdetails');

Route::get('/deliverdorders', function () {
    return view('dashboard.deliverd', );
})->name('deliverdorders');
```

Figure 42 – WEB.php

```php
<?php

use App\Http\Controllers\OrderController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ApiUserController;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\ApiCartController;
use App\Http\Controllers\CartController;



/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::middleware('auth:sanctum')->post('/user', function (Request $request) {
    return $request->user();
});


Route::post('login', [ApiUserController::class, 'requestToken']);

Route::post('register', [ApiUserController::class, 'register']);

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


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

//Cart
// Route::post('/add-to-cart', 'CartController@addToCart')->middleware('api.token');


// Route::group(['middleware' => 'api.token'], function () {
//     Route::post('/add-to-cart', 'CartController@addToCart');

// });

// Route::middleware('auth:sanctum')->post('/add-to-cart', function (Request $request) {
//     return $request->user();

// });

 //adding  the items to cart
Route::middleware('auth:sanctum')->post('/add-to-cart', [CartController::class, 'addToCart']);

//creatimg an order and gettin the order details
Route::middleware('auth:sanctum')->post('/complete-cart/{cart}', [OrderController::class, 'completeCart']);



// Route::post('/complete-cart/{cart}', 'OrderController@completeCart');




// http://127.0.0.1:8000/api/products/categories/Fruit/




// send hello
Route::get('hello', function () {
    return 'Hello World';
});
//getting customer details to display in the profile page
Route::middleware(['auth:sanctum'])->group(function () {
    Route::get('/customer/details', [ApiUserController::class, 'getCustomerDetails']);
    Route::get('user-loyalty-points', [ApiUserController::class, 'getUserLoyaltyPoints']);

});

//getting all the promotions
Route::get('/promotions', [ApiUserController::class, 'getPromotions']);


```

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

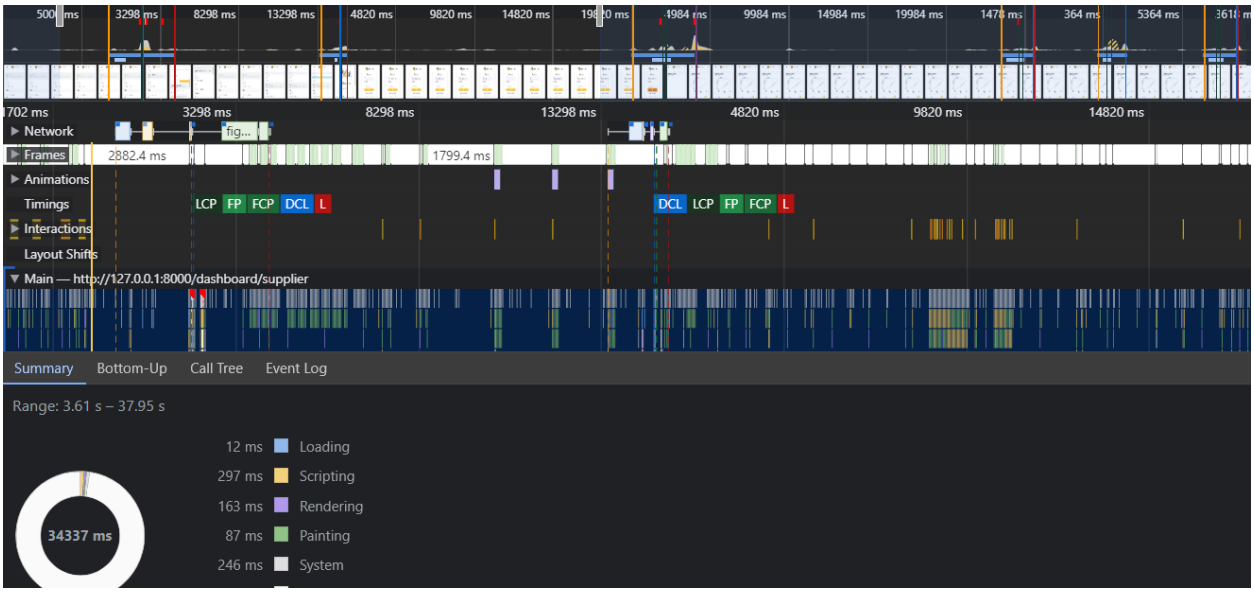


Figure 44 - Performance testing

LightHouse Testing

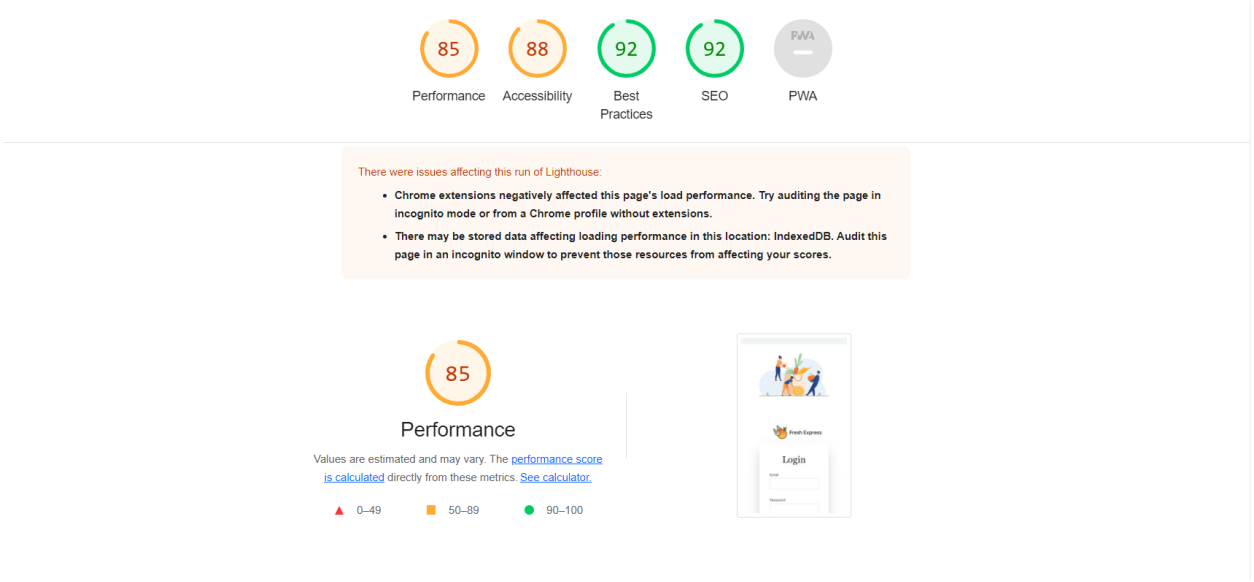


Figure 45 - Lighthouse testing

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.