اونيۏرسيتي مليسيا ڤهڠ السلطان عبد الله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

SOFTWARE MAINTENANCE & EVOLUTION

BCS3153

SEMESTER II SESSION 2024/2025

PROJECT

LAB SECTION: 01A

STUDENT DETAILS:		
NAME	**STUDENT ID**	**STUDENT PHOTO**
MUHAMMAD HARIZ HAIKAL BIN NORISMAN	CB23130	

Table of contents

1.0 Introduction

1.1 Project Overview

The Plasticware SME System (PSS) is a web-based application designed to streamline and digitalize operations for small and medium-sized enterprises (SMEs) in the plasticware industry. Developed for a business named Perniagaan Heap Loong Hin, this system integrates the full chain from product sales to plastic recycling. The system offers comprehensive features including product and order management, payment processing, recycling activities, reward systems, and insightful dashboards. Its goal is to improve business efficiency, promote sustainable practices, and reduce the environmental impact of plastic waste. Built using Laravel and Tailwind CSS and structured with a Model-View-Controller (MVC) architecture, the system ensures modularity, scalability, and maintainability. It also underwent User Acceptance Testing (UAT) to validate usability and performance, ensuring the platform meets user expectations and business needs.

Module 1: Manage Order

In the Manage Order module of the Plasticware SME System (PSS), two key functionalities serve both the administrator and user roles: New Order (Pending) for the admin side and History Order for the user side.

On the admin side, the New Order (Pending) feature allows administrators to view and manage all newly placed orders that are still awaiting action. These orders typically have an "order_status" of "pending," indicating that payment has not yet been verified or the order is not yet processed. Through this interface, admins can view order details, verify uploaded payment proofs (such as receipts), and then update the order status to either "completed" or "canceled" based on verification results. This process enhances the transparency and accountability of order handling, reducing the risk of errors, fraud, or missed transactions. The system also adopts a responsive design, allowing administrators to efficiently manage these tasks from desktops or mobile devices.

On the user side, the History Order feature provides customers with a personal record of all their completed orders. Users can view a list of past transactions that have been successfully fulfilled, with details such as order ID, date, total amount, and item breakdown. Unlike the admin interface, this view is strictly read-only to maintain the integrity of completed records.

This feature empowers users to track their purchase history for reference, follow-up, or participation in reward or recycling programs. The mobile-friendly interface ensures a smooth experience, even when accessed on smaller screens.

Together, these two functionalities within the Manage Order module improve operational control for administrators and enhance transparency and convenience for customers.

Module 2: Manage Product

The Manage Product section in the Plasticware SME System functions as a unified platform for the plasticware shop owner to effectively manage product-related tasks such as creating, viewing, updating, and deleting products. This module guarantees efficient inventory management and offers intuitive features to classify items and handle quantities instantly. Recently, a search feature with filters was introduced to help users swiftly find products by name, category, or stock status—enhancing navigation and user effectiveness. Furthermore, activity logs were incorporated to monitor all product-related activities including additions, modifications, and deletions, enhancing system transparency and accountability. These enhancements facilitate more precise inventory management, quicker user access, and improved decision-making for managing products.

Module 3: Manage Payment

The Manage Payment module in this system allows users to make payments for products using either Cash or Online Payment methods. For Online Payments, the system is designed to securely store the user's bank details in the database. After completing their payment, users can view detailed information about their transactions. To improve the system’s maintainability and reliability, there are plans to implement a proper transaction procedure to ensure data consistency during the payment process, as the current module lacks this feature. On the admin side, enhancements are also being planned to address current limitations such as the inability to sort data and the mixing of both ‘Pending’ and ‘Paid’ statuses. The improvement will include prioritizing the display of pending payments to help administrators focus on unresolved transactions. Additionally, a pagination system is planned to replace the current slider-based navigation, making it easier for admins to browse through payment records page

by page. Although admins will only have viewing rights, these planned updates aim to enhance their overall experience in monitoring payment activity.

Module 4: Manage Login

The *Manage Login* module is an essential part of the Plasticware SME System, responsible for authenticating users and granting secure access to the platform. This module ensures that users such as Customers and Shop Owners can log in to the system using their registered credentials. Initially, the login process relied solely on traditional email and password authentication. However, this approach presented usability challenges, especially for users who frequently forget their passwords or encounter difficulty typing on mobile devices. To enhance both security and user convenience, the module has been improved with two major features: Google Social Login and a toggle password visibility option. The addition of these features aims to modernize the login experience, simplify access for users, and reduce password-related login issues.

1.2 Module Delegation

All members will be given modules to be managed and maintained and modules to be evolved. Table 1.1 below shows the details of the task delegation presented to the team members.

Name	**Student ID**	**Module Assigned**	**Maintenance Type**
NUR IZZAH AFIAH BINTI MOHD MAZAILI	CB22033	Manage Order	• Corrective Maintenance (Admin confused when user buys item, no clear new order alert.) • Perfective Maintenance (Users cannot view past purchases.)
VARSHINI JAGARAJAN	CB22139	Manage Product	• Perfective Maintenance: 1. Implemented a search feature with filters (by product name, category, and status) to enhance product accessibility and user navigation. 2. Established automatic updates for product status (e.g., “Out of Stock,” “Low Stock,” “Available”) according to product quantity to enhance inventory clarity. • Adaptive Maintenance: Introduced activity log support, allowing for the monitoring of product actions such as create, update, and delete, enhancing system visibility and audit preparedness.

MUHAMMAD HARIZ HAIKAL BIN NORISMAN	CB23130	Manage Payment	• Perfective Maintenance (No transaction bank procedure) • Perfective Maintenance (Not prioritizing pending status and does not have any sorting logic)
MUHAMAD ADAM BIN SHAFIE	CB22055	Manage Login	• Perfective Maintenance (Implement user registration and login via Google account.) • Perfective Maintenance (Allow users to toggle password visibility during login)

1.3 Report Organization

This technical report documents the software evolution and maintenance efforts carried out for the Plasticware SME System (PSS). The report is structured into three main chapters: introduction, tool setup, and change implementation. The introduction provides background information about the PSS project, outlines its purpose, and highlights the specific issues addressed by the team, including the assigned modules for each member and the overall report structure. The second chapter focuses on the tools and technologies used for the project, such as the development environment, version control system, and frameworks, as well as the setup and structure of the project files. The third chapter discusses the maintenance approach applied to each module, the process of handling changes requests, proposed improvements, and the actual implementation of those changes. This chapter also includes the individual contributions of each team member based on their assigned modules.

2.0 Change Implementation

2.1 Manage Payment – User Transaction (MUHAMMAD HARIZ HAIKAL BIN NORISMAN)

This module implements the perfective maintenance to improve the payment process for the users specifically focusing on enhancing the online payment procedure. Currently, the system did not utilize the database transaction which could lead to inconsistencies such as payment details being saved without confirming the full process had been complete successfully. If an error occurred midway through the payment process, only partial data could be saved or the payments might not be properly recorded. This will lead to incomplete orders or mismatched payment records. With this, a transaction procedure has been implemented to the payment process. This ensures that all payment operations such as saving user payment details, update the order status and storing the bank info are executed and secure the database transaction. The perfective maintenance improves the integrity of the payment module and gives users a more stable and trustworthy payment experience. The goal of the maintenance is to ensure a reliable payment process for online transactions and prevent partial or failed payment entries.

2.1.1 Change Request Form for Manage Payment

CHANGE-REQUEST FORM			
Project Name	Plasticware SME System		
Change ID	CR-PSME-02-05	**Date**	11.06.2025
Change Requestor	Muhammad Hariz Haikal Bin Norisman	**Name of Request**	Perfective Maintenance – Add Transaction Process
Contact Number	012-7427471		
Change Description	Create a proper database transaction procedure for the online payment process within the Manage Payment module.		
Change Reason	• The module lacked proper transaction handling which lead to data inconsistencies and incomplete payment records. • User does not have any transaction process when choose the online payment.		
Change Impact	• Improves data reliability and payment accuracy for users. • Enhances user experience in the online payment process. • Have new database information of the user bank account.		
Proposed Action	• Add the new interface for the user to fill in the form of their bank account information. • Add new function in the ManageCartController where it will show the interface of the bank account information. • Add new function in the ManageCartController where it will save the user bank account information in the database. • Adjust the function in the ManageCartController to change the current status of the payment in the order table in database. • Add new model ‘banking’ in the system to save the bank account information.		

The Priority of the change [/]	Low	Medium	High
			/

APPROVALS TO PROCCED:			
Approval Status:	**Approval Date:**	**Approval By:**	**Project Leader:**
Approved	11.06.2025		

2.1.2 Proposed Change

The proposed change for this module is to add the transaction procedure during the online payment process. Currently, when the users choose the online payment, the system will automatically save the data into the database table which say that the user successfully pay for their order but the payment status for their order still be pending. For example in the Figure 2.2 where after the user confirm on online payment payment it will display the order progress.

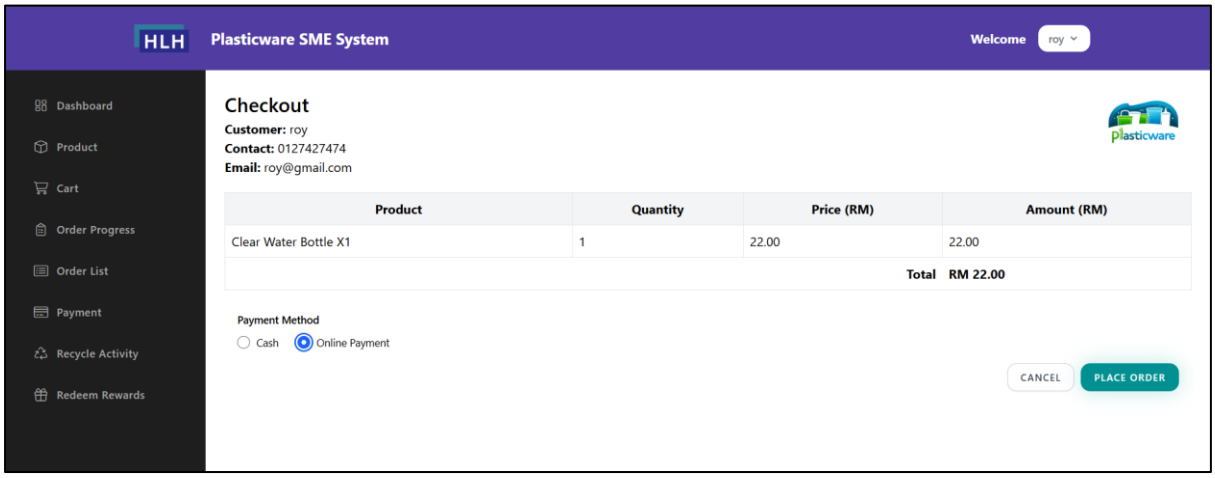


Figure 2.1 Checkout Interface

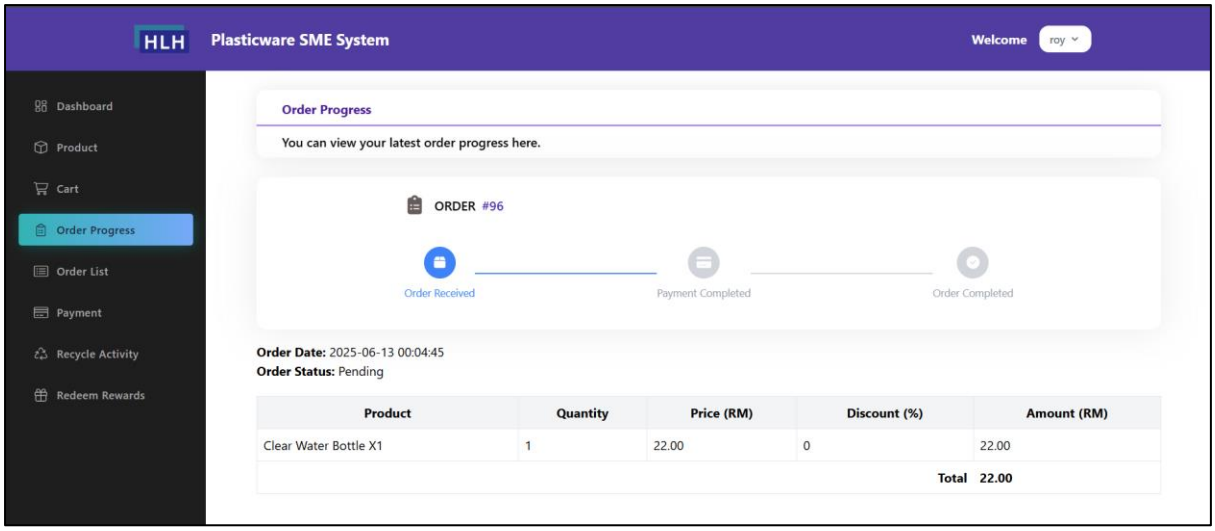


Figure 2.2 Order Progress Interface

2.1.3 Implemented Change

In figure 2.3, first implemented change is by adding the new model ‘banking’ to save the bank account information and the order id of the user order.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

2 references | 0 implementations
class Banking extends Model
{
    use HasFactory;

    0 references
    protected $fillable = [
        'user_id',
        'order_id',
        'bank_name',
        'account_holder',
        'reference_number',
    ];

    0 references | 0 overrides
    public function user(): BelongsTo
    {
        return $this->belongsTo(related: User::class);
    }

    0 references | 0 overrides
    public function order(): BelongsTo
    {
        return $this->belongsTo(related: Order::class);
    }
}

```

Figure 2.3 Banking Model Code

Figure 2.4 is the new banking form for the user to enter their bank account information after choose the online payment to submit their order in the system. This new interface help to get the user bank account information save in the database.

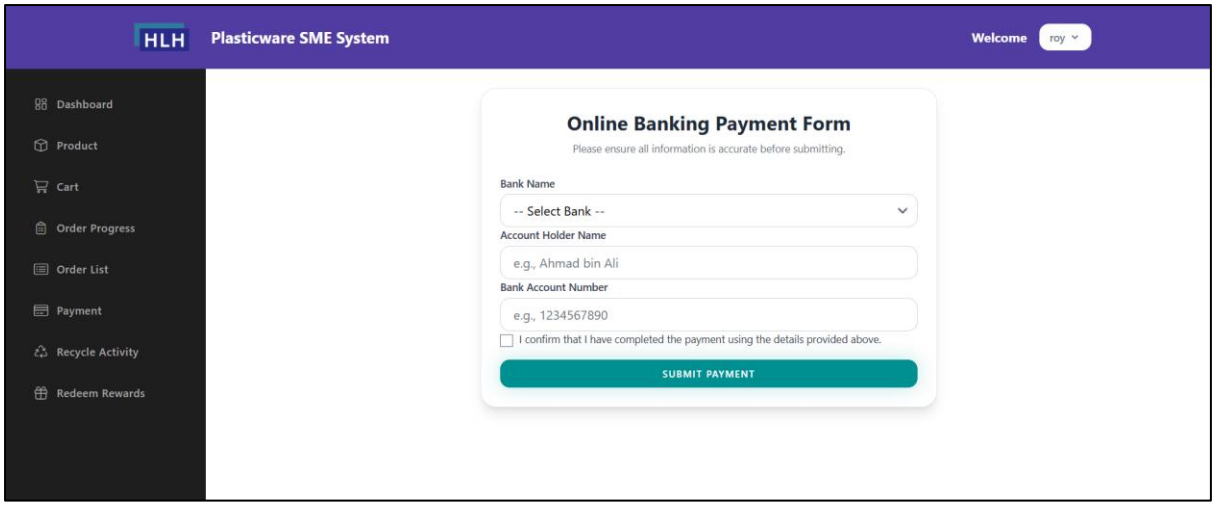


Figure 2.4 Online Banking Payment Form

```
<x-app-layout>
    <div class="max-w-xl mx-auto mt-10 bg-white p-6 rounded-lg shadow-lg border border-gray-200">
        <div class="mb-6 text-center">
            <h2 class="text-2xl font-bold text-gray-800">Online Banking Payment Form</h2>
            <p class="text-sm text-gray-500 mt-1">Please ensure all information is accurate before submitting.</p>
        </div>

        <form action="{{ route(name: 'banking.submit') }}" method="POST" class="space-y-5">
            @csrf
            <input type="hidden" name="order_id" value="{{ $order_id }}">

            <!-- Bank Name -->
            <div>
                <label for="bank_name" class="block text-sm font-medium text-gray-700 mb-1">Bank Name</label>
                <select name="bank_name" id="bank_name" class="w-full border border-gray-300 px-4 py-2 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500" required>
                    <option value="">-- Select Bank --</option>
                    <option value="Maybank">Maybank</option>
                    <option value="CIMB">CIMB</option>
                    <option value="RHB">RHB</option>
                    <option value="Public Bank">Public Bank</option>
                    <option value="Bank Islam">Bank Islam</option>
                    <option value="Hong Leong Bank">Hong Leong Bank</option>
                    <option value="Ambank">Ambank</option>
                    <option value="Bank Rakyat">Bank Rakyat</option>
                </select>
            </div>

            <!-- Account Holder -->
            <div>
                <label for="account_holder" class="block text-sm font-medium text-gray-700 mb-1">Account Holder Name</label>
                <input type="text" name="account_holder" id="account_holder" placeholder="e.g., Ahmad bin Ali" class="w-full border border-gray-300 px-4 py-2 rounded-md
                focus:outline-none focus:ring-2 focus:ring-blue-500" required>
            </div>

            <!-- Reference Number -->
            <div>
                <label for="reference_number" class="block text-sm font-medium text-gray-700 mb-1">Bank Account Number</label>
                <input type="text" name="reference_number" id="reference_number" placeholder="e.g., 1234567890" class="w-full border border-gray-300 px-4 py-2 rounded-md
                focus:outline-none focus:ring-2 focus:ring-blue-500" required>
            </div>

            <!-- Confirmation Checkbox -->
            <div class="flex items-start">
                <input type="checkbox" name="confirmation" id="confirmation" required class="mt-1 mr-2">
                <label for="confirmation" class="text-sm text-gray-700">
                    I confirm that I have completed the payment using the details provided above.
                </label>
            </div>

            <!-- Submit Button -->
            <div class="pt-4">
                <x-button class="w-full justify-center">Submit Payment</x-button>
            </div>
        </form>
    </div>
</x-app-layout>
```

Figure 2.5 Online Banking Payment Form Code

Then, in the manageCartController the payment status had been change from always set to 'Pending' to a conditional logic, if the user selects Online Payment, status is set to 'Paid' while if the user selects cash, the status will remain 'Pending'. The old code will redirected to the same route after order placement regardless of payment type while the new code will have if online payment is chosen, the user will redirected to the banking form to enter bank details and if the user chose cash it will remain the same as the old code.

```
// Create payment record
$paymentStatus = $request->payment_type === 'Online Payment' ? 'Paid' : 'Pending';

$order->payment()->create(attributes: [
    'payment_amount' => $order->order_total_price,
    'payment_status' => $paymentStatus,
    'payment_type' => $request->payment_type,
]);

// Delete selected cart items
Cart::whereIn(column: 'id', values: $request->cart_ids)->delete();
session()->forget(keys: ['cartItems', 'totalAmount']);

if ($request->payment_type === 'Online Payment') {
    return redirect()->route(route: 'banking.form', parameters: ['order_id' => $order->id]);
}

return redirect()->route(route: 'order.progress')->with(key: 'success', value: 'Order placed successfully.');
```

Figure 2.6: Old Code of manageCartController

```
// Create payment record
$order->payment()->create(attributes: [
    'payment_amount' => $order->order_total_price,
    'payment_status' => 'Pending',
    'payment_type' => $request->payment_type,
]);

// Delete selected cart items
$cart = Cart::whereIn(column: 'id', values: $request->cart_ids)->delete();

// Clear session data
session()->forget(keys: ['cartItems', 'totalAmount']);

return redirect()->route(route: 'order.progress')->with(key: 'success', value: 'Order placed successfully.');
}
```

Figure 2.7: New Code of manageCartController

In manageCart Controller also had add two new function name showBankingFrom() and submitBankingForm(). In showBankingForm(), it is use for displaying the banking form view to the user when they select online payment during checkout. Then, in submitBankingForm() is to handles the form submission when the user fills in and submits their banking details. First it will creates a new record in the banking model to store the user_id, order_id, bank_name, account_holder and reference_number from the user input. After saving the data, the user will redirected to the order.progress route to finish up the order process.

```php
    1 reference | 0 overrides
    public function showBankingForm($order_id): View
    {
        return view(view: 'banking.form', data: ['order_id' => $order_id]);
    }

    1 reference | 0 overrides
    public function submitBankingForm(Request $request): RedirectResponse
    {
        $request->validate(rules: [
            'order_id' => 'required|exists:orders,id',
            'bank_name' => 'required|string|max:255',
            'account_holder' => 'required|string|max:255',
            'reference_number' => 'required|string|max:255',

        ]);

        Banking::create(attributes: [
            'user_id' => auth()->id(),
            'order_id' => $request->order_id,
            'bank_name' => $request->bank_name,
            'account_holder' => $request->account_holder,
            'reference_number' => $request->reference_number,
        ]);

        return redirect()->route(route: 'order.progress')->with(key: 'success', value: 'Payment completed successfully.');
    }
}

```

Figure 2.8: Banking Function

2.2 Manage Payment – New Admin Payment List Interface (MUHAMMAD HARIZ HAIKAL BIN NORISMAN)

The manage payment module within the Plasticware SME System is allowing the administrator to monitor all the user payment transactions within the system. Currently, the admin payment list interface faced several usability issues, such as the inability to sort payments by amount or by date, making it difficult for the admin to analyze payment trends efficiently. Then, the table displayed both ‘Pending’ and ‘Paid’ statuses together without prioritization which hindered the admin to focus on transactions that still required action. Another key issue was the use the use of a continuous scrolling table which made navigation through numerous records time-consuming and inefficient. To solve these problems, the interface has been improved by introducing sorting functionalities for amount and date, as well ass ensuring that ‘Pending’ payments are prioritized at the top of the list. Furthermore, the slider table will be replace with a pagination-based layout, allowing admins to view payment records in organized pages rather than scrolling endlessly. Pagination offers better performance, and easier navigation which enhancing the admin ability to manage and track payments more effective.

2.2.1 Change Request Form for Manage Payment

CHANGE-REQUEST FORM			
Project Name	Plasticware SME System		
Change ID	CR-PSME-02-06	**Date**	11.06.2025
Change Requestor	Muhammad Hariz Haikal Bin Norisman	**Name of Request**	Perfective Maintenance – Revamp the admin payment list interface.
Contact Number	012-7427471		
Change Description	Improvise admin payment list interface to streamline payment tracking, enhance usability and support better administrative decision making.		
Change Reason	• The interface lacked essential functionality for efficient tracking and management. • Admins were unable to sort payments by amount or date, which difficult to monitor high value transactions or recent activities. • Pending and paid transactions were mixed together in the list leading to confusion to prioritize unresolved payments.		
Change Impact	• Improves the usability and efficiency of the admin panel in managing user payments. • Admins can quickly identify key transactions and recent payment activities to focus on unresolved or outstanding transaction first.		
Proposed Action	• Add sorting functionalities that allow administrators to sort payments based on total amount and by payment date. • Modify the display logic to ensure payments with a ‘Pending’ status are shown at the top of the list. • Replace the slider table with a pagination system.		

The Priority of the change [/]	Low	Medium	High
		/	
APPROVALS TO PROCCED:			
Approval Status:	**Approval Date:**	**Approval By:**	**Project Leader:**
Approved	11.06.2025		

2.2.2 Proposed Change

The proposed change for this module is to revamp the existing admin payment list interface to improve usability and data visibility. Currently in the payment list view it only show the table and graph data of the payment data in the system as shown in figure 2.9. The interface does not have the sorting by logic and not prioritizing the 'Pending' status.

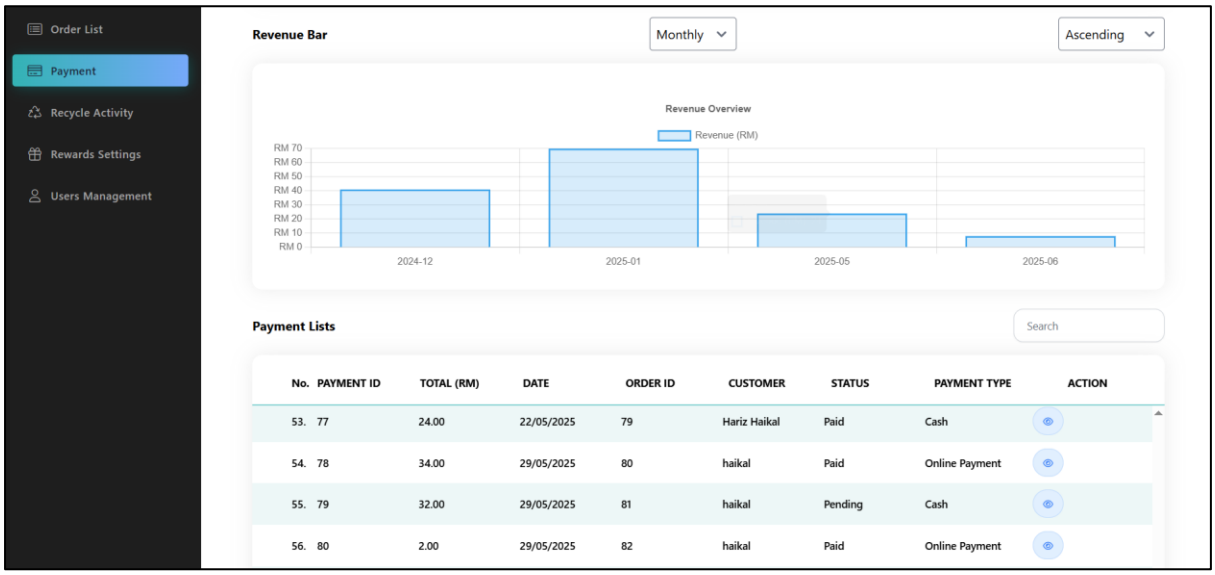
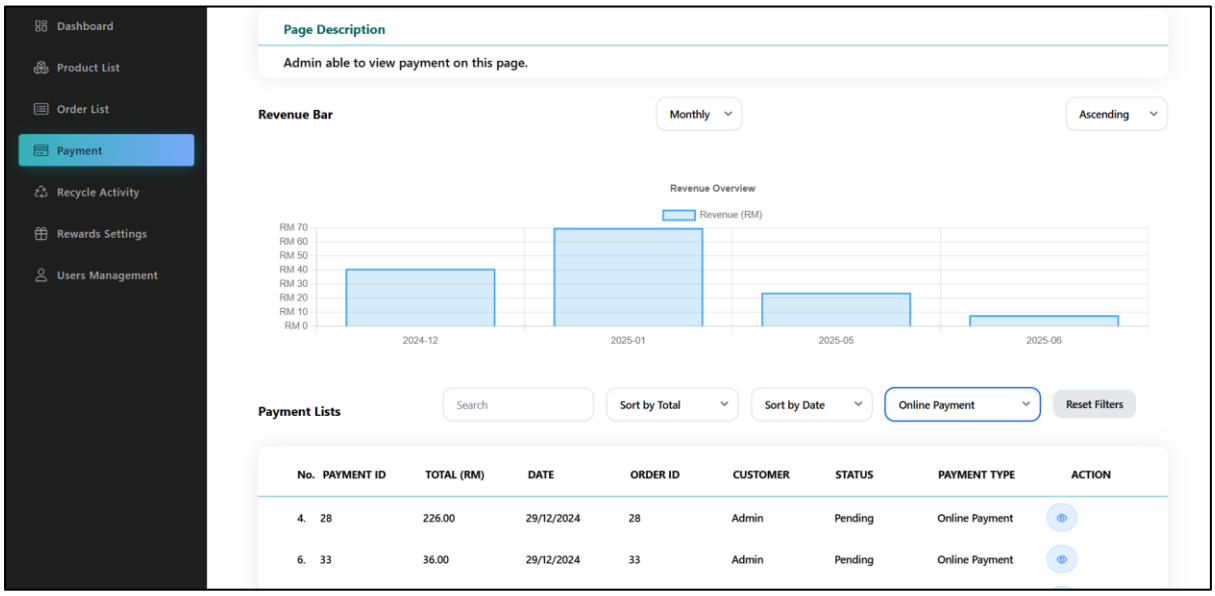


Figure 2.9 Old Payment List Interface.

2.2.3 Implemented Change

The first function method that had been added is the sorting by amount with option from highest to lowest and lowest to highest and sorting by date with option from latest to oldest and oldest to latest. The interface also implementing the filtering type to filter it by payment type which is cash or online payment. Then, the new reset button also had been added to ease the admin to clear the sort and filter function as shown in figure 2.10.



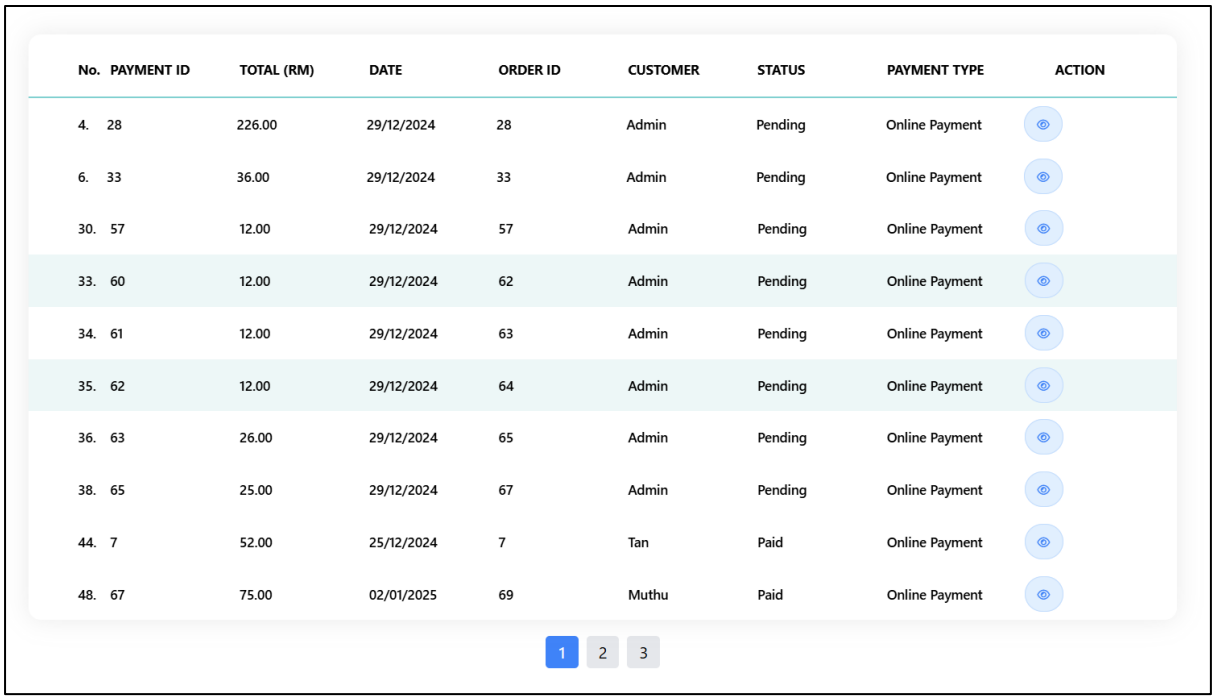


Figure 2.10 New Payment List Interface

In the new payment list new UI adds multiple filter inputs for amount date and type. It also includes a reset button for resetting the filter to the original filter.

```
<div class="grid grid-cols-3 gap-12 mx-10 my-6">
    <div class="col-span-3">
        <div class="flex justify-between items-center w-auto">
            <p class="font-bold text-md">Payment Lists</p>
            <div>
                <input type="text" id="searchInput" placeholder="Search"
                class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm focus:outline-none focus:ring {{ auth()->user()->hasRole(roles: 'admin') ?
                'focus:ring-primary-500 focus:border-primary-500' : 'focus:ring-purple-500 focus:border-purple-500' }}">
            </div>
        </div>
```

Figure 2.11 Old Payment List Before Sorting

```
40      <div class="grid grid-cols-3 gap-12 mx-10 my-6">
41          <div class="col-span-3">
42              <div class="flex justify-between items-center w-auto mb-2">
43                  <p class="font-bold text-md">Payment Lists</p>
44                  <div class="grid grid-cols-1 gap-4 mx-10 mb-4">
45          <div class="flex gap-4 items-center">
46      <input type="text" id="searchInput" placeholder="Search"
47          class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm focus:outline-none focus:ring {{ auth()->user()->hasRole(roles: 'admin') ? 'focus:ring-primary-500
        focus:border-primary-500' : 'focus:ring-purple-500 focus:border-purple-500' }}">
48
49      <select id="sortAmount" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
50          <option value="">Sort by Total</option>
51          <option value="high">Highest to Lowest</option>
52          <option value="low">Lowest to Highest</option>
53      </select>
54
55      <select id="sortDate" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
56          <option value="">Sort by Date</option>
57          <option value="latest">Latest to Oldest</option>
58          <option value="oldest">Oldest to Latest</option>
59      </select>
60
61      <select id="filterType" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
62          <option value="">Filter by Payment Type</option>
63          <option value="Cash">Cash</option>
64          <option value="Online Payment">Online Payment</option>
65      </select>
66
67      <button id="resetFilters" class="py-2 px-4 bg-gray-200 hover:bg-gray-300 rounded-md text-sm font-semibold">
68          Reset Filters
69      </button>
70  </div>
```

Figure 2.12 New Payment List After Sorting

The new code has adds custom HTML data attribute which are data-total, data-date, data-type, data-name and data-status for JavaScript based client side filtering. It also add the sorts pending before paid using sortBy. At const pendingRows and paidRows is to make the ‘Pending’ payments always show up first. This separation makes it easier to maintain visual priority where ‘Pending’ items are always more prominent.

```blade
<x-show-table :headers="['Payment ID', 'Total (RM)', 'Date', 'Order ID', 'Customer', 'Status', 'Payment Type', 'Action']">
    <tbody id="paymentTableBody" class="w-full">
    @php
        $sortedPayments = $payments->sortBy(function ($payment): int {
            return $payment->payment_status === 'Pending' ? 0 : 1;
        });
    @endphp
    @foreach ($sortedPayments as $i => $payment)
        <tr class="flex px-8 py-2 {{ auth()->user()->hasRole(roles: 'admin') ? (($loop->index % 2 == 0) ? 'bg-primary-50' : '') : (($loop->index % 2 == 0) ?
        'bg-purple-50' : '') }}"
            data-total="{{ $payment->payment_amount }}"
            data-date="{{ $payment->created_at }}"
            data-type="{{ $payment->payment_type }}"
            data-name="{{ strtolower(string: $payment->order->user->name) }}"
            data-status="{{ $payment->payment_status }}">
            <td class="mx-4 py-2 text-gray text-sm font-semibold w-4">{{ $loop->iteration }}.</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->id }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_amount }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->created_at->format('d/m/Y') }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->order->id }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->order->user->name }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_status }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_type }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">
                <a href="{{ route(name: 'view.payment', parameters: $payment->order_id) }}" class="rounded-full py-2 px-3 bg-blue-100 border border-blue-200
                justify-center items-center hover:bg-blue-200 ml-2">
                    <i class="fa-regular fa-eye text-blue-500 fa-sm"></i>
                </a>
            </td>
        </tr>
    @endforeach
</tbody>
```

```javascript
function filterAndSortTable() {
    let filtered = [...rows];

    const type = filterType.value;
    if (type) {
        filtered = filtered.filter(row => row.dataset.type === type);
    }

    const search = searchInput.value.toLowerCase();
    if (search) {
        filtered = filtered.filter(row => row.dataset.name.includes(search));
    }

    const pendingRows = filtered.filter(row => row.dataset.status === 'Pending');
    const paidRows = filtered.filter(row => row.dataset.status !== 'Pending');

    const sortRows = (list) => {
        if (sortAmount.value === 'high') {
            list.sort((a, b) => b.dataset.total - a.dataset.total);
        } else if (sortAmount.value === 'low') {
            list.sort((a, b) => a.dataset.total - b.dataset.total);
        }

        if (sortDate.value === 'latest') {
            list.sort((a, b) => new Date(b.dataset.date) - new Date(a.dataset.date));
        } else if (sortDate.value === 'oldest') {
            list.sort((a, b) => new Date(a.dataset.date) - new Date(b.dataset.date));
        }

        return list;
    };

    const sortedPending = sortRows(pendingRows);
    const sortedPaid = sortRows(paidRows);

    const combined = [...sortedPending, ...sortedPaid];
    const totalPages = Math.ceil(combined.length / rowsPerPage);
    updatePagination(totalPages);

    const pageRows = paginate(combined, currentPage, rowsPerPage);
    tableBody.innerHTML = '';
    pageRows.forEach(row => tableBody.appendChild(row));
}
```

Figure 2.13 Prioritize Pending Code

3.0 Tools Setup

3.1 Tools Setup

Figure 3.1: Github logo

GitHub and Visual Studio Code (VS Code) were the primary tools used throughout the maintenance process of the Plasticware SME System. GitHub served as the central version control platform, allowing team members to collaborate efficiently by managing branches, commits, and pull requests. All source code files for the project were hosted in a GitHub repository, which maintained a complete and transparent history of changes. Each team member worked within their own feature branch to avoid code conflicts and facilitate focused development. Pull requests were submitted to merge updates into the main branch after review and approval. GitHub's built-in functionalities such as branching, merging, pull request tracking, and commit history played a vital role in ensuring organized collaboration. It also enabled rollback testing, where changes could be reverted in case of errors, preserving the stability of the system. The terms commonly used in this environment include commit, pull, branch, and merge, which describe the workflows that developers follow when collaborating on shared codebases.

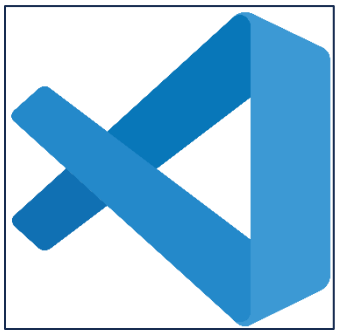


Figure 3.1: Visual Studio Code Logo

Visual Studio Code was used as the primary code editor for implementing and testing the maintenance changes. As a lightweight, open-source code editor developed by Microsoft, VS Code supports PHP and Blade syntax, making it highly suitable for Laravel development. It offers advanced features such as syntax highlighting, IntelliSense (for smart code suggestions), and seamless Git integration, which allowed team members to commit and push changes directly from the editor. The built-in terminal and debugging tools within VS Code also supported local testing and troubleshooting. Its extension marketplace provided useful tools like Laravel Blade snippets, GitLens for commit tracking, and PHP Debug for backend validation. VS Code's cross-platform compatibility and customizable layout made it a developer-friendly choice for the tasks involved in corrective and perfective maintenance. These tools collectively ensured that the team could efficiently code, test, track, and manage the entire maintenance cycle with minimal disruption and maximum productivity.

3.2 Project Structure Implementation

3.2.1 Main View of Project with Repository

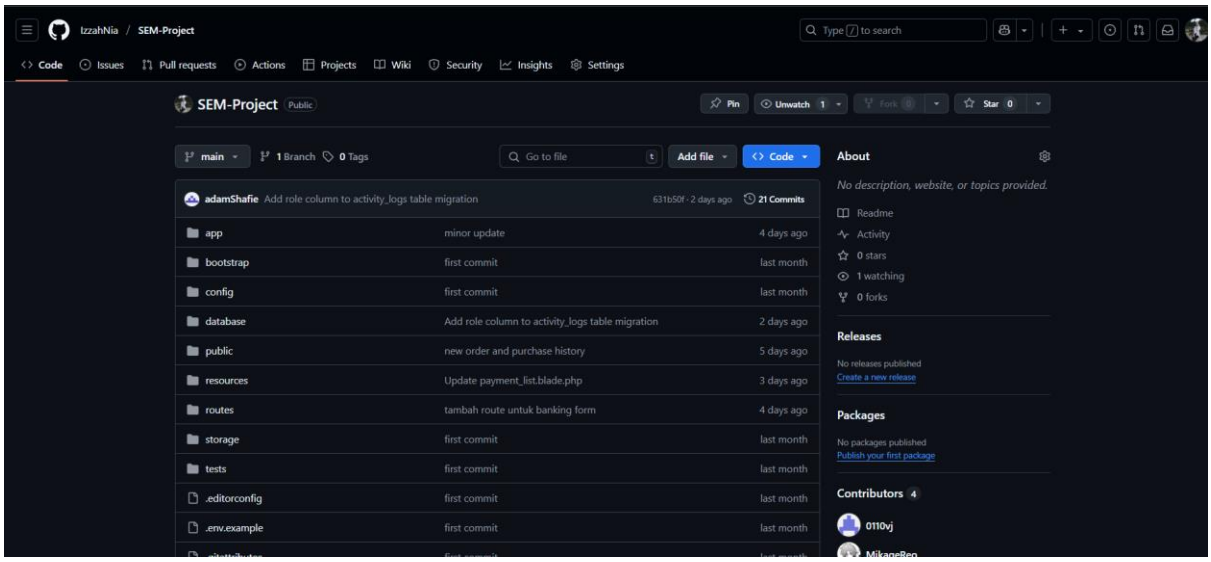


Figure 3.2 : Source Code of Platicware SME in GitHub

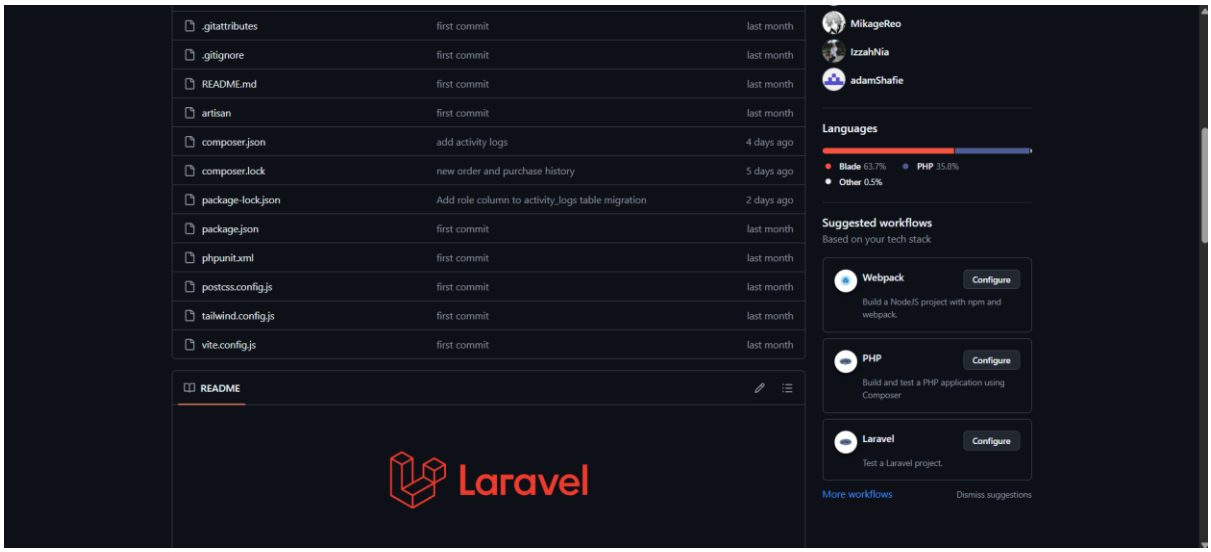
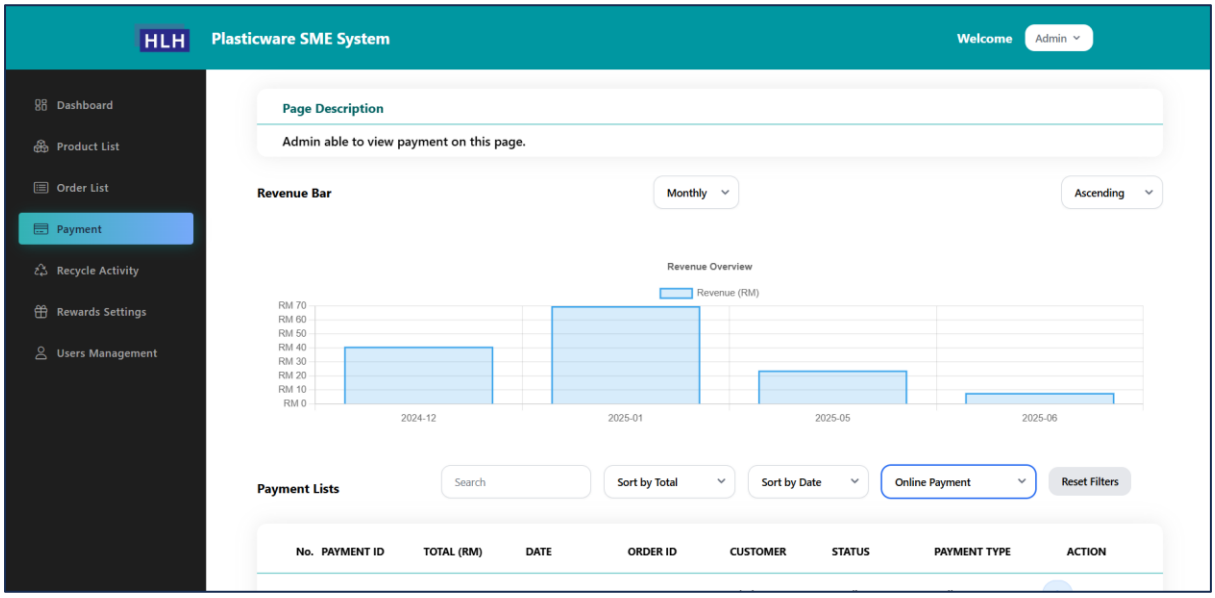


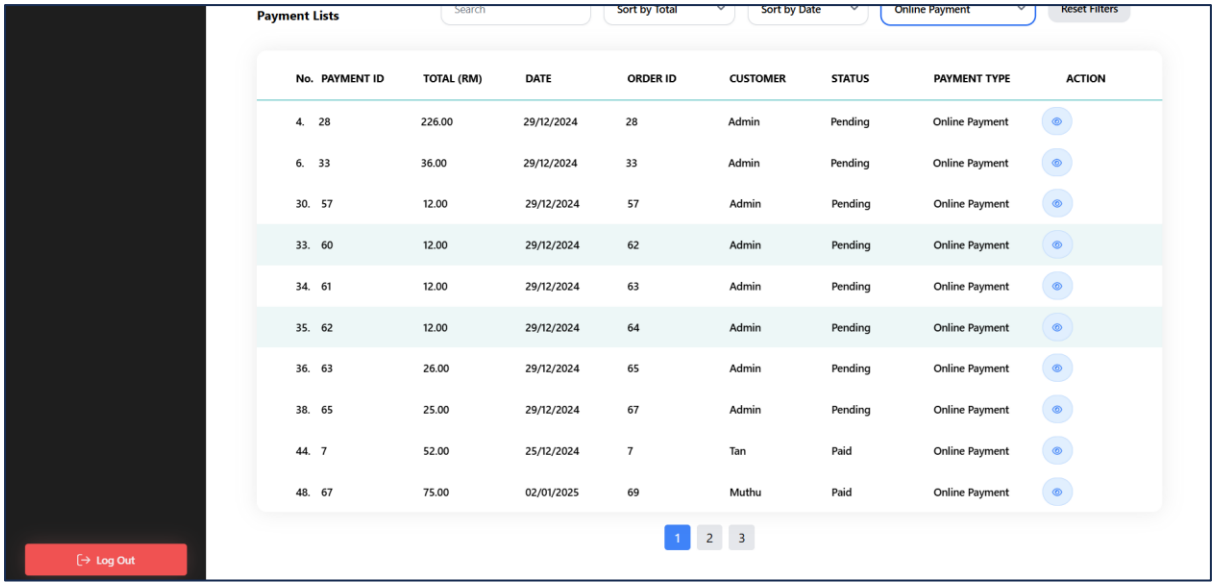
Figure 3.3 : Source Code of Platicware SME in GitHub

3.2.2 View of Project

3.2.2.1 Manage Payment (MUHAMMAD HARIZ HAIKAL BIN NORISMAN)

The improvements added sorting functionalities that allow users to organize the payment list by total amount, date, and payment type. Additionally, a filtering system has been implemented to quickly view specific types of payments. To improve usability and task prioritization, the system also implements a logic that automatically prioritizes and displays payments with a "Pending" status at the top of the table. This prioritization ensures that critical and unresolved payments receive immediate attention, improving workflow efficiency and decision-making. These enhancements contribute to a more intuitive and responsive user interface for better payment tracking and management.





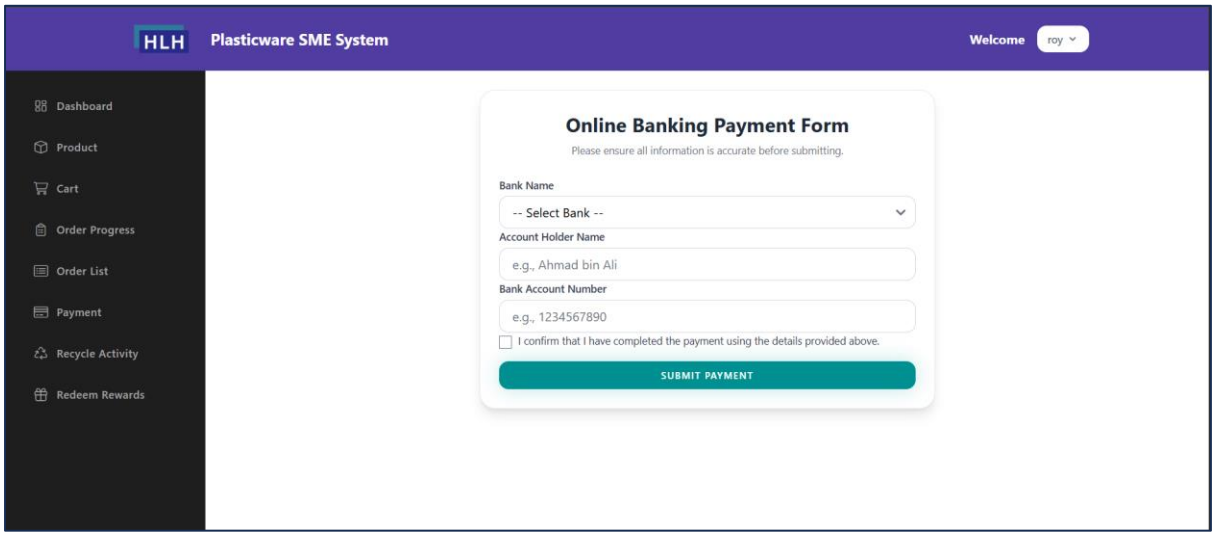

```blade
    <div class="grid grid-cols-3 gap-12 mx-10 my-6">
        <div class="col-span-3">
            <div class="flex justify-between items-center w-auto mb-2">
                <p class="font-bold text-md">Payment Lists</p>
                <div class="grid grid-cols-1 gap-4 mx-10 mb-4">
        <div class="flex gap-4 items-center">
    <input type="text" id="searchInput" placeholder="Search"
        class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm focus:outline-none focus:ring {{ auth()->user()->hasRole(roles: 'admin') ? 'focus:ring-primary-500 focus:border-primary-500' : 'focus:ring-purple-500 focus:border-purple-500' }}">

    <select id="sortAmount" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
        <option value="">Sort by Total</option>
        <option value="high">Highest to Lowest</option>
        <option value="low">Lowest to Highest</option>
    </select>

    <select id="sortDate" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
        <option value="">Sort by Date</option>
        <option value="latest">Latest to Oldest</option>
        <option value="oldest">Oldest to Latest</option>
    </select>

    <select id="filterType" class="py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-semibold">
        <option value="">Filter by Payment Type</option>
        <option value="Cash">Cash</option>
        <option value="Online Payment">Online Payment</option>
    </select>

    <button id="resetFilters" class="py-2 px-4 bg-gray-200 hover:bg-gray-300 rounded-md text-sm font-semibold">
        Reset Filters
    </button>
</div>
```

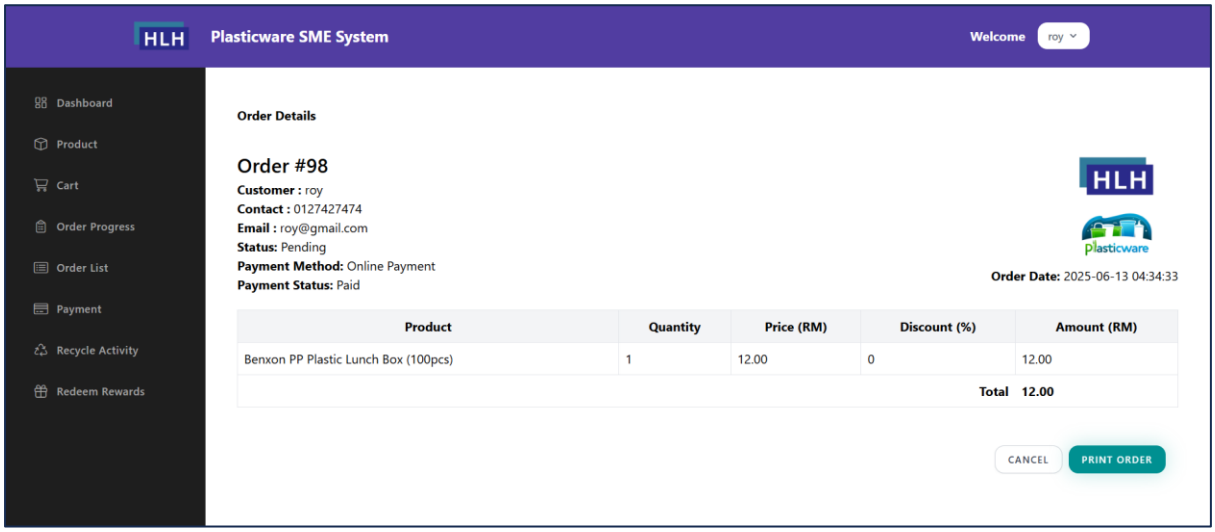
```blade
<x-show-table :headers="['Payment ID', 'Total (RM)', 'Date', 'Order ID', 'Customer', 'Status', 'Payment Type', 'Action']">
    <tbody id="paymentTableBody" class="w-full">
    @php
        $sortedPayments = $payments->sortBy(function ($payment): int {
            return $payment->payment_status === 'Pending' ? 0 : 1;
        });
    @endphp
    @foreach ($sortedPayments as $i => $payment)
        <tr class="flex px-8 py-2 {{ auth()->user()->hasRole(roles: 'admin') ? (($loop->index % 2 == 0) ? 'bg-primary-50' : '') : (($loop->index % 2 == 0) ? 'bg-purple-50' : '') }}"
            data-total="{{ $payment->payment_amount }}"
            data-date="{{ $payment->created_at }}"
            data-type="{{ $payment->payment_type }}"
            data-name="{{ strtolower(string: $payment->order->user->name) }}"
            data-status="{{ $payment->payment_status }}">
            <td class="mx-4 py-2 text-gray text-sm font-semibold w-4">{{ $loop->iteration }}.</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->id }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_amount }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->created_at->format('d/m/Y') }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->order->id }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->order->user->name }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_status }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">{{ $payment->payment_type }}</td>
            <td class="py-2 text-gray text-sm font-semibold text-left w-1/6">
                <a href="{{ route(name: 'view.payment', parameters: $payment->order_id) }}" class="rounded-full py-2 px-3 bg-blue-100 border border-blue-200 justify-center items-center hover:bg-blue-200 ml-2">
                    <i class="fa-regular fa-eye text-blue-500 fa-sm"></i>
                </a>
            </td>
        </tr>
    @endforeach
</tbody>
```

```
function filterAndSortTable() {
    let filtered = [...rows];

    const type = filterType.value;
    if (type) {
        filtered = filtered.filter(row => row.dataset.type === type);
    }

    const search = searchInput.value.toLowerCase();
    if (search) {
        filtered = filtered.filter(row => row.dataset.name.includes(search));
    }

    const pendingRows = filtered.filter(row => row.dataset.status === 'Pending');
    const paidRows = filtered.filter(row => row.dataset.status !== 'Pending');

    const sortRows = (list) => {
        if (sortAmount.value === 'high') {
            list.sort((a, b) => b.dataset.total - a.dataset.total);
        } else if (sortAmount.value === 'low') {
            list.sort((a, b) => a.dataset.total - b.dataset.total);
        }

        if (sortDate.value === 'latest') {
            list.sort((a, b) => new Date(b.dataset.date) - new Date(a.dataset.date));
        } else if (sortDate.value === 'oldest') {
            list.sort((a, b) => new Date(a.dataset.date) - new Date(b.dataset.date));
        }

        return list;
    };

    const sortedPending = sortRows(pendingRows);
    const sortedPaid = sortRows(paidRows);

    const combined = [...sortedPending, ...sortedPaid];
    const totalPages = Math.ceil(combined.length / rowsPerPage);
    updatePagination(totalPages);

    const pageRows = paginate(combined, currentPage, rowsPerPage);
    tableBody.innerHTML = '';
    pageRows.forEach(row => tableBody.appendChild(row));
}
```

A new page has been introduced, allowing users to submit their bank account statement details as part of the payment process. This enhancement ensures that payment verification includes the necessary financial documentation, improving traceability and trust.

In support of this functionality, a new Banking database model has been created to securely store the user's banking details submitted through the form. The system logic has also been updated so that when users select the Online Payment option, they are now required to complete the online transaction form before finalizing and placing their order. This improvement not only promotes better data collection but also enhances the integrity and accountability of the payment process.





	id	user_id	order_id	bank_name	account_holder	reference_number	created_at	updated_at
Edit Copy Delete	2	18	93	CIMB	Muhammad	154017028321	2025-06-12 02:13:47	2025-06-12 02:13:47
Edit Copy Delete	3	18	94	Bank Islam	Ahmad Bin Ali	123456789	2025-06-12 02:54:57	2025-06-12 02:54:57
Edit Copy Delete	4	18	98	Maybank	Roy Halique	154027012212	2025-06-13 04:34:57	2025-06-13 04:34:57

```php
public function submitBankingForm(Request $request): RedirectResponse
{
    $request->validate(rules: [
        'order_id' => 'required|exists:orders,id',
        'bank_name' => 'required|string|max:255',
        'account_holder' => 'required|string|max:255',
        'reference_number' => 'required|string|max:255',

    ]);

    Banking::create(attributes: [
        'user_id' => auth()->id(),
        'order_id' => $request->order_id,
        'bank_name' => $request->bank_name,
        'account_holder' => $request->account_holder,
        'reference_number' => $request->reference_number,
    ]);

    return redirect()->route(route: 'order.progress')->with(key: 'success', value: 'Payment completed successfully.');
}
```

```php
public function showBankingForm($order_id): View
{
    return view(view: 'banking.form', data: ['order_id' => $order_id]);
}
```

```blade
<x-app-layout>
    <div class="max-w-xl mx-auto mt-10 bg-white p-6 rounded-lg shadow-lg
border border-gray-200">
        <div class="mb-6 text-center">
            <h2 class="text-2xl font-bold text-gray-800">Online Banking
Payment Form</h2>
            <p class="text-sm text-gray-500 mt-1">Please ensure all
information is accurate before submitting.</p>
        </div>

        <form action="{{ route('banking.submit') }}" method="POST"
class="space-y-5">
            @csrf
            <input type="hidden" name="order_id" value="{{ $order_id }}">

            <!-- Bank Name -->
            <div>
                <label for="bank_name" class="block text-sm font-medium text-
gray-700 mb-1">Bank Name</label>
                <select name="bank_name" id="bank_name" class="w-full border
border-gray-300 px-4 py-2 rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500" required>
                    <option value="">-- Select Bank --</option>
                    <option value="Maybank">Maybank</option>
                    <option value="CIMB">CIMB</option>
```

```
                    <option value="RHB">RHB</option>
                    <option value="Public Bank">Public Bank</option>
                    <option value="Bank Islam">Bank Islam</option>
                    <option value="Hong Leong Bank">Hong Leong Bank</option>
                    <option value="Ambank">Ambank</option>
                    <option value="Bank Rakyat">Bank Rakyat</option>
                </select>
            </div>

            <!-- Account Holder -->
            <div>
                <label for="account_holder" class="block text-sm font-medium
text-gray-700 mb-1">Account Holder Name</label>
                <input type="text" name="account_holder" id="account_holder"
placeholder="e.g., Ahmad bin Ali" class="w-full border border-gray-300 px-4
py-2 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500" required>
            </div>

            <!-- Reference Number -->
            <div>
                <label for="reference_number" class="block text-sm font-medium
text-gray-700 mb-1">Bank Account Number</label>
                <input type="text" name="reference_number"
id="reference_number" placeholder="e.g., 1234567890" class="w-full border
border-gray-300 px-4 py-2 rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500" required>
            </div>

            <!-- Confirmation Checkbox -->
            <div class="flex items-start">
                <input type="checkbox" name="confirmation" id="confirmation"
required class="mt-1 mr-2">
                <label for="confirmation" class="text-sm text-gray-700">
                    I confirm that I have completed the payment using the
details provided above.
                </label>
            </div>

            <!-- Submit Button -->
            <div class="pt-4">
                <x-button class="w-full justify-center">Submit Payment</x-
button>
            </div>
        </form>
    </div>
</x-app-layout>
```

3.2.3 Members and role

GitHub repositories can be held by personal users or collectively managed by an organization. In user-owned repositories, the owner can invite collaborators, enabling various contributors to participate in the project. This project involves a repository that comprises three team members, featuring the project leader. Once the repository is established, the project leader, recognized by the GitHub ID, IzzahNia—will invite the remaining team members to participate as collaborators. Figure displays the roster of collaborators presently engaged in the project.

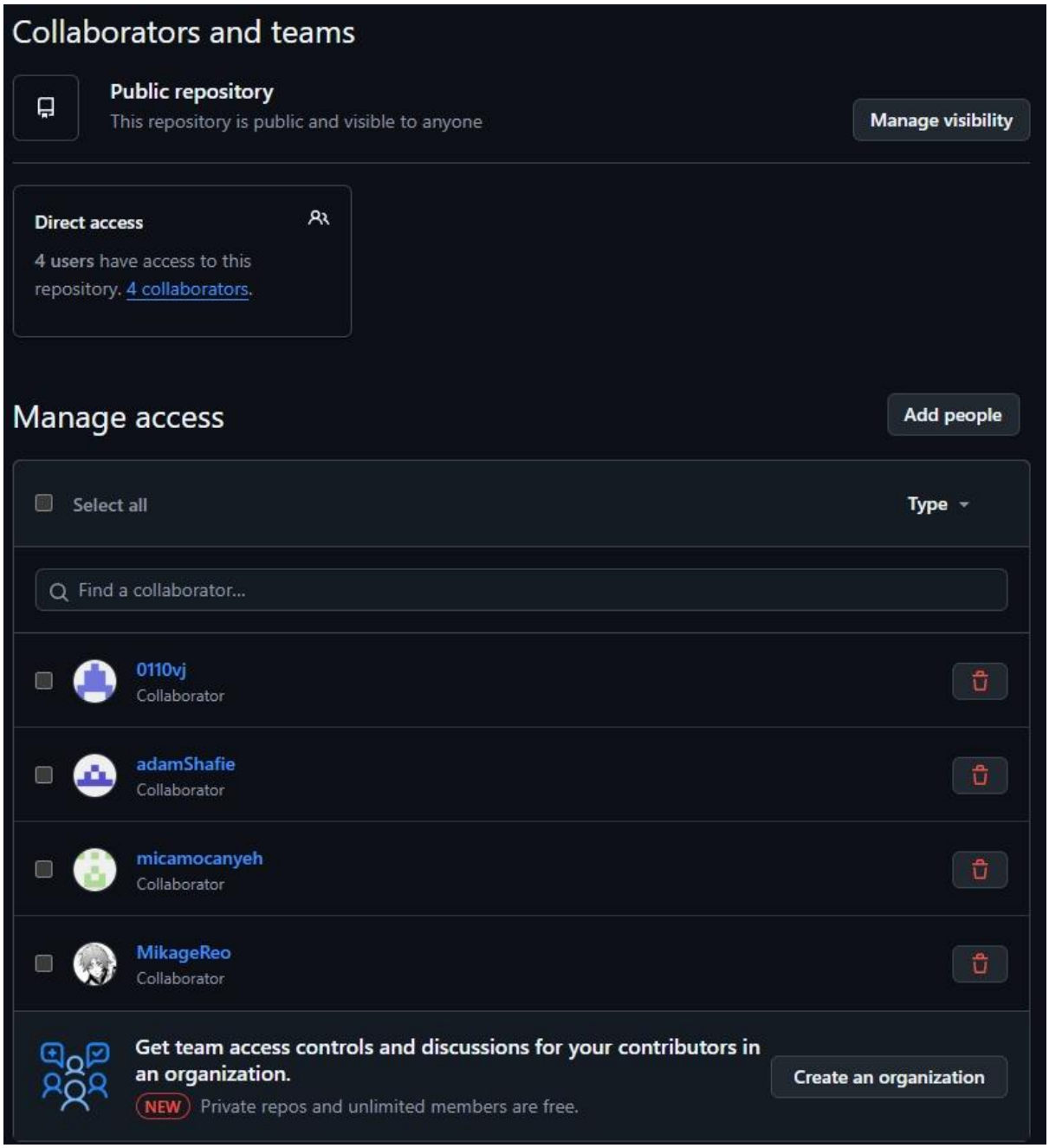


Figure 0.1 List of Collaborators involved

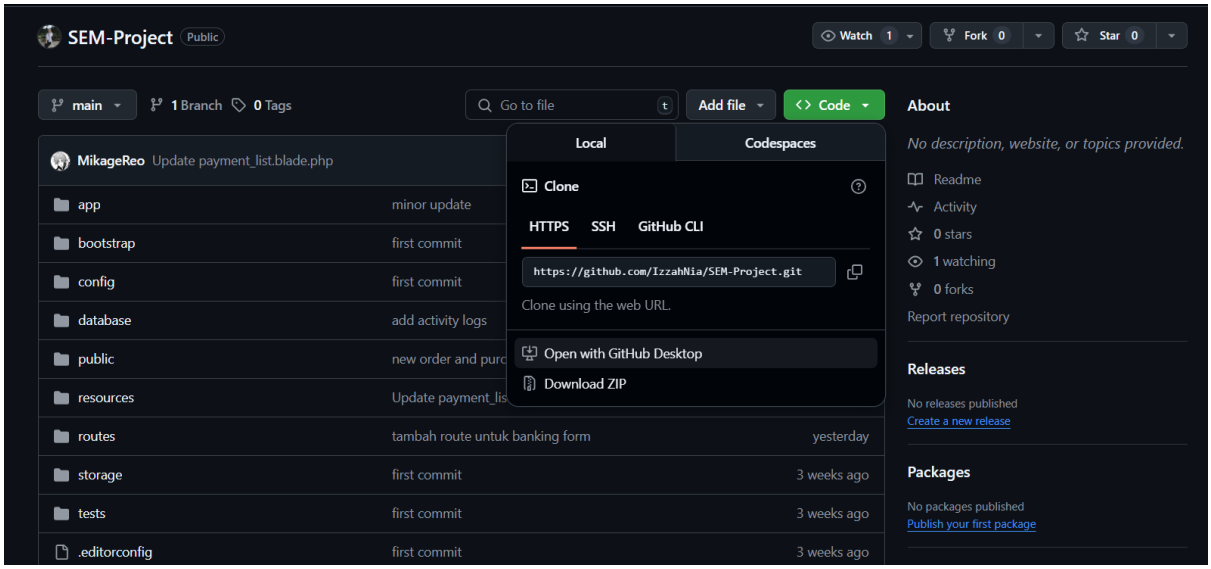
Every team member will be given particular modules to oversee, sustain, and enhance. The details of task distribution for each team member are outlined in Table ? below.

Table 0.1 Task Delegation

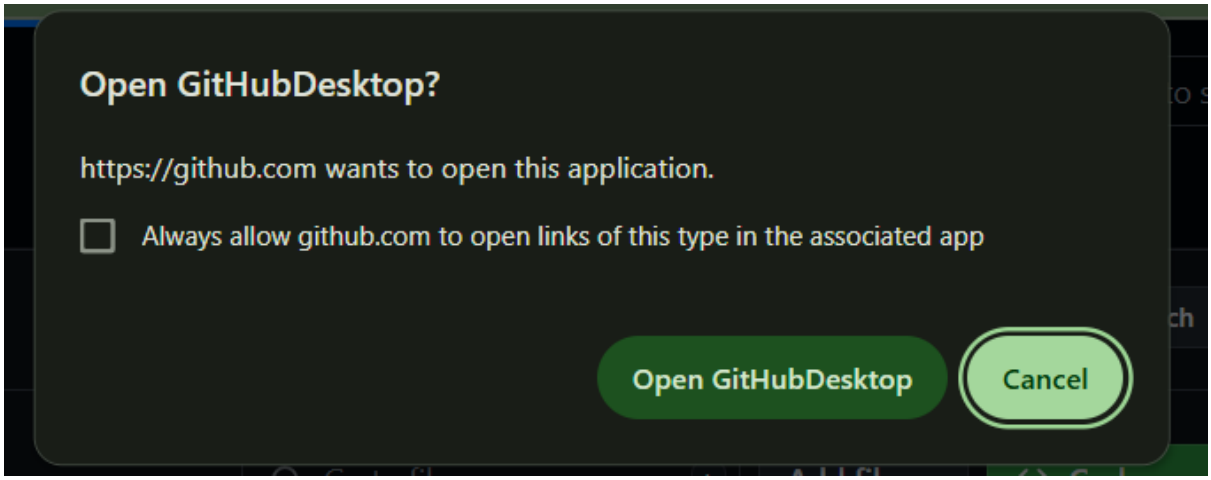
GitHub ID	Name	Student ID	Role	Module
IzzahNia	NUR IZZAH AFIAH BINTI MOHD MAZAILI	CB22033	Owner	Manage Order
0110vj	VARSHINI JAGARAJAN	CB22139	Collaborator	Manage Login and Manage Product
adamShafie	MUHAMAD ADAM BIN SHAFIE	CB22055	Collaborator	Manage Login
MikageReo	MUHAMMAD HARIZ HAIKAL BIN NORISMAN	CB23130	Collaborator	Manage Payment

3.2.4 Clone from existing project

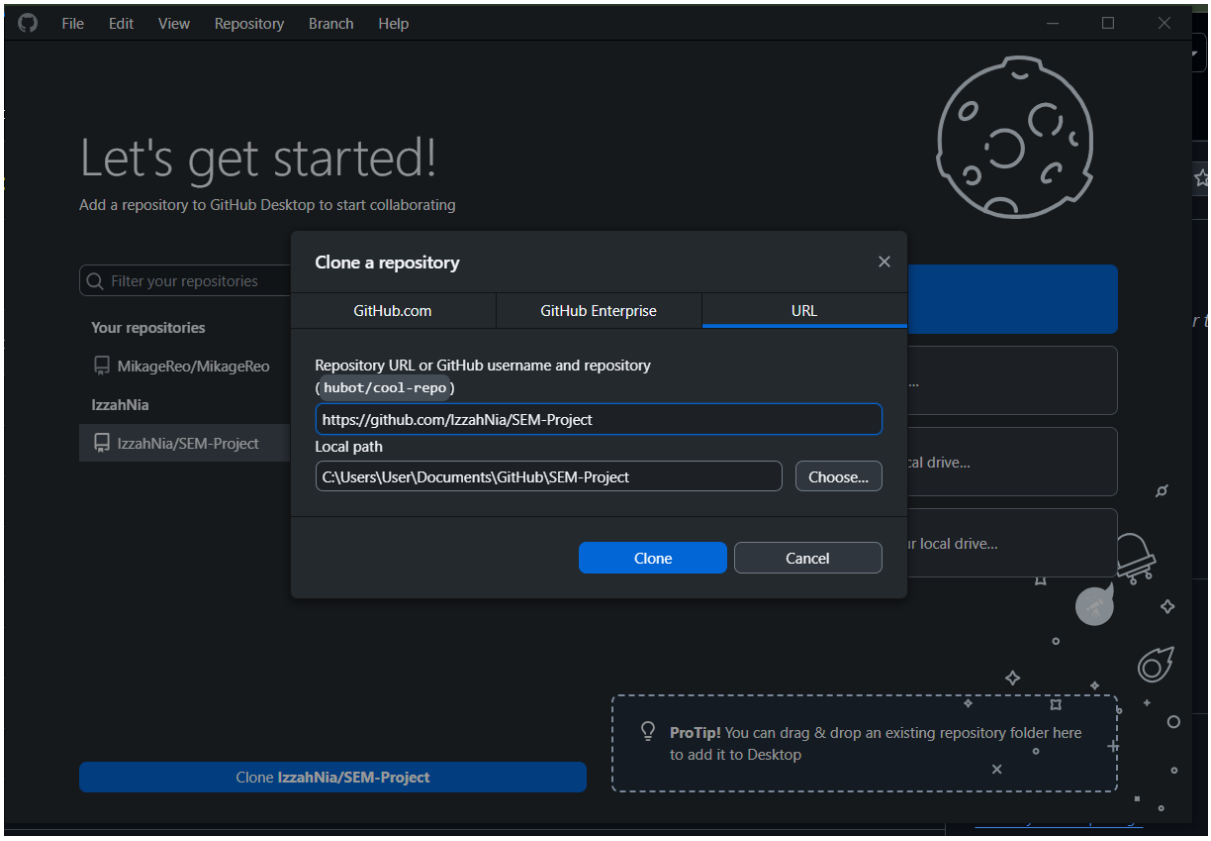
The project leader is responsible for publishing the project source code to the GitHub repository. Once published, team members should clone the code into their local files. To clone the project source code, initiate the process by clicking on the "Open with GitHub Desktop" option.



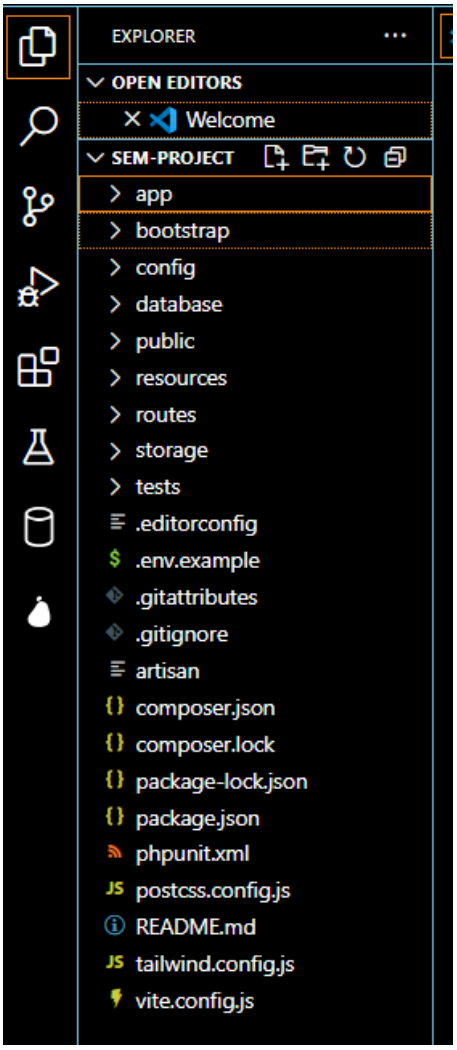
Next, after clicking the “Open with GitHub Desktop” option, a popup window to open the GitHub Desktop will appear. Click “Open GitHub Desktop”.



After opening GitHub Desktop, a window for cloning the repository will be presented. In this window, the GitHub URL link and the local path options will be visible to the user. Team members have the flexibility to choose their preferred local path. Once the local path is selected, proceed to click "Clone" to initiate the cloning process for the repository.



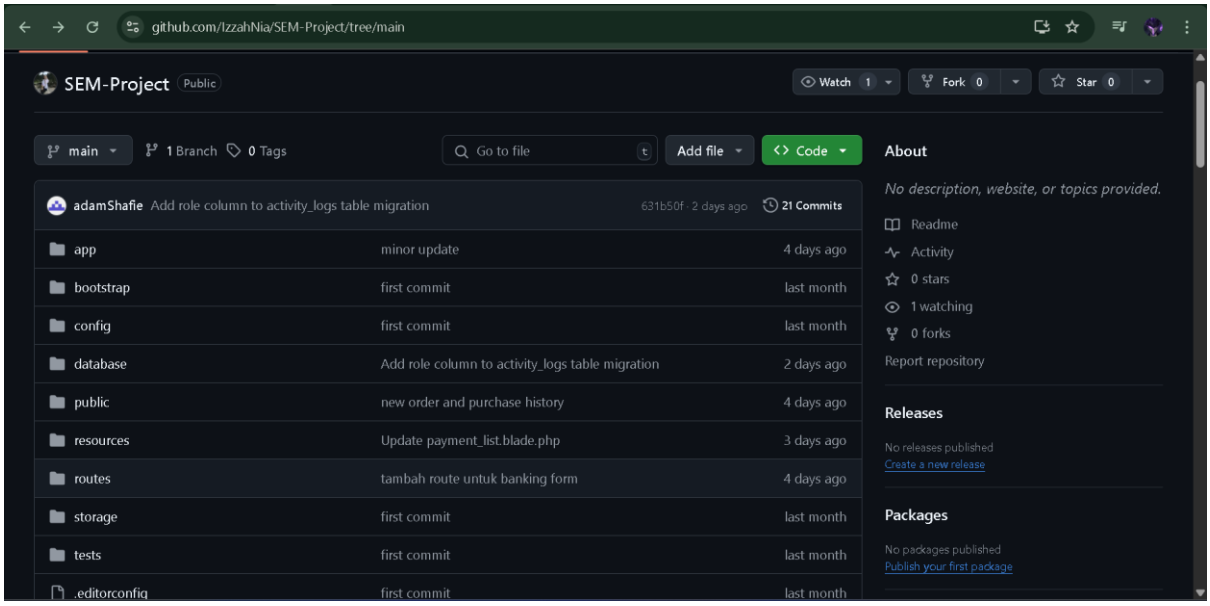
Next, the progress of cloning the repository. The project files, cloned from the GitHub repository, are illustrated in the figure below. All team members have the capability to edit the code in alignment with their respective change implementations.


EXPLORER
OPEN EDITORS
Welcome
SEM-PROJECT
app
bootstrap
config
database
public
resources
routes
storage
tests
.editorconfig
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
README.md
tailwind.config.js
vite.config.js

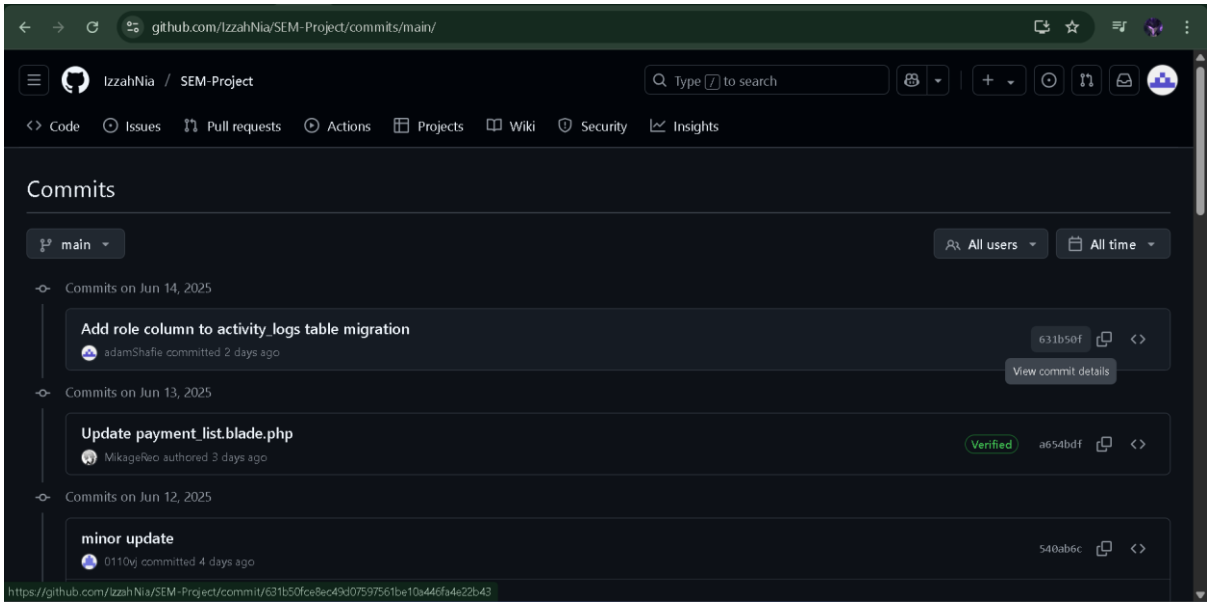
3.2.5 Build and Test Code on The Preferred Branch.

Once the modifications made by team members have undergone review and received approval from the leader, the "Run and Debug" tool in Visual Studio Code should be utilized to test the impact of changes in one module on the other modules. The detailed explanations of these processes are depicted in the figures shown below.

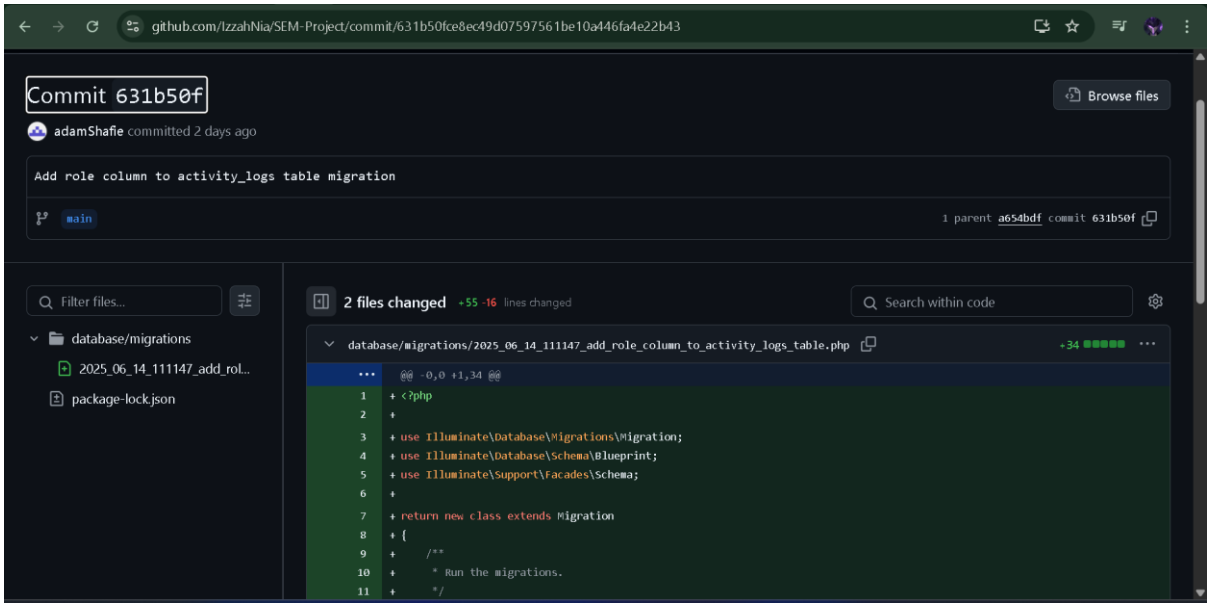
1. Click the "Commit" button on the main repository page of the project.



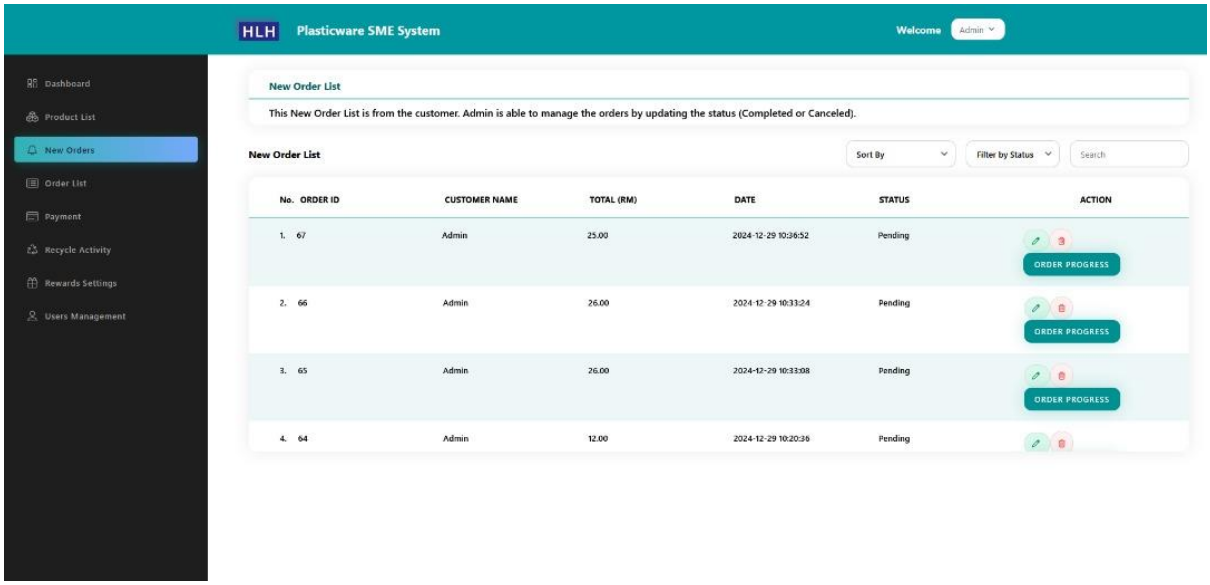
2. Select the desired commit and click on "View Commit Details."



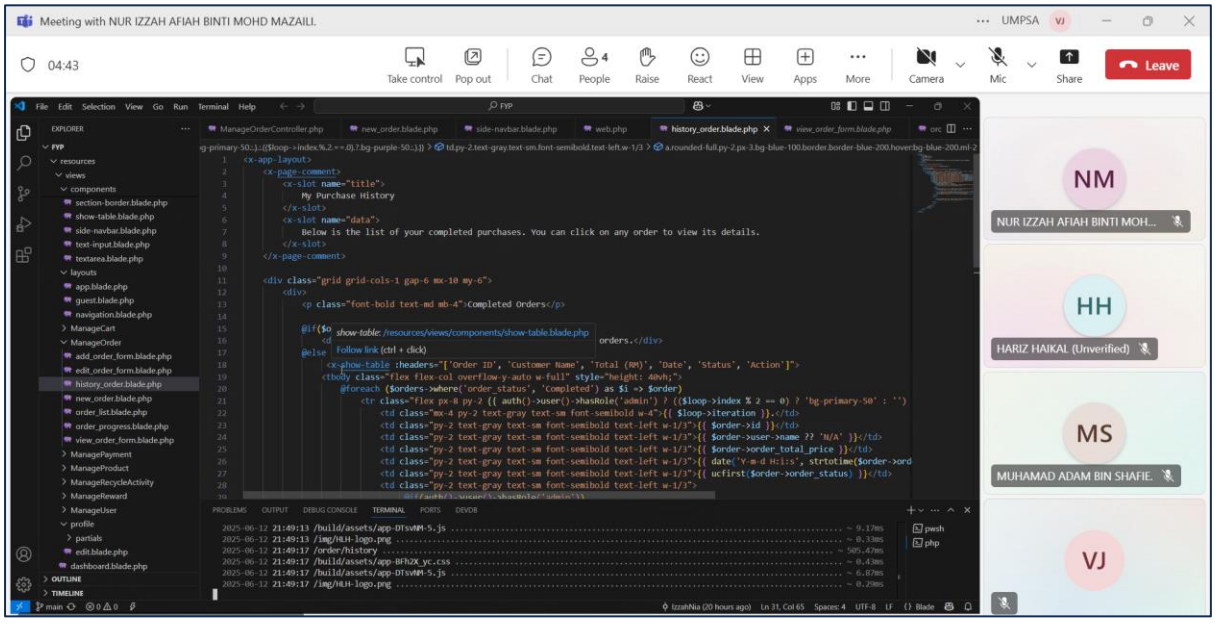
3. The page will display the changes introduced in the selected commit, comparing the old file with the updated file to illustrate modifications.



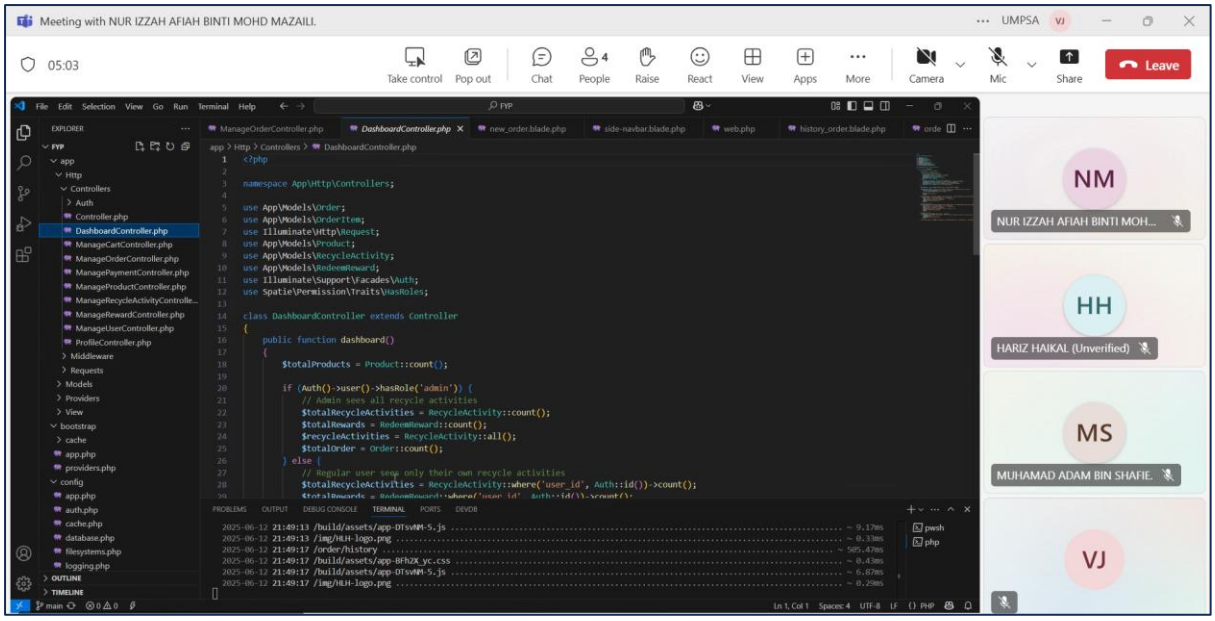
4. Utilize the "Run and Debug" tool available in Visual Studio Code to test the updated system and verify the impact of the changes.



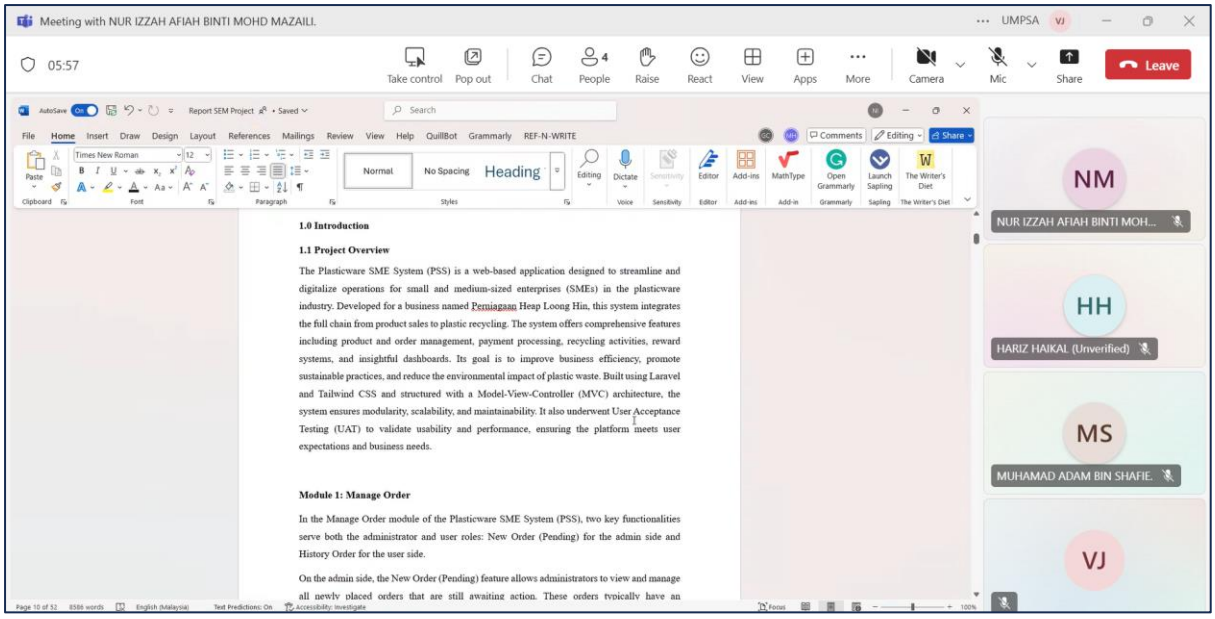
3.3 Minutes of Meeting



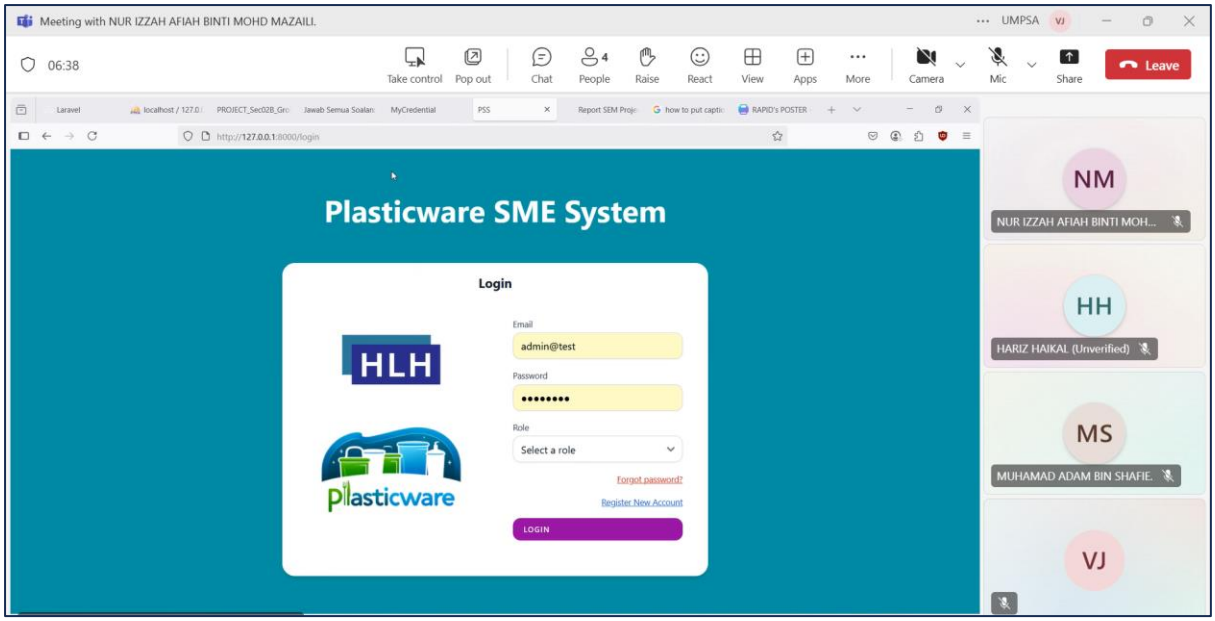
Group Meeting 1	
Date	18/05/2025
Agenda	Review initial codebase and discuss feature enhancement plan
Progress	• Reviewed base Laravel project structure. • Discussed new modules and maintenance features to be added. • Assigned tasks to study existing functionality.
Attendance	All
Absence	None
Problems	Initial difficulty in running the project locally due to missing Node.js and Vite files.

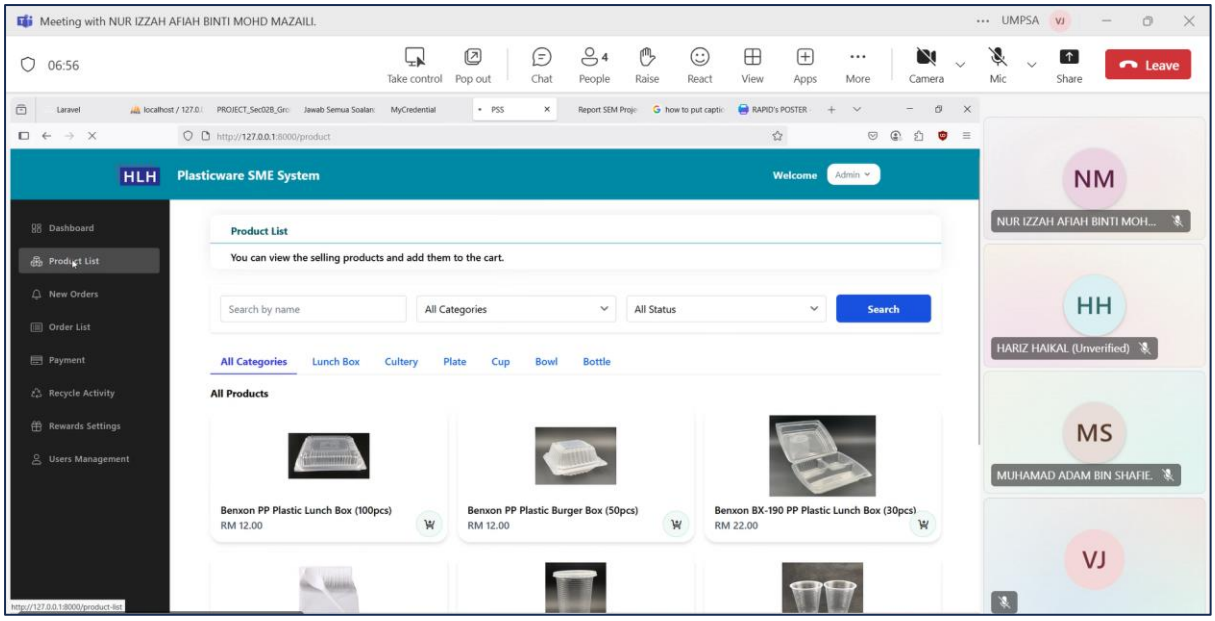


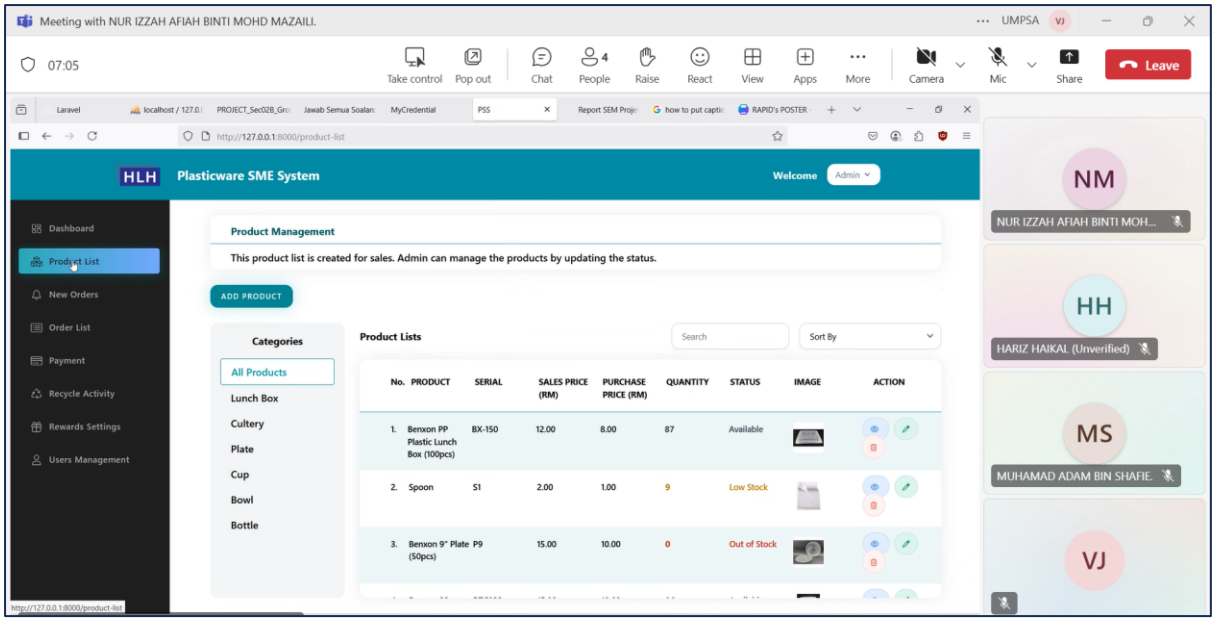
Group Meeting 2	
Date	23/05/2025
Agenda	Review edited code with new features
Progress	• Search functionality implemented in Manage Product. • Filter by category and status added. • Discussed how to log activities into the database. • Product history page was added.
Attendance	All
Absence	None
Problems	• Search button not displaying due to missing Tailwind classes. • Activity logs not capturing user role properly.

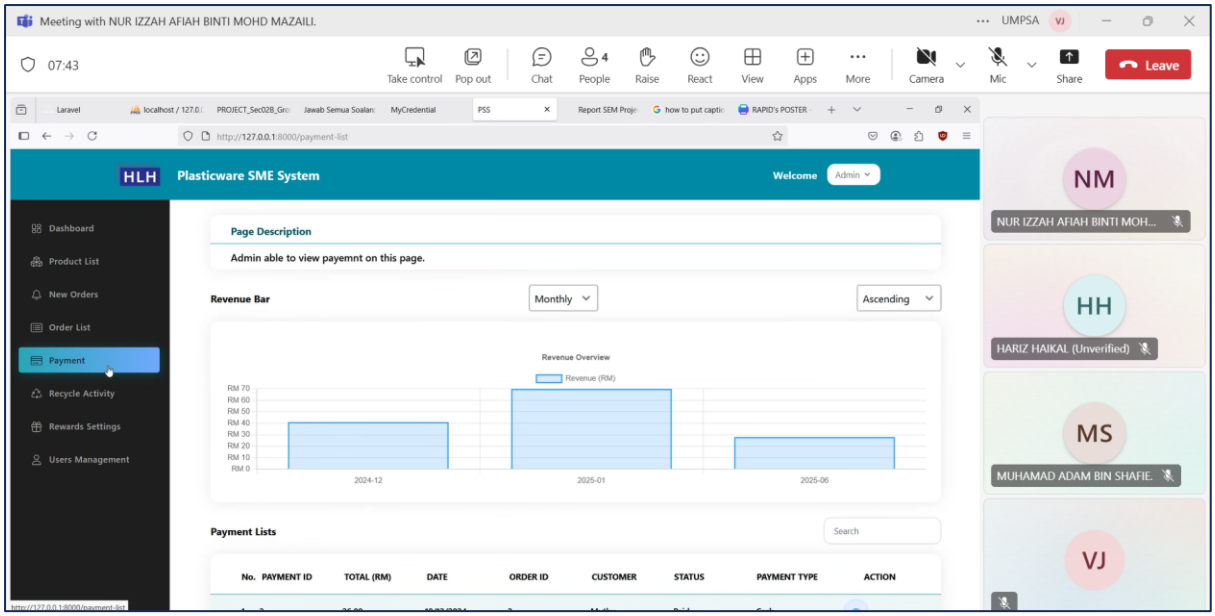


Group Meeting 3	
Date	04/06/2025
Agenda	Task delegation for report
Progress	• Documentation responsibilities assigned. • Finalized modules for maintenance explanation for every module. • Created draft for Change Request Forms.
Attendance	All
Absence	None
Problems	None


Meeting with NUR IZZAH AFIAH BINTI MOHD MAZAILL
UMPSA
06:38
Take control
Pop out
Chat
People
Raise
React
View
Apps
More
Camera
Mic
Share
Leave
http://127.0.0.1:8000/login
Plasticware SME System
Login
HLH
Plasticware
Email
admin@test
Password
Role
Select a role
Forgot password?
Register New Account
LOGIN
NM
NUR IZZAH AFIAH BINTI MOH...
HH
HARIZ HAIKAL (Unverified)
MS
MUHAMAD ADAM BIN SHAFIE.
VJ


Meeting with NUR IZZAH AFIAH BINTI MOHD MAZAILL
UMPSA
06:56
Take control
Pop out
Chat
People
Raise
React
View
Apps
More
Camera
Mic
Share
Leave
http://127.0.0.1:8000/product
HLH
Plasticware SME System
Welcome
Admin
Dashboard
Product List
New Orders
Order List
Payment
Recycle Activity
Rewards Settings
Users Management
Product List
You can view the selling products and add them to the cart.
Search by name
All Categories
All Status
Search
All Categories
Lunch Box
Cultery
Plate
Cup
Bowl
Bottle
All Products
Benxon PP Plastic Lunch Box (100pcs)
RM 12.00
Benxon PP Plastic Burger Box (50pcs)
RM 12.00
Benxon BX-190 PP Plastic Lunch Box (30pcs)
RM 22.00
http://127.0.0.1:8000/product-list
NM
NUR IZZAH AFIAH BINTI MOH...
HH
HARIZ HAIKAL (Unverified)
MS
MUHAMAD ADAM BIN SHAFIE.
VJ





Group Meeting 4	
Date	10/06/2025
Agenda	Demonstration of the system
Progress	• Demonstrated working search functionality and filtering in product module. • Showed recent activity logs in admin dashboard including login and product actions. • Showed revenue bar charts in the payment page with filters by daily, monthly, and yearly views.

Attendance	All
Absence	None
Problems	None

Reference

1. GeeksforGeeks. (2018, October 11). *Software Engineering | Software Maintenance - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-software-maintenance/

2. Singh, R. (2025, January 16). *Learn to Use GitHub Actions: a Step-by-Step Guide*. FreeCodeCamp.org. https://www.freecodecamp.org/news/learn-to-use-github-actions-step-by-step-guide/

3. Romeo, J. (2025, January 23). *How to create a payment gateway and set it up in 4 steps*. Airwallex.com; Airwallex. https://www.airwallex.com/blog/how-to-create-payment-gateway

4. *What is corrective maintenance? | Definition and examples*. (n.d.). Onupkeep. https://upkeep.com/learning/corrective-maintenance/#:~:text=Corrective%20maintenance%20is%20the%20category,with%20breakdown%20or%20reactive%20maintenance.

5. Kirvan, P. (2023, August 29). *change request*. Search CIO. https://www.techtarget.com/searchcio/definition/change-request

6. GeeksforGeeks. (2024, October 15). Perfective Maintenance– Software Engineering. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering/perfective-maintenance-software-engineering/

7. Dave, A. (2024, December 4). *How to set up Google Authentication in Laravel Applications*. freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-set-up-google-auth-in-laravel-apps/