Universiti Malaysia PAHANG

Engineering • Technology • Creativity

PETAKOM Mart	

Group Name

Muhammad Syakir Bin Hazli [CB20125]

Muhammad Ammar bin Mohd Rosli [CB20037]

Ahmad Danial Aqil bin Salihin [CB20049]

Siti Shazwanie Nurain Binti Mohd Shafiq [CB20008]

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This document’s purpose is to provide a complete blueprint of the software requirements for “PETAKOM MART”. In addition, this paper discusses the system's intended audience as well as the hardware, software, and user interface requirements. It will display the goal and comprehensive description of the system's development. Additionally, it is used to lay the groundwork for agreement between clients and software developers outlining the specifics of how the software will function. This document is primarily intended to be given to the appropriate stakeholders for approval and to serve as a guide for the development team to speed up future upgrades and avoid software project failure.

1.2 System Identification

This document uses the following conventions for the PETAKOM Mart Management System (PMMS)

System title	: PETAKOM MART MANAGEMENT SYSTEM
System abbreviation	: PMMS
System identification number	: SDD-MM-PMMS-2022-V1
Use Case identification number	: PMMS-UC-V1
Requirement identification number	: PMMS-REQ-V1
Year	: Years of Development, 2023
Version	: VERSION 1

The format that is used to record PETAKOM Mart Management System (PMMS) is SRS-COM-PMMS-XXXX-VX. MM represents the company's name which is Maguire Market SDN BHD. Then, PMMS stands for PETAKOM Mart Management System where it is also known as the system's name. The "XXXX" is the placeholder to put the year of the system developed and finally the "VX" is to show the version number of the system developed. The format for our use case identification number is PMS-UC-XX. "UC " is referred to as a use case diagram and "XX" is the number of the use case. Finally, the requirement identification number is PMMS-REQ-XX where "REQ" stands for requirement and "XX" is the number of the requirements.

1.3 System Overview/ Module Overview

PETAKOM Mart Management System (PMMS) is a web-based system that is developed to manage all the activity or any documentation that is related to Petakom Society in PETAKOM Mart in a systematic way. The users of this system are the FK Students and FK Lecturers that consist of students, lecturers, coordinator, HOSD, Dean and PETAKOM Committee. There are four modules included in this system to ease the management which are Manage Inventory, Manage Vendor, Generate Sale Reports and Manage Schedule.

For the **first module which is the Manage Inventory**, the module is mainly to be used by the Admin and PETAKOM Committee. In this module, the Admin and PETAKOM Committee is adding, updating and removing the inventory levels in the PMMS based on the inventory levels. **For the second module, Manage vendor** which the module user is Admin where the sole user of the module to determine the vendor that allowed to start or open kiosk inside PETAKOM Mart. Admin responsible to add new vendors, update vendors, remove vendors and view selected vendors. All vendor related task control by admin with manage vendor module Next **for the third module which is manage sales**, the module is for PETAKOM mart define daily, weekly,monthly and yearly sales of mart. It is important to identify every sale to know the stability of the financial market. Besides, the managed sales module also can perform the report generated by administrators.The report of mart is important because the report can be an important evidence for PETAKOM mart. **Lastly, the Manage Schedule module**, which allows the Cashier as the person in charge to register their working hour simply by clicking on the time slot. The cashier is able to cancel their option by clicking on the option button. Then, the registered time for the cashier can be viewed by the PETAKOM Committee.

Therefore, the existence of this PETAKOM Mart Management System (PMMS) will aid the PETAKOM Committee in the management of all data and activities corresponding to society. It will make managing the data and activities easier and improve the system's organization, dependability, and effectiveness.

2. REFERENCED DOCUMENT

2.1 Referenced Document

1. *What is vendor management? | Definition & Process*. (n.d.). Taulia. https://taulia.com/glossary/what-is-vendor-management/#:~:text=Vendor%20management%20is%20a%20term

2. *PETAKOM*. (n.d.). Faculty of Computing. Retrieved April 8, 2023, from https://fk.ump.edu.my/index.php/ms/others/staff-student/petakom

3. Custom Software Requirements Specification Document - Belitsoft. (n.d.). Retrieved November 27, 2021, from https://belitsoft.com/custom-application-development-services/software-requirements-specification-document-example-international-standard

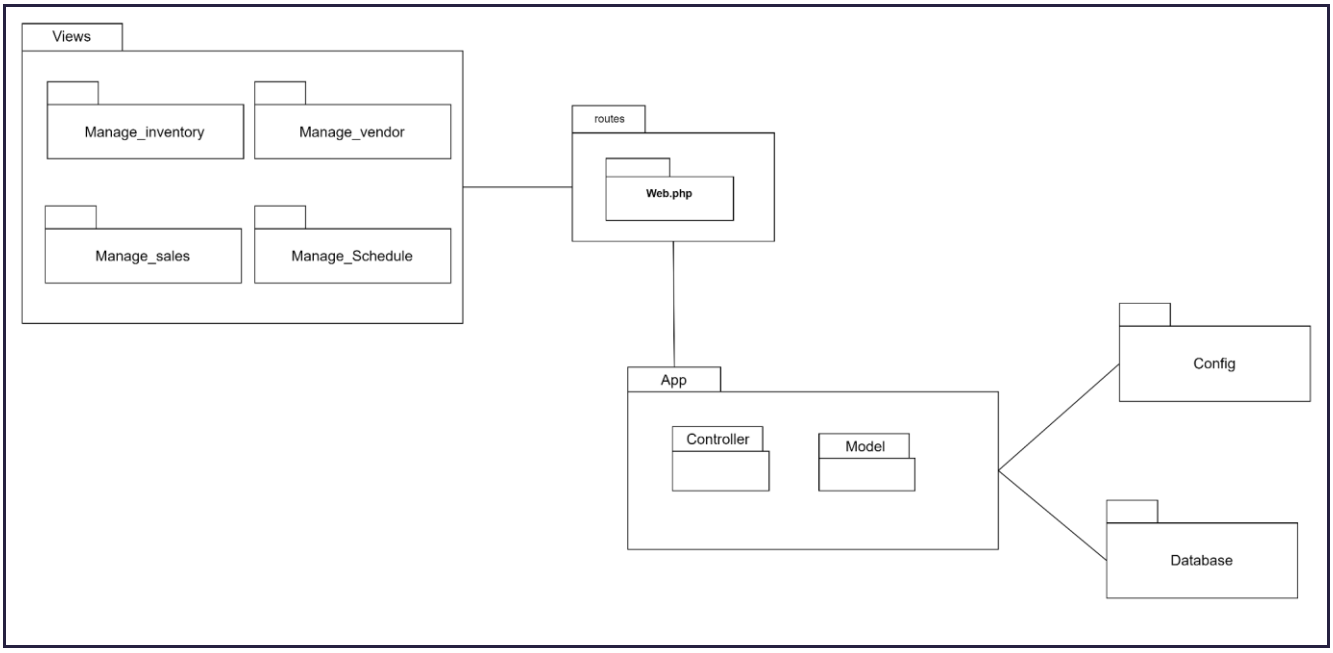
4. Ohnishi, A. (1996, April). Software requirements specification database based on requirements frame model. In *Proceedings of the Second International Conference on Requirements Engineering* (pp. 221-228). IEEE, from Software requirements specification database based on requirements frame model | IEEE Conference Publication | IEEE Xplore

5. *How to Create Software Design Documents. (2020b, November 18). Lucidchart. https://www.lucidchart.com/blog/how-to-create-software-design-documents*

6. Maguire Market Sdn Bhd. (2023, March 19). *Software Requirement Specifications (SRS) - Petakom Mart*. Google Docs. Retrieved May 11, 2023, from https://docs.google.com/document/d/1_ESCQNg5mqx1BN-fG0x4ytjKkNhxybnpBW6kOjC9WrE

3. ARCHITECTURE DESCRIPTION

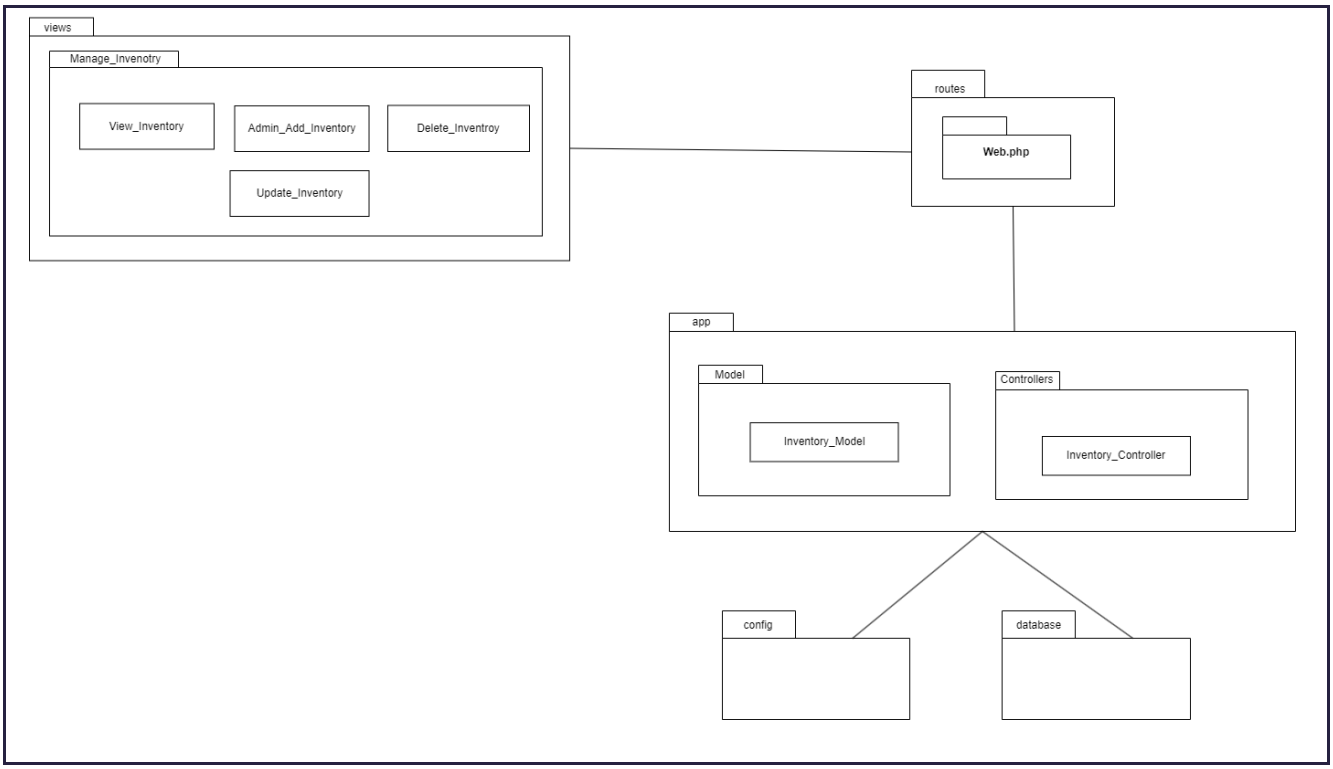
3.1 General Architecture



3.2 Package Module

3.2.1 Manage Inventory (SDD-REQ-100)

Muhammad Ammar bin Mohd Rosli (CB20037)



3.2.1.1 Manage_Inventory Package [SDD-REQ-101]

Class Name	Description
View_Inventory	The interface for the Admin, PETAKOM Committee and Cashier to view products in the inventory. It will allow the Admin, PETAKOM Committee and Cashier to view the product details in the inventory for the PETAKOM Mart.
Admin_Add_Inventory	The interface for the Admin to add products into inventory. It will allow the Admin to add new products that come into the PETAKOM Mart.
Update_Inventory	The interface for the Admin, PETAKOM Committee and Cashier to update products in the inventory. It will allow the Admin, PETAKOM Committee and Cashier to update the low level products in the inventory for the PETAKOM Mart.
Delete_Inventory	The interface for the Admin, PETAKOM Committee and Cashier to delete products in the inventory. It will allow the Admin, PETAKOM Committee and Cashier to delete the discontinued products in the inventory for the PETAKOM Mart.

3.1.1.2 Controller [SDD-REQ-102]

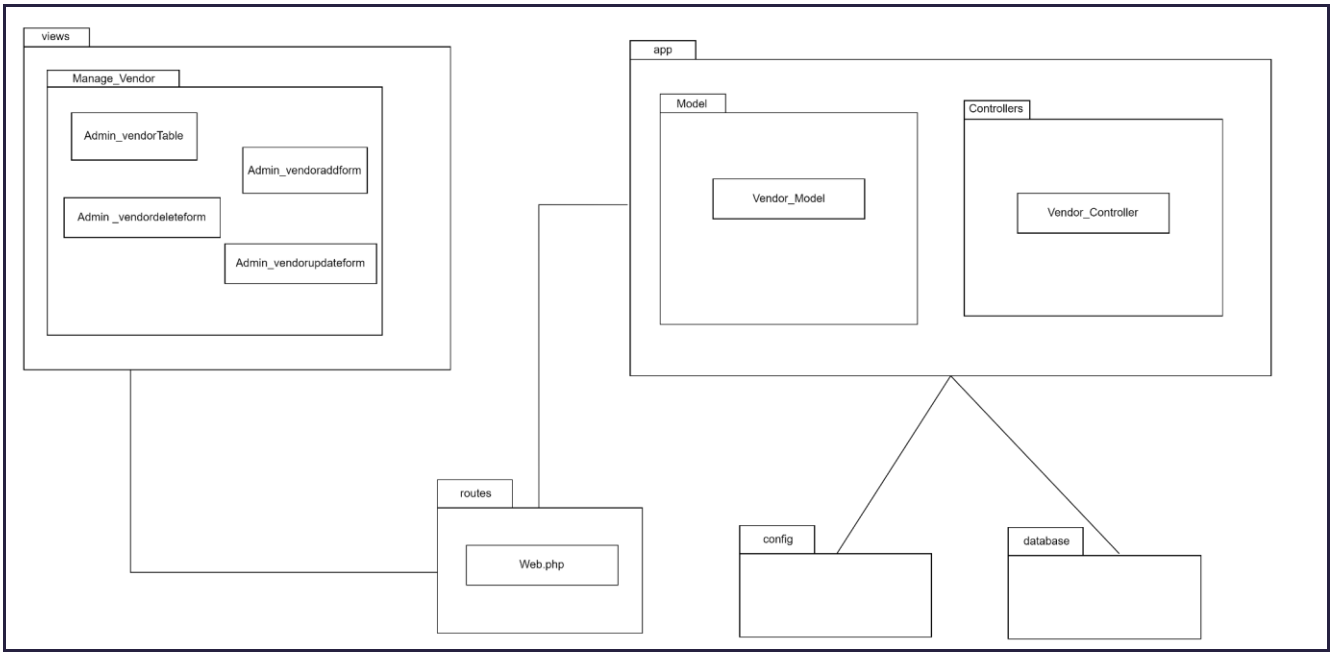
Class Name	Description
Inventory_Controller	This controller is to manage the functions of Users for managing the inventory

3.2.1.3 Model [SDD-REQ-103]

Class Name	Description
Inventory_Model	This model is to retrieve and send data of the products in the inventory.

3.2.2 Manage Vendor (SDD-REQ-200)

AHMAD DANIAL AQIL BIN SALIHIN (CB20049)



3.2.2.1 Manage Vendor

Class Name	Description
Admin_vendorTable	Allow Admin to view list of vendor
Admin_vendoraddform	Allow Admin to input new vendor information
Admin _vendordeleteform	Allow Admin to delete selected vendor information
Admin_vendorupdateform	Allow Admin to update selected vendor information

3.2.2.2 Controller

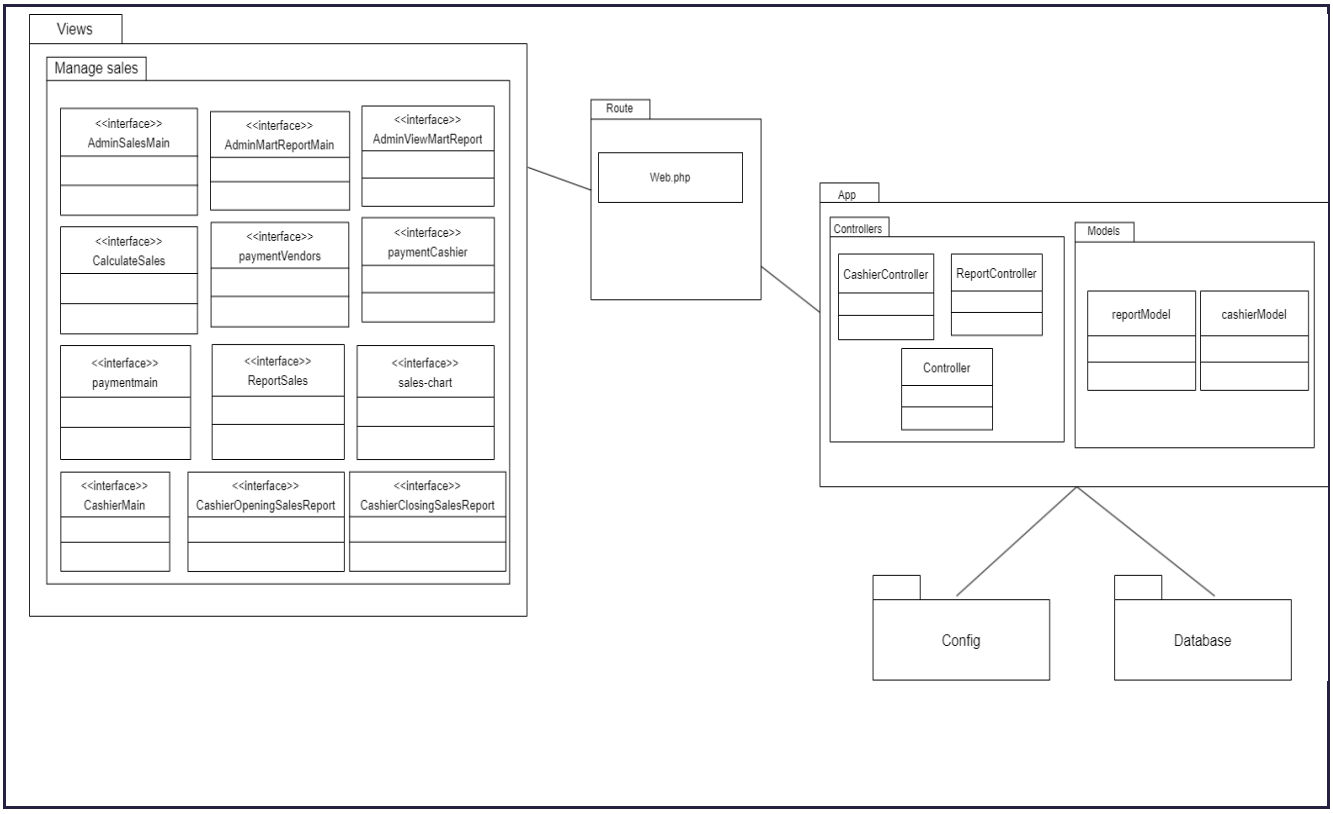
Class Name	Description
Vendor_Controller	Act as middleman which is method and function between Vendor_Model and Manage_Vendor view components

3.2.2.3 Model

Class Name	Description
Vendor_Model	Represent database tables,field, relationship, query, retrieve and update data of Vendor

3.2.3 Manage Sales (SDD-REQ-300)

MUHAMMAD SYAKIR BIN HAZLI (CB20125)



3.2.3.1 Manage sales interface [SDD-REQ-300]

Class Name	Description
AdminSalesMain	This interface allows the admin to enter the manage sales of PETAKOM Mart page.
CalculateSales	This interface allows admin to edit the mart sales for PETAKOM Mart.
sales-chart	This interface allows the admin to view the sales result for PETAKOM Mart everyday.
AdminMartReportMain	This interface allows the admin to enter the PETAKOM report mainpage.
AdminViewMartReport	This interface allows the admin to view/ generate reports of PETAKOM Mart.
paymentVendors	This interface allows the admin to make payment for vendors.
paymentCashier	This interface allows the admin to make a payment for the cashier.
CashierMain	This interface allows the cashier to enter the Sales main page of PETAKOM Mart.

CashierOpeningSalesReport	This interface allows the cashier to fill the opening sales report of PETAKOM Mart.
CashierClosingSalesReport	This interface allows the cashier to add/fill closing sales forms of PETAKOM Mart.

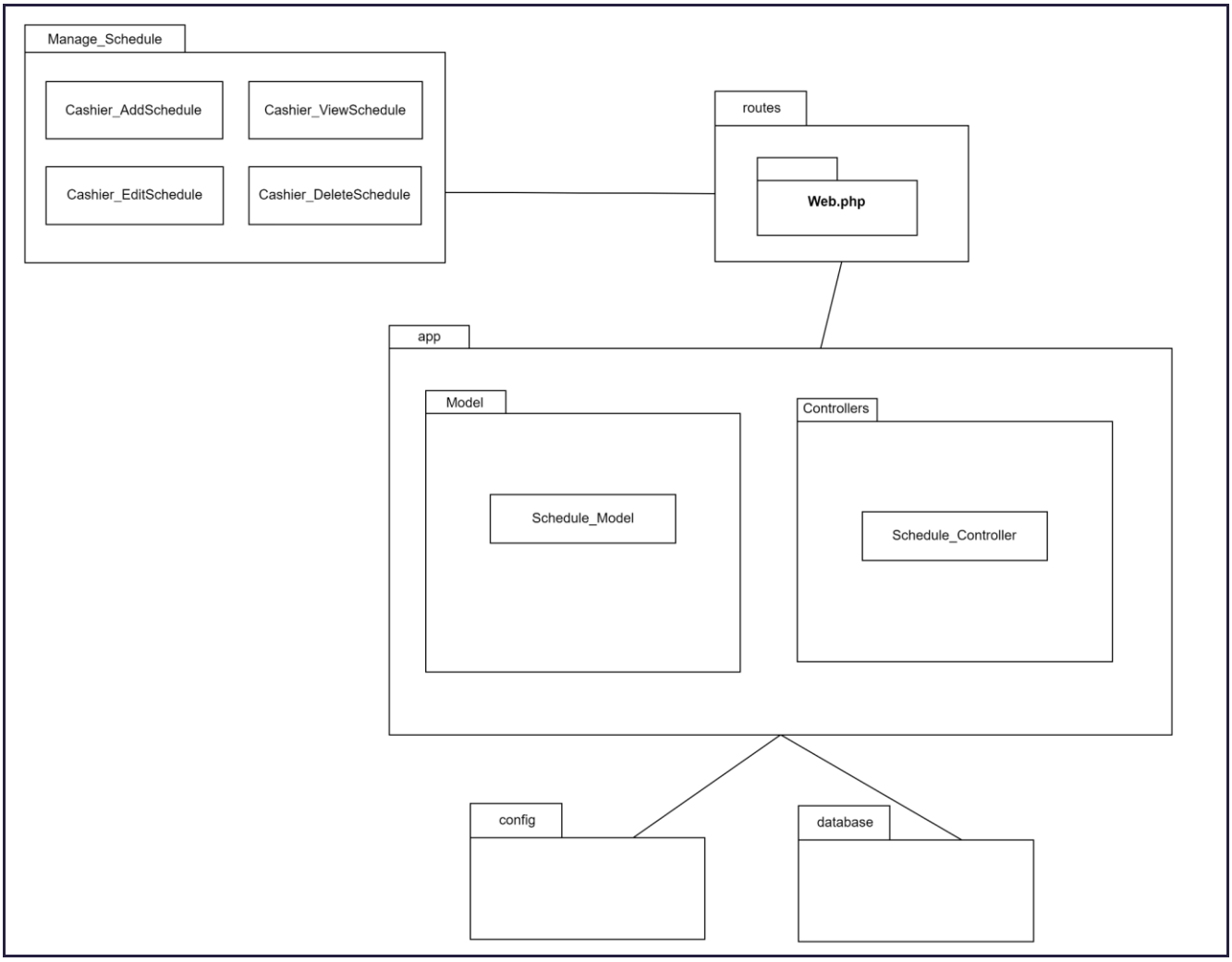
3.2.3.2 Controller [SDD-REQ-400]

Class Name	Description
Admin_Sales_Controller	This controller is to manage the functions of Admin for managing the sales.
Cashier_Sales_Controller	This controller is to manage the functions of Cashier for managing the sales.

3.2.3.3 Model [SDD-REQ-400]

Class Name	Description
reportModel	This model is to retrieve and send data of the sales report in the.
cashierModel	This model is to retrieve and send data of opening and closing sales.

3.2.4 Manage Schedule (SDD-REQ-400)



3.2.4.1 Manage_Schedule Package [SDD-REQ-400]

Class Name	Description
AddSchedule	The interface for the Cashier to Add a new schedule in the system. It will allow the Cashier to add a new schedule for PETAKOM Mart.
DeleteSchedule	The interface for the Cashier to Delete schedules in the system. It will allow the Cashier to delete the schedule for PETAKOM Mart.
ViewSchedule	The interface for the Cashier to View schedules in the system. It will allow the Cashier to view the schedule for PETAKOM Mart.
EditSchedule	The interface for the Cashier to Update schedules in the system. It will allow the Cashier to view the schedule for PETAKOM Mart.

3.2.4.2 Manage_Schedule Controller [SDD-REQ-400]

Class Name	Description
addSchedule_Controller	This controller is to manage the functions of the Cashier for managing the schedule.

3.2.4.3 Manage_Schedule Model [SDD-REQ-400]

Class Name	Description
addSchedule_Model	This model is to fetch and send data of the cashier for Manage Schedule.

3.3 Data Dictionary

3.3.1 Manage Inventory

Muhammad Ammar bin Mohd Rosli

Field Name	Description	Data Type	Constraint
product_id	The unique number of the product	VARCHAR (20)	PK
product_name	The name of product	VARCHAR (20)	
company_name	The company name of the product	VARCHAR (20)	
quantity	The quantity left of the product	INT	
Category	The category of the product	VARCHAR (20)	
Price	The price of the product	BOOL	
Status	The status available or unavailable of product	VARCHAR (20)	
Oder_Date	The latest order date of the product	DATE	

3.3.2 Manage Vendor

Ahmad Danial Aqil Bin Salihin

Field Name	Description	Data Type	Constraint
v_id	The vendor unique id	int	PK
v_name	The company name of the product	VARCHAR (30)	
owner_name	The owner name of the company	VARCHAR (30)	
v_contact	The contact number of the company	VARCHAR (20)	
v_type	Product type	VARCHAR (20)	
v_quantity	Total quantity	INT	
v_date	The date company info added	DATE	

3.3.3 Manage Sales

Muhammad Syakir Bin Hazli

Field Name	Description	Data Type	Constraint
Admin_id	The unique id for admin.	VARCHAR(15)	PK
Admin_name	The name of admin	VARCHAR(30)	
Cashier_ID	The unique id of the cashier	VARCHAR (15)	
Cashier_password	The unique password for cashier	VARCHAR (15)	
opening_sales_id	The unique id for opening sales.	VARCHAR (15)	
closing_sales_id	The unique id for closing sales.	VARCHAR (15)	
product_id	The unique id for each product of PETAKOM mart.	VARCHAR (20)	
product_name	The name of each product.	VARCHAR (20)	
company_name	The company name of the product	VARCHAR (20)	
quantity_in	The quantity left of the product	INT	
quantity_out	The quantity left of the product	INT	
Category	The category of the product	VARCHAR (20)	
Price	The price of the product	BOOL	
Status	The status available or unavailable of product	VARCHAR (20)	
Order_Date	The latest order date of the product	DATE	
report_id	The unique id for each report.	VARCHAR (15)	
vendor_id	The unique id for the vendor.	VARCHAR (15)	
v_name	The company name of the product	VARCHAR (30)	
vendor_pricePay	The price to pay for vendors.	BOOL	
Cashier_Name	The name of the cashier	VARCHAR (30)	
cashier_salary	The salary for each cashier	BOOL	

3.3.4 Manage Schedule

Siti Shazwanie Nurain Binti Mohd Shafiq

Field Name	Description	Data Type	Constraint
Schedule _ID	The schedule ID	VARCHAR(20)	PK
Cashier_ID	The cashier ID	VARCHAR(15)	
Cashier_Name	The cashier name	VARCHAR(30)	
Schedule_Date	The schedule date	DATE	
Schedule_Time	The schedule time	TIME	
Work_Hours	The cashier and PETAKOM Committee working hours	INT	
Timeslot	The schedule time slot	BOOL	

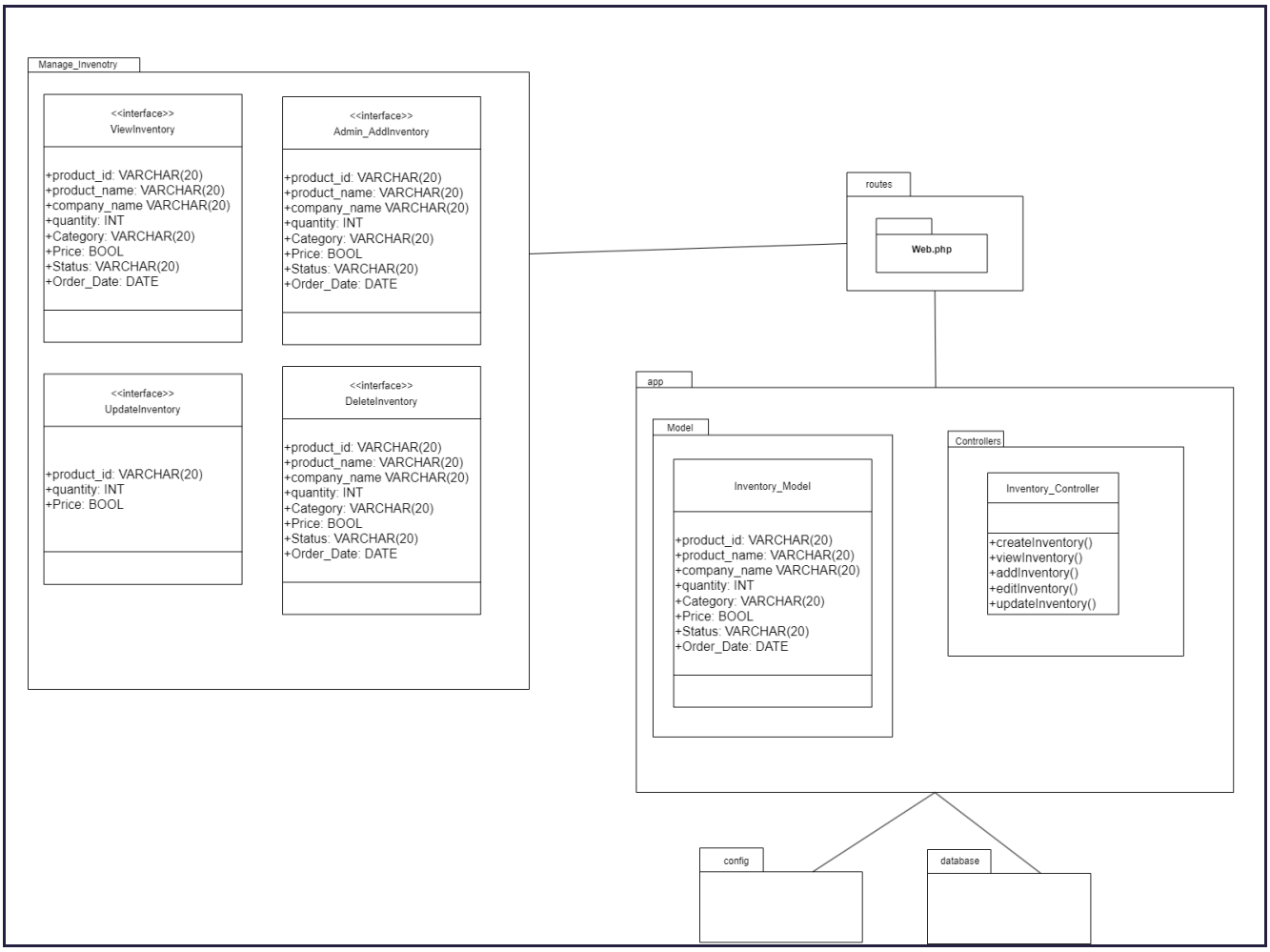
User

Field Name	Description	Data Type	Constraint
User_ID	The cashier ID	VARCHAR(15)	PK
Cashier_ID	The cashier name	VARCHAR(30)	FK
Admin_ID	The schedule date	DATE	FK
Cashier_Name			
Admin_Name			
V_ID			
Vendor_Name			
Password			

4. DETAILED DESIGN

4.1 Package 1 [SDD-REQ-100]

Muhammad Ammar bin Mohd Rosli (CB20037)



4.1.1 ViewInventory.blade.php [SDD-REQ-101]

Class Type	Boundary class	
Responsibility	An interface used by the cashier, PETAKOM Committee and Admin to view the inventory details.	
Attributes	Attributes Name	Attributes Type
	product_id product_name company_name quantity Category Price Status Order_Date Back view product details	VARCHAR (20) VARCHAR (20) VARCHAR (20) INT VARCHAR (20) BOOL VARCHAR (20) DATE Button Button
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	BEGIN System display all the activities details in the table DISPLAY product_id DISPLAY product_name DISPLAY company_name DISPLAY quantity DISPLAY Category DISPLAY Price DISPLAY Status DISPLAY Order_Date	

	IF Admin, PETAKOM Committee OR Cashier click on “Back” button
	THEN the page direct to the homepage
	ELSE
	Remain at the ViewInventory.blade.php page
	END IF
	
	The Admin, PETAKOM Committee OR Admin click on “View product details” button
	
	IF the inventory level is low
	THEN system pop-up low inventory level message
	ELSE
	The system direct to display the selected product details
	END IF
	END

4.1.2 Admin_AddInventory.blade.php [SDD-REQ-102]

<table>
<tr><td>Class Type</td><td colspan="2">Boundary Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">An interface used by the Admin to add new product into the inventory</td></tr>
<tr><td rowspan="9">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>product_id</td><td>VARCHAR (20)</td></tr>
<tr><td>product_name</td><td>VARCHAR (20)</td></tr>
<tr><td>company_name</td><td>VARCHAR (20)</td></tr>
<tr><td>quantity</td><td>INT</td></tr>
<tr><td>Category</td><td>VARCHAR (20)</td></tr>
<tr><td>Price</td><td>BOOL</td></tr>
<tr><td>Order_Date</td><td>DATE</td></tr>
<tr><td>Back</td><td>Button</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">BEGIN
The Admin click on “Add Product” button
System display all the activities details in the form
DISPLAY product_id
DISPLAY product_name
DISPLAY company_name
DISPLAY quantity
DISPLAY Category
DISPLAY Price

The Admin enter the product_id, product_name, company_name, quantity, Category and Price in the form

The Admin click on “Add” button</td></tr>
</table>

	IF data successfully enter in the database
	THEN
	System saves new added product in the database
	System pop-up successful message
	System direct to ViewInventory.blade.php
	ELSE
	System pop-up error message
	The Admin remains at the AddInventory.blade.php page
	END IF
	END

4.1.3 EditInventory.blade.php [SDD-REQ-103]

<table>
<tr><td>Class Type</td><td colspan="2">Boundary Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">An interface used by the Cashier, PETAKOM Committee and the Admin to update low level inventory.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>product_id
quantity
Price</td><td>VARCHAR (20)
INT
DATE</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">BEGIN
The Users click on “Update Product” button
System display all the activities details in the form
DISPLAY product_id
DISPLAY quantity
DISPLAY Price

Users update product_id
Users update quantity
Users update Price

The Users click on “Update” button
IF data successfully enter in the database
THEN
System saves new added product in the database
System pop-up successful message
System direct to ViewInventory.blade.php
ELSE
System pop-up error message
Users remains at the EditInventory.blade.php page
END IF
END</td></tr>
</table>

4.1.4 DeleteInventory.blade.php [SDD-REQ-104]

<table>
<tr><td>Class Type</td><td colspan="2">Boundary Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">An interface used by the Cashier, PETAKOM Committee and the Admin to delete products from the inventory.</td></tr>
<tr><td rowspan="8">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>product_id</td><td>VARCHAR (20)</td></tr>
<tr><td>product_name</td><td>VARCHAR (20)</td></tr>
<tr><td>company_name</td><td>VARCHAR (20)</td></tr>
<tr><td>quantity</td><td>INT</td></tr>
<tr><td>Category</td><td>VARCHAR (20)</td></tr>
<tr><td>Price</td><td>BOOL</td></tr>
<tr><td>Order_Date</td><td>DATE</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">BEGIN
System display all the activities details in the table
DISPLAY product_id
DISPLAY product_name
DISPLAY company_name
DISPLAY quantity
DISPLAY Category
DISPLAY Price
DISPLAY Status
DISPLAY Order_Date

IF Users click on “Delete” button on any proposal details
THEN

The selected product details will be deleted on click</td></tr>
</table>

	System delete the selected existing product details from the database and the product details table will be updated
	System displays a new updated product details table in the inventory.
	
	ELSE IF Users click on the “Back” button
	THEN the page direct to the homepage
	
	ELSE
	Remains at the DeleteInventory.blade.php page
	
	END IF
	
	END

4.1.5 InventoryModel.php [SDD-REQ-105]

<table>
<tr><td>Class Type</td><td colspan="2">Model Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">To store, retrieve and manage database of Inventory</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>product_id
product_name
company_name
quantity
Category
Price
Order_Date</td><td>VARCHAR (20)
VARCHAR (20)
VARCHAR (20)
INT
VARCHAR (20)
BOOL
DATE</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">View Inventory
BEGIN

public function index(){
$data_inventory = \App\Models\Inventory::all();
return view('index',['data_inventory'=> $data_invenotry]);
}

END</td></tr>
<tr><td></td><td colspan="2">Add Inventory
BEGIN

public function create(Request $request){
\App\Models\Inventory::create($request->all());
return redirect('/datainventory')->with('success','New product has been added!');
}

END</td></tr>
<tr><td></td><td colspan="2">Update Inventory
BEGIN

public function update(Request $request, $product_id){
$data_inventory = \App\Models\Inventory::find($product_id);
$data_inventory->update($request->all());</td></tr>
</table>

```
    return redirect('/datainventory')->with('success','Product is
Updated');
  }

END
```

Delete Inventory

BEGIN

```
 public function delete($product_id){
    $data_inventory = \App\Models\Inventory::find($product_id);
    $data_inventory->delete($data_inventory);
    return redirect('/datainventory')->with('success','Product has been
deleted');
  }

END
```

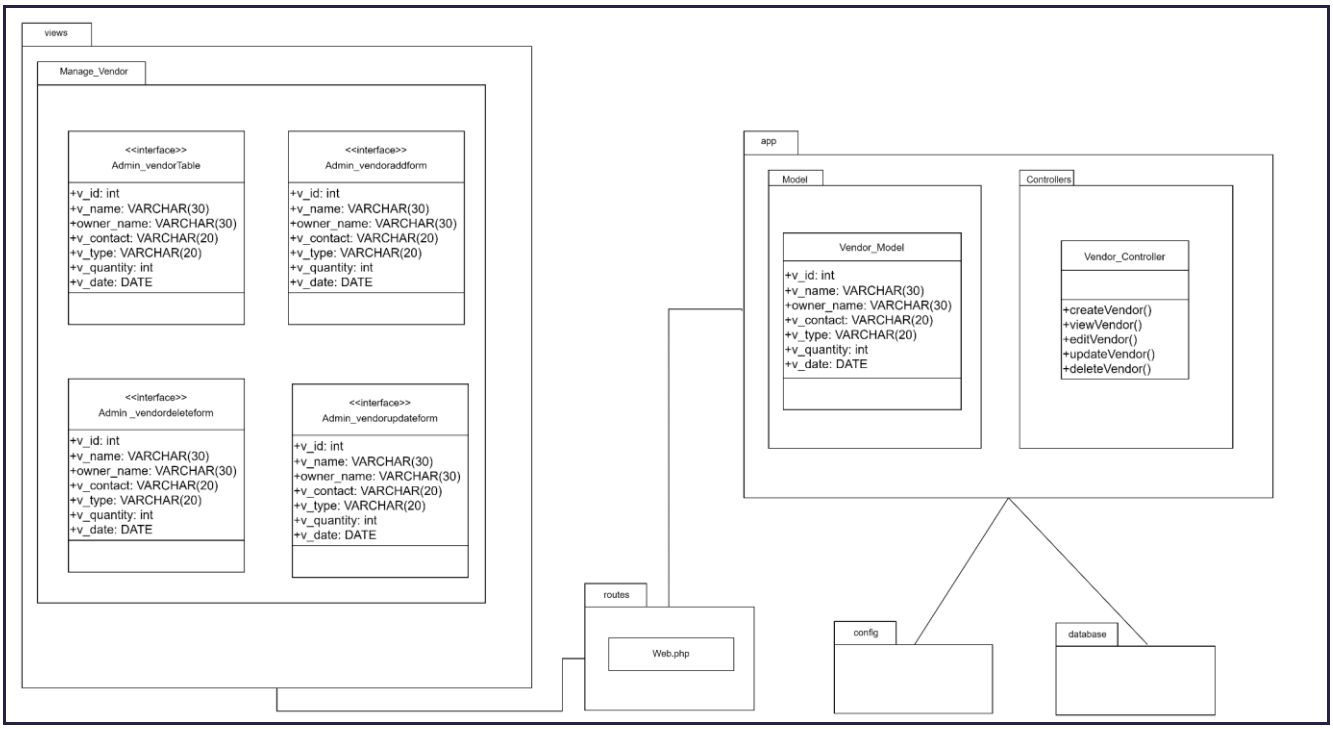
4.1.6 InventoryController.php [SDD-REQ-106]

<table>
<tr><td>Class Type</td><td colspan="2">Model Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">To store, retrieve and manage database of Inventory</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="6">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>index()</td><td>Method for PETAKOM Committee, cashier and Admin to view inventory level</td></tr>
<tr><td>ViewInventory()</td><td>Method for PETAKOM Committee, cashier and Admin to view the product details from the database</td></tr>
<tr><td>AddInventory()</td><td>Method for Admin to add the product details into the database</td></tr>
<tr><td>UpdateInventory()</td><td>Method for PETAKOM Committee, cashier and Admin to update the existing product details from the database</td></tr>
<tr><td>DeleteInvenotry()</td><td>Method for PETAKOM Committee, cashier and Admin to delete product details from the database</td></tr>
<tr><td>Algorithm</td><td colspan="2">index()
BEGIN
Receive a GET request to show all inventory data.
Retrieve all the necessary inventory data from the database, using Laravel's query ORM like Eloquent.
Return the inventory data to the view, as an array or collection.
Render the view with the inventory data and return it to the user's browser.
END

addVendor()
BEGIN
Receive a GET request to display the form to add a new inventory data.
Display the form view to the user.
Receive a POST request with the data submitted by the user.
Validate the submitted data using Laravel's built-in validation functionality.
If the validation fails, return the user to the form view with error messages.
If the validation succeeds, create a new inventory data in the database using Laravel's query ORM like Eloquent.</td></tr>
</table>

	Redirect the user to the vendor main page for the product details, and show a success message. END **viewVendor()** BEGIN Receive a GET request to show specific product details. Retrieve the necessary inventory data from the database, using Laravel's query ORM like Eloquent, based on the product's ID Return the inventory data to the view, as an array or object. Render the view with the inventory data and return it to the user's browser. END **updateVendor()** BEGIN Receive a POST request to submit the edited inventory data. Validate the user input, using Laravel's built-in validation tools Update the vendor data in the database, based on the user's input. Redirect the user to the inventory main page for the report, and show a success message. END **DeleteVendor()** BEGIN Receive a POST request to delete a specific inventory data. Retrieve the necessary inventory data from the database, using Laravel's query ORM like Eloquent, based on the product's ID Delete the inventory data from the database. Redirect the user to the inventory main page, and show a success message. END

4.2 Manage Vendor [SDD-REQ-200]



\

4.2.1 Admin_vendortable [SDD-REQ-201]

Class Type	Boundary Class	
Responsibility	An interface used by the admin to view the list of vendor	
Attributes	Attributes Name	Attributes Type
	v_id v_name owner_name v_contact v_type v_quantity v_date	int VARCHAR(30) VARCHAR(30) VARCHAR(20) VARCHAR(20) int DATE
Methods	Method Name	Description
	viewVendor( )	Display list of vendor information

Algorithm	BEGIN
	System display all the activities details in the table
	Display v_id
	Display v_name
	Display owner_name
	Display v_contact
	Display v_type
	Display v_quantity
	Display v_date
	IF Admin click on “edit” button
	THEN System display the Admin_vendorupdateform page
	ELSE IF Admin click on “add” button
	THEN the page display the Admin_vendoraddform
	ELSE IF Admin click on “delete” button
	THEN the page display the Admin_vendordeleteform
	ELSE
	Remain at the Admin_vendortable page
	END IF
	END	

4.2.2 Admin_vendoraddform [SDD-REQ-202]

Class Type	Boundary Class	
Responsibility	An interface used by the admin to add new the list of vendor	
Attributes	Attributes Name	Attributes Type
	v_id v_name owner_name v_contact v_type v_quantity v_date	int VARCHAR(30) VARCHAR(30) VARCHAR(20) VARCHAR(20) int DATE
Methods	Method Name	Description
	createVendor()	Add new list of vendor information

Algorithm	BEGIN
	System display all the activities details in the table
	System display input form
	The admin input vendor information the v_name, owner_name, v_contact, v_type, v_quantity and v_date
	Admin click “confirm” button
	IF data successfully enter in the database
	THEN
	System saves new added vendor information into database
	System display successful message
	System direct to Admin_vendortable
	ELSE
	System pop-up error message
	END IF
	END	

4.2.3 Admin_vendordeleteform [SDD-REQ-203]

Class Type	Boundary Class	
Responsibility	An interface used by the admin to remove vendor information	
Attributes	Attributes Name	Attributes Type
	v_id v_name owner_name v_contact v_type v_quantity v_date	int VARCHAR(30) VARCHAR(30) VARCHAR(20) VARCHAR(20) int DATE
Methods	Method Name	Description
	deleteVendor	Remove existing vendor information

Algorithm	BEGIN
	System display all the activities details in the table
	System display delete form
	Admin click “delete” button
	IF data successfully enter in the database
	THEN
	System update the vendor information
	System display success delete message
	System direct to Admin_vendortable
	ELSE
	System pop-up error message
	END IF
	END	

4.2.4 Admin_vendorupdateform [SDD-REQ-204]

Class Type	Boundary Class	
Responsibility	An interface used by the admin to update the vendor information	
Attributes	Attributes Name	Attributes Type
	v_id v_name owner_name v_contact v_type v_quantity v_date	int VARCHAR(30) VARCHAR(30) VARCHAR(20) VARCHAR(20) int DATE
Methods	Method Name	Description
	editVendor() updateVendor()	Display vendor information to edit Update vendor information

Algorithm	BEGIN
	System display all the activities details in the table
	System display input form
	The admin update vendor information the v_name, owner_name, v_contact, v_type, v_quantity and v_date
	Admin click “update” button
	IF data successfully enter in the database
	THEN
	System saves new updated vendor information into database
	System display successful message
	System direct to Admin_vendortable
	ELSE
	System pop-up error message
	END IF
	END	

4.2.5 Vendor_Model [SDD-REQ-205]

<table>
<tr><td>Class Type</td><td colspan="2">Model Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">To store, retrieve and manage database of Inventory</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>v_id
v_name
owner_name
v_contact
v_type
v_quantity
v_date</td><td>int
VARCHAR(30)
VARCHAR(30)
VARCHAR(20)
VARCHAR(20)
int
DATE</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">View Vendor
BEGIN

public function viewVendor(){
$data_inventory = \App\Models\Vendor_Model::all();
return view('Admin_vendorTable',['data_vendor'=> $data_vendor]);
}
END</td></tr>
<tr><td></td><td colspan="2">Add Vendor
BEGIN

public function createVendor(Request $request){
\App\Models\Vendor_Model::create($request->all());
return redirect('/vendorpage')->with('success','New vendor has been added!');
}
END</td></tr>
<tr><td></td><td colspan="2">Update Vendor
BEGIN

public function updateVendor(Request $request, $v_id){
$data_vendor = \App\Models\Vendor_Model::find($v_id);
$data_vendor->updateVendor($request->all());
return redirect('/vendorpage')->with('success','Vendor is Updated');</td></tr>
</table>

	} END

	Delete Vendor
	BEGIN
	
	public function delete($v_id){
	$data_vendor = \App\Models\Vendor_Model::find($v_id);
	$data_vendor->deleteVendor($data_vendor);
	return redirect('/vendorpage')->with('success','Vendor has been deleted');
	}
	END

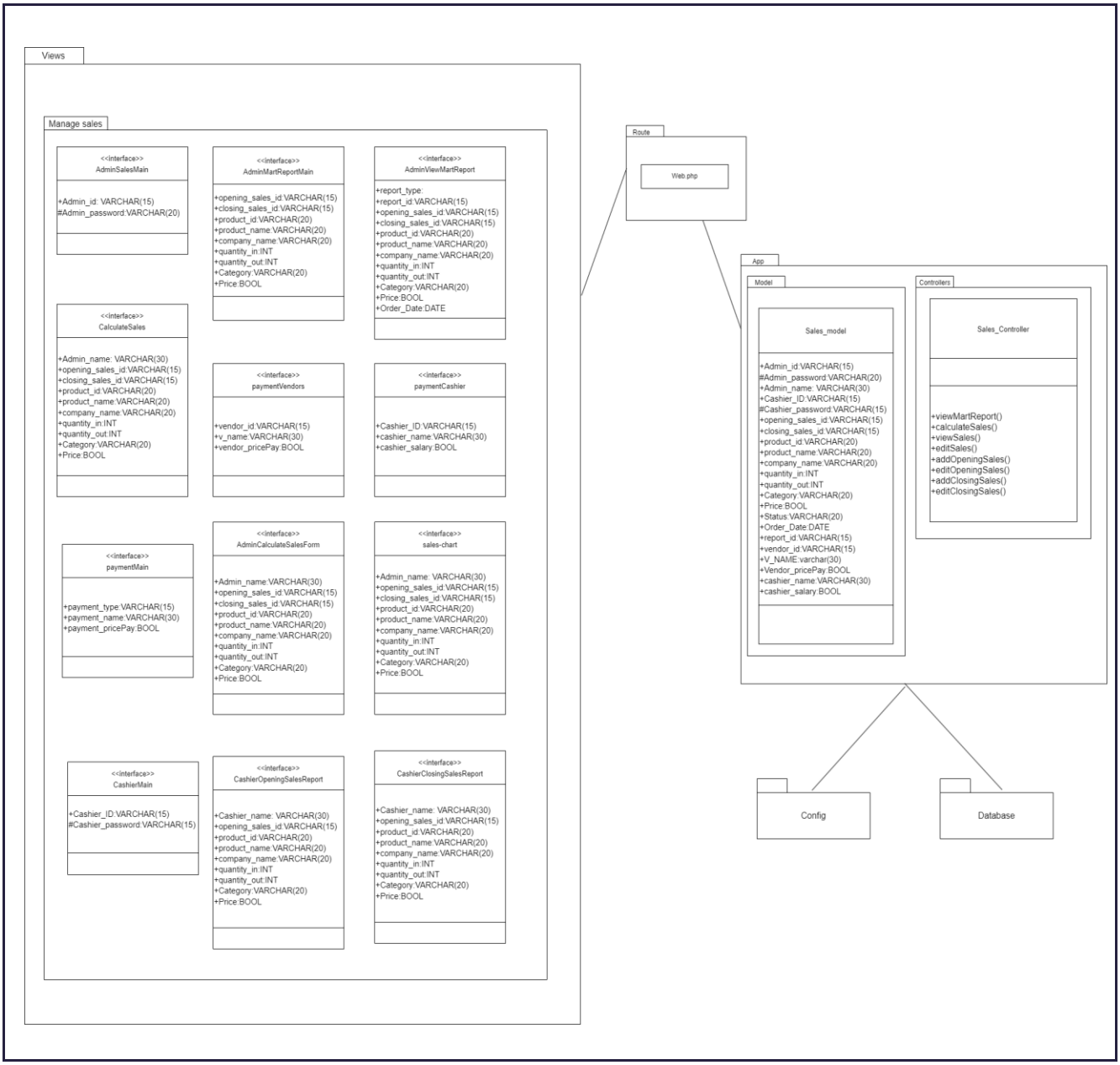
4.2.6 Vendor_Controller [SDD-REQ-206]

Class Type	Model Class	
Responsibility	To store, retrieve and manage database of Inventory	
Attributes	Attributes Name	Attributes Type
	Not applicable	Not applicable
Methods	Method Name	Description
	viewVendor()	Display list of vendor information
	createVendor()	Add new list of vendor information
	deleteVendor()	Remove existing vendor information
	editVendor()	Display vendor information to edit
	updateVendor()	Update vendor information
Algorithm	**viewVendor()** BEGIN Receive a GET request to show all vendor data.	

	Retrieve all the necessary vendor data from the database, using Laravel's query ORM like Eloquent. Return the vendor data to the view, as an array or collection. Render the view with the inventory data and return it to the user's browser. END **createVendor()** BEGIN Receive a GET request to display the form to add a new vendor data. Display the form view to the user. Receive a POST request with the data submitted by the user. Validate the submitted data using Laravel's built-in validation functionality. If the validation fails, return the user to the form view with error messages. If the validation succeeds, create a new inventory data in the database using Laravel's query ORM like Eloquent. Redirect the user to the vendor main page for the product details, and show a success message. END **editVendor()** BEGIN Receive a GET request to show specific vendor details. Retrieve the necessary vendor data from the database, using Laravel's query ORM like Eloquent, based on the vendor ID Return the inventory data to the view, as an array or object. Render the view with the inventory data and return it to the user's browser. END **updateVendor()** BEGIN Receive a POST request to submit the edited vendor data. Validate the user input, using Laravel's built-in validation tools Update the vendor data in the database, based on the user's input. Redirect the user to the vendor main page and show a success message. END **deleteVendor()** BEGIN Receive a POST request to delete a specific vendor data. Retrieve the necessary inventory data from the database, using Laravel's query ORM like Eloquent, based on the vendor ID Delete the vendor data from the database.

	Redirect the user to the vendor main page and show a success message. END

4.3 Manage Sales [SDD-REQ-300]



4.3.1 AdminSalesMain.blade.php [SDD-REQ-301]

Class Type	Boundary class	
Responsibility	An interface used by the admin to enter the admin main page.	
Attributes	Attributes Name	Attributes Type
	Admin_id	VARCHAR(15)
	Admin_password	VARCHAR(20)
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	BEGIN System display login page Admin insert Admin_id and Admin_password Admin click “Login” button System display admin sales mainpage END	

4.3.2 CalculateSales [SDD-REQ-302]

Class Type	Boundary class	
Responsibility	An interface used by admin to calculate the sales report based on opening and closing sales form.	
Attributes	Attributes Name	Attributes Type
	Admin_name	VARCHAR(30)
	opening_sales_id	VARCHAR(15)
	closing_sales_id	VARCHAR(15)
	product_id	VARCHAR(20)
	product_name	VARCHAR(20)
	company_name	VARCHAR(20)
	quatity_in	INT

<table>
<tr><td></td><td>quantity_out
Category
Price</td><td>INT
VARCHAR(20
BOOL</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>calculateSales()</td><td>Method for Admin to calculate the mart sales from the database.</td></tr>
<tr><td>Algorithm</td><td colspan="2">BEGIN
System
Admin enter opening_sales_id
Admin enter closing_sales_id
System display all the activities details in the table
DISPLAY product_id
DISPLAY product_name
DISPLAY company_name
DISPLAY quantity_in
DISPLAY quantity_out
DISPLAY Category
DISPLAY Price
IF Admin click on “Calculate” button
THEN
The sales will be calculated based on calculation of
sales report (opening sales - closing sales)
ELSE
The sales will not be calculated
END IF
END</td></tr>
</table>

4.3.3 AdminMartReportMain.blade.php [SDD-REQ-305]

Class Type	Boundary class	
Responsibility	An interface used by the admin to choose the type of report they want.	
Attributes	Attributes Name	Attributes Type
	report_type report_id	BUTTON VARCHAR(15)
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Begin Admin clicks “Mart Report” button System display Main page for mart report END	

4.3.4 AdminViewMartReport.blade.php [SDD-REQ-306]

Class Type	Boundary class	
Responsibility	An interface used by the admin to view the mart report.	
Attributes	Attributes Name	Attributes Type
	report_type report_id opening_sales_id closing_sales_id product_id product_name company_name quatity_in quantity_out Category Price Order_Date	 VARCHAR VARCHAR(15) VARCHAR(15) VARCHAR(15) VARCHAR(20) VARCHAR(20) VARCHAR(20) INT INT VARCHAR(20) BOOL DATE

Methods	Method Name	Description
	viewMartReport()	Method for Admin to view mart report details from the database.

Algorithm	Begin
	System display selection report type for admin
	IF Admin clicks “Daily report”
	System display daily report of PETAKOM mart
	ELSE IF Admin clicks “Weekly report” button
	System display weekly report of PETAKOM mart
	ELSE IF Admin clicks “Monthly report” button
	System display monthly report of PETAKOM mart
	ELSE
	System will not generated any report
	
	END IF
	END IF
	END IF
	END	

4.3.5 paymentVendors.blade.php [SDD-REQ-307]

Class Type	Boundary class	
Responsibility	An interface used by the admin to make payment for vendors.	
Attributes	Attributes Name	Attributes Type
	vendor_id v_name vendor_pricePay	VARCHAR(15) VARCHAR(15) BOOL
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Begin Admin enter vendor_id System will display all vendors details on the table DISPLAY vendor_id	

	DISPLAY v_name
	DISPLAY vendor_pricePay
	IF Admin clicks “Pay”
	System make payment for selected vendors
	ELSE
	System will not make any payment
	END IF
	END

4.3.6 paymentCashier.blade.php [SDD-REQ-308]

Class Type	Boundary class	
Responsibility	An interface used by the admin to make salary payment for cashiers.	
Attributes	Attributes Name	Attributes Type
	Cashier_ID cashier_name cashier_salary	VARCHAR(15) VARCHAR(30) BOOL
Methods	Method Name	Description
	Not applicable	Not applicable

Algorithm	Begin
	Admin enter Cashier_ID
	System will display all cashier details on the table
	DISPLAY Cashier_ID
	DISPLAY cashier_name
	DISPLAY cashier_salary
	IF Admin clicks “Pay” button
	System will make a salary payment for selected cashier
	ELSE
	System not make a payment
	END IF
	END	

4.3.7 CashierMain.blade.php [SDD-REQ-309]

Class Type	Boundary class

Responsibility	An interface used by the cashier to enter sales form mainpage.	
Attributes	Attributes Name	Attributes Type
	Cashier_ID cashier_name	VARCHAR(15) VARCHAR(30)
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Begin Cashier enter Cashier_ID Cashier enter cashier_name System will display Cashier sales main page END	

4.3.8 CashierOpeningSalesReport.blade.php [SDD-REQ-310]

Class Type	Boundary class	
Responsibility	An interface used by the cashier to add the opening data sales in the form.	
Attributes	Attributes Name	Attributes Type
	Cashier_name opening_sales_id product_id product_name company_name quantity_in quantity_out Category Price	VARCHAR(30) VARCHAR(15) VARCHAR(20) VARCHAR(20) VARCHAR(20) INT INT VARCHAR(20) BOOL
Methods	Method Name	Description
	addOpeningSales()	Method for Cashier to add opening sales data into database.
Algorithm	Begin System will display opening sales form Cashier enter Cashier_name	

	Cashier enter product_id
	Cashier enter product_name
	Cashier enter company_name
	Cashier enter quantity_in
	Cashier enter quantity_out
	Cashier enter Category
	Cashier enter Price
	IF Cashier clicks "Save"
	System will save the opening sales data
	System display successful data saved
	ELSE
	System displays error message data not save
	END IF
	END

4.3.9 CashierClosingSalesReport.blade.php [SDD-REQ-312]

Class Type	Boundary class	
Responsibility	An interface used by the cashier to add the closing data sales in the form.	
Attributes	Attributes Name	Attributes Type
	Cashier_name closing_sales_id product_id product_name company_name quantity_in quantity_out Category Price	VARCHAR(30) VARCHAR(15) VARCHAR(20) VARCHAR(20) VARCHAR(20) INT INT VARCHAR(20) BOOL
Methods	Method Name	Description
	addClosingSales()	Method for Cashier to add closing sales data into database.
Algorithm	Begin	

	System will display closing sales form
	Cashier enter Cashier_name
	Cashier enter product_id
	Cashier enter product_name
	Cashier enter company_name
	Cashier enter quantity_in
	Cashier enter quantity_out
	Cashier enter Category
	Cashier enter Price
	IF Cashier clicks "Save"
	System will save the closing sales data
	System display successful data saved
	ELSE
	System displays error message data not save
	END IF
	END

4.3.10 reportmodel.blade.php [SDD-REQ-314]

Class Type	Model class	
Responsibility	To store, retrieve and manage a database of activities.	
Attributes	Attributes Name	Attributes Type
	Admin_id Admin_name Cashier_name closing_sales_id product_id product_name company_name quantity_in quantity_out Category Price	VARCHAR(15) VARCHAR(30) VARCHAR(30) VARCHAR(15) VARCHAR(20) VARCHAR(20) VARCHAR(20) INT INT VARCHAR(20) BOOL
Methods	Method Name	Description
	viewMartReport()	Method for Admin to view mart report details from the database.

	calculateSales()	Method for Admin to calculate the mart sales from the database.
	viewSales()	Method for Admin to view the calculated mart sales result from the database.
	editSales()	Method for Admin to edit the calculated mart sales from the database.
	addOpeningSales()	Method for Cashier to add opening sales data into database.
	editOpeningSales()	Method for Cashier to edit opening sales data from the database.
	addClosingSales()	Method for Cashier to add closing sales data into database.
	editClosingSales()	Method for Cashier to edit the closing sales data from the database.

Algorithm	**viewMartReport()**
	BEGIN
	public function index(){
	$sales_data = \App\Models\Sales_model::all();
	return view('index',['sales_data'=>$sales_data]);
	}
	
	calculateSales()
	BEGIN
	public function calculate(Request$request,$id){
	$sales_data = \App\Models\Sales_model::find($id);
	$sales_data->calculate)$request->all());
	return redirect('/sales_data')->with('success','Data is been calculated');
	}
	viewSales()
	BEGIN
	public function index(){
	$sales_data = \App\Models\Sales_model::all();
	return view('index',['sales_data'=>$sales_data]);
	}
	
	editSales()
	BEGIN
	public function edit($id){
	$sales_data = \App\Models\Sales_model::find($id);	

	return view('edit',['sales_data'=>$sales_data]);
	}
	
	addOpeningSales()
	BEGIN
	public function create(Request $request){
	\App\Models\Sales_model::create($request->all());
	return redirect('/sales_data')->with('success';New Data Insert!);
	}
	
	editOpeningSales()
	BEGIN
	public function edit($id){
	$sales_data = \App\Models\Sales_model::find($id);
	return view('edit',['sales_data'=>$sales_data]);
	}
	
	addClosingSales()
	BEGIN
	public function create(Request $request){
	\App\Models\Sales_model::create($request->all());
	return redirect('/sales_data')->with('success';New Data Insert!);
	}
	
	editClosingSales()
	BEGIN
	public function edit($id){
	$sales_data = \App\Models\Sales_model::find($id);
	return view('edit',['sales_data'=>$sales_data]);
	}

4.3.11 Sales_Controller.blade.php [SDD-REQ-315]

Class Type	Controller class	
Responsibility	Act as a getaway from one action to another action.	
Attributes	Attributes Name	Attributes Type
	Not applicable	Not applicable
Methods	Method Name	Description

	viewMartReport()	Method for Admin to view mart report details from the database.
	calculateSales()	Method for Admin to calculate the mart sales from the database.
	viewSales()	Method for Admin to view the calculated mart sales result from the database.
	editSales()	Method for Admin to edit the calculated mart sales from the database.
	addOpeningSales()	Method for Cashier to add opening sales data into database.
	editOpeningSales()	Method for Cashier to edit opening sales data from the database.
	addClosingSales()	Method for Cashier to add closing sales data into database.
	editClosingSales()	Method for Cashier to edit the closing sales data from the database.
Algorithm	**viewMartReport()** BEGIN Receive a GET request to show all mart report data. Retrieve all the necessary mart report data from the database, using Lravel's query ORM like Eloquent. Return the mart report data to the view, as an array or collection. Render the view with the vendor data and return it to the Admin's browser. END **calculateSales()** BEGIN Receive a POST request to submit the edited data. Validate the Cashier input, using Laravel's built-in validation tools Update the calculated sales data in the database, based on the sales's input. Redirect the user to the Admin main page for the report, and show a success message. END **viewSales()** BEGIN	

Receive a GET request to show specific sales data.
Retrieve the necessary sales data from the database, using Laravel's query ORM like Eloquent, based on the vendor's ID
Return the sales data to the view, as an array or object.
Render the view with the sales data and return it to the Admin's browser.
END

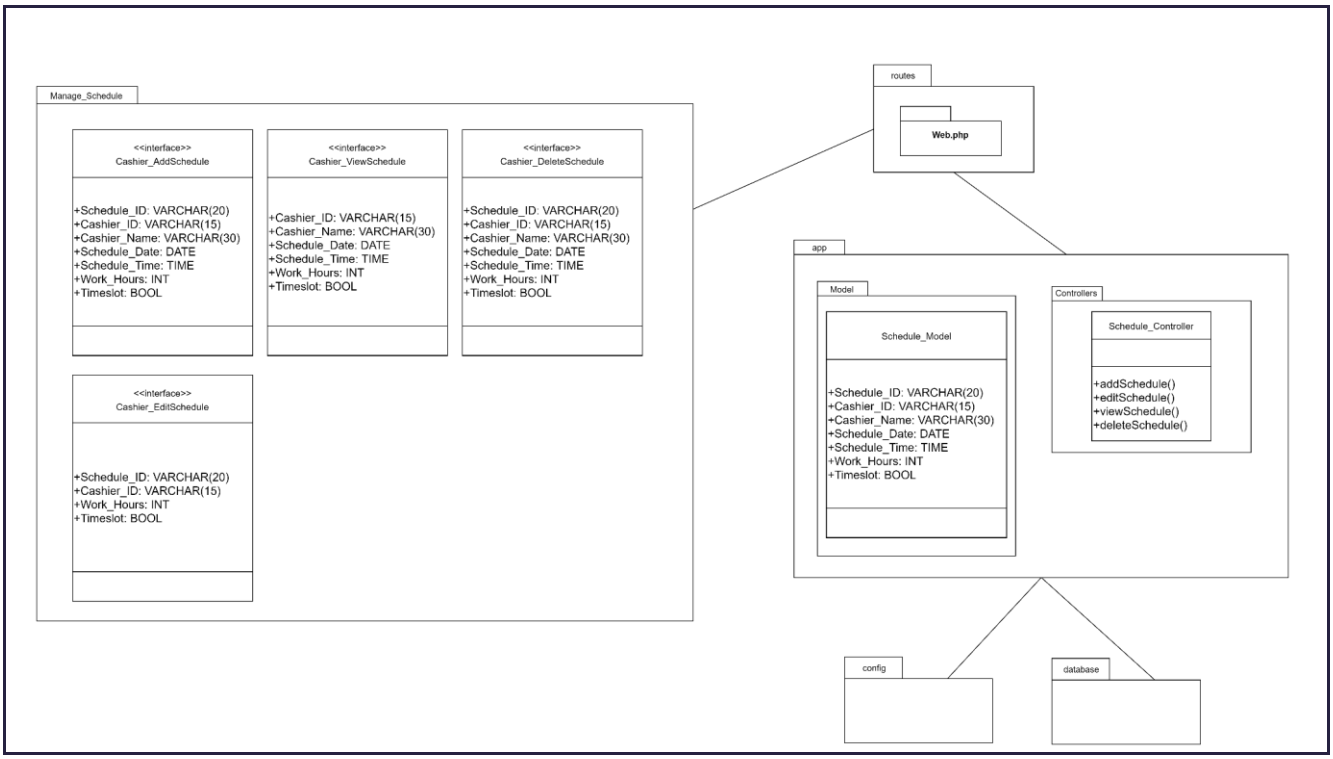
editSales()
BEGIN
Receive a GET request to show the form to edit a specific calculated sales data.
Retrieve the necessary sales data from the database, using Laravel's query ORM like Eloquent, based on the mart report sales ID
Return the form view, with the data item pre-populated in the form fields.
END

addOpeningSales()
BEGIN
Receive a GET request to display the form to add a new opening sales data.
Display the form view to the Cashier.
Receive a POST request with the data submitted by the Cashier.
Validate the submitted data using Laravel's built-in validation functionality.
If the validation fails, return the Cashier to the form view with error messages.
If the validation succeeds, create a new opening sales data in the database using Laravel's query ORM like Eloquent.
Redirect the user to the cashier main page for the report, and show a success message.
END

editOpeningSales()
BEGIN
Receive a GET request to show the form to edit a specific opening sales form data.
Retrieve the necessary opening sales data from the database,

	using Laravel's query ORM like Eloquent, based on the opening sales ID, Return the form view, with the data item pre-populated in the form fields. END **addClosingSales()** BEGIN Receive a GET request to display the form to add a new closing sales data. Display the form view to the Cashier. Receive a POST request with the data submitted by the Cashier. Validate the submitted data using Laravel's built-in validation functionality. If the validation fails, return the Cashier to the form view with error messages. If the validation succeeds, create a new closing sales data in the database using Laravel's query ORM like Eloquent. Redirect the user to the Cashier main page for the report, and show a success message. END **editClosingSales()** BEGIN Receive a GET request to show the form to edit a specific closing sales form data. Retrieve the necessary closing sales data from the database, using Laravel's query ORM like Eloquent, based on the closing sales ID, Return the form view, with the data item pre-populated in the form fields. END

4.4 Manage Schedule [SDD-REQ-400]



4.4.1 Cashier_AddSchedule.blade.php [SDD-REQ-401]

Class Type	Boundary class	
Responsibility	An interface is used by the Cashier to add new schedule.	
Attributes	Attributes Name	Attributes Type
	Schedule_ID Cashier_ID Cashier_Name Schedule_Date Schedule_Time Work_Hours Timeslot	VARCHAR(20) VARCHAR(15) VARCHAR(30) DATE TIME INT BOOL
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	BEGIN The Cashier clicks on “Add Schedule” button	

	System display the latest schedule details form
	
	The Cashier enters the Cashier_ID, Cashier_Name, Timeslot, and Work_Hours in the form.
	
	The cashier clicks on "Add" button.
	IF the Timeslot is available
	
	THEN
	System saves the latest schedule
	System pop-up successful message
	System display the latest schedule in ViewSchedule.blade.php
	
	ELSE
	System pop-up errors message
	END IF
	
	END

4.4.2 Cashier_EditSchedule.blade.php [SDD-REQ-402]

Class Type	Boundary class	
Responsibility	An interface is used by the Cashier to edit schedules.	
Attributes	Attributes Name	Attributes Type
	Schedule_ID Cashier_ID Work_Hours Timeslot	VARCHAR(20) VARCHAR(15) INT BOOL
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	BEGIN The Users click on “Edit Schedule” button System display the latest schedule DISPLAY Schedule_ID DISPLAY Cashier_ID DISPLAY Work_Hours DISPLAY Timeslot	

	Cashier updates Work_Hours
	Cashier updates Timeslot
	
	The Cashier clicks on "Update" button
	IF Timeslot is available
	THEN
	System saves new timeslot in the database
	System pop-up successful message
	System display the latest schedule in ViewSchedule.blade.php
	ELSE
	System pop-up error message
	Cashier remains at the EditSchedule.blade.php page
	END IF
	END

4.4.3 Cashier_DeleteSchedule.blade.php [SDD-REQ-403]

Class Type	Boundary class	
Responsibility	An interface is used by the Cashier to add a new schedule.	
Attributes	Attributes Name	Attributes Type
	Schedule_ID Cashier_ID Cashier_Name Schedule_Date Schedule_Time Work_Hours Timeslot	VARCHAR(20) VARCHAR(15) VARCHAR(30) DATE TIME INT BOOL
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	BEGIN System display the latest schedule DISPLAY Schedule_ID DISPLAY Schedulue_Date DISPLAY Schedule_Time DISPLAY Cashier_ID DISPLAY Cashier_Name DISPLAY Work_Hours	

	DISPLAY Timeslot
	
	IF Cashier clicks on “Delete” button on appears on the timeslot list
	
	THEN
	
	The selected timeslot will be deleted
	System will pop-out successful message
	System delete the selected details on the database and update the latest schedule
	System displays new latest schedule in ViewSchedule.blade.php
	
	ELSE IF Cashier clicks on the “Cancel” button
	
	THEN
	
	The delete process is canceled
	System pop-out message on cancellation
	
	ELSE
	
	Remains at the DeleteSchedule.blade.php page
	
	END IF
	
	END

4.4.4 Cashier_ViewSchedule.blade.php [SDD-REQ-404]

<table>
<tr><td>Class Type</td><td colspan="2">Boundary class</td></tr>
<tr><td>Responsibility</td><td colspan="2">An interface is used by the Cashier to view schedule.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>Cashier_ID
Cashier_Name
Schedule_Date
Schedule_Time
Work_Hours
Timeslot</td><td>VARCHAR(15)
VARCHAR(30)
DATE
TIME
INT
BOOL</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">BEGIN
System display all the latest schedule
DISPLAY Schedule_Date
DISPLAY Schedule_Time
DISPLAY Cashier_ID
DISPLAY Cashier_Name
DISPLAY Work_Hours
DISPLAY Timeslot

IF Cashier click on “Back” button
THEN the page direct to the homepage
ELSE
Remain at the ViewSchedule.blade.php page
END IF

END</td></tr>
</table>

4.4.5 Schedule_Model.blade.php [SDD-REQ-404]

<table>
<tr><td>Class Type</td><td colspan="2">Model Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">To store, retrieve and manage database of schedule</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>Schedule_ID
Cashier_ID
Cashier_Name
Schedule_Date
Schedule_Time
Work_Hours
Timeslot</td><td>VARCHAR(20)
VARCHAR(15)
VARCHAR(30)
DATE
TIME
INT
BOOL</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>Not applicable</td><td>Not applicable</td></tr>
<tr><td>Algorithm</td><td colspan="2">

Add Schedule()
BEGIN

```
public function create(Request $request){
    \App\Models\Schedule_model::create($request->all());
    return redirect('/schedule_data')->with('success','New schedule  has been added!');
  }
```

END

EditSchedule()
BEGIN

```
public function update(Request $request, $product_id){
  $data_inventory = \App\Models\Inventory::find($product_id);
  $data_inventory->update($request->all());
  return redirect('/scheduledata')->with('success','Product is Updated');
 }
```

END

</td></tr>
</table>

	DeleteSchedule()
	BEGIN
	
	public function delete($product_id){
	$schedule_data = \App\Models\Schedule_mode::find($product_id);
	$schedule_data->delete($schedule_data);
	return redirect('/scheduledata')->with('success','Deleted');
	}
	
	END
	
	View Schedule
	BEGIN
	
	public function index(){
	$schedule_data = \App\Models\Schedule_model::all();
	return view('index',['schedule_data'=> $schedule_data]);
	}
	
	END

4.4.5 Schedule_Controller.blade.php [SDD-REQ-404]

Class Type	Model Class	
Responsibility	To store, retrieve and manage database of schedule	
Attributes	Attributes Name	Attributes Type
	Schedule_ID Cashier_ID Cashier_Name Schedule_Date Schedule_Time Work_Hours Timeslot	VARCHAR(20) VARCHAR(15) VARCHAR(30) DATE TIME INT BOOL
Methods	Method Name	Description
	addSchedule()	Method for the cashier to add new schedule

		into the database
	editSchedule()	Method for the cashier to edit the existing schedule from the database
	viewSchedule( )	Method for the cashier to view the latest schedule from the database
	deleteSchedule ()	Method for the cashier to delete schedule from the database.
Algorithm	**addSchedule()** BEGIN Receive a GET request to display the form to add a new schedule data. Display the form view to the user. Receive a POST request with the data submitted by the cashier. Validate the submitted data using Laravel's built-in validation functionality. If the validation fails, return the cashier to the form view with error messages. If the validation succeeds, create a new inventory data in the database using Laravel's query ORM like Eloquent. Show a success message and redirect to the main page of schedule. END **ViewSchedule()** BEGIN Receive a GET request to show schedule details. Retrieve the necessary schedule data from the database, using Laravel's query ORM like Eloquent, based on the chosen date and time that refers to the schedule ID Return the schedule data to the view, as an array or object. Render the view with the schedule data and return it to the cashier's browser. END **editSchedule()** BEGIN Receive a POST request to submit the edited schedule data. Validate the cashier input, using Laravel's built-in validation tools Update the schedule data in the database, based on the cashier's input. Show a success message and redirect to the main page of schedule with the latest schedule displays. END	

	deleteSchedule() BEGIN Receive a POST request to delete a specific schedule data. Retrieve the necessary schedule data from the database, using Laravel's query ORM like Eloquent, based on the product's ID Delete the schedule data from the database. Show a success message and redirect to the main page of schedule with the latest schedule displays. END

5. REQUIREMENT TRACEABILITY

5.1 Manage Inventory [SRS-REQ-100]

Muhammad Ammar bin Mohd Rosli [CB20037]

Requirement ID	Description	Design ID
PMMS_REQ_101	The Admin, PETAKOM Committee and Cashier able to to view the inventory level and product details	SDD-REQ-101
PMMS_REQ_102		
PMMS_REQ_105		
PMMS_REQ_103	The PETAKOM Committee should be able to update the low level inventory product by inserting the quantity of the products	SDD-REQ-103
PMMS_REQ_107		
PMMS_REQ_107	The Admin should be able to Add the new product arrive from the invoice by inserting the product code, name, company name and the quantity of the product	SDD-REQ-102
PMMS_REQ_104		
PMMS_REQ_106	The PETAKOM Committee and Admin should be able to delete the product.	SDD-REQ-104

5.2 Manage Vendor [SRS-REQ-200]

AHMAD DANIAL AQIL BIN SALHIN [CB20049]

Requirement ID	Description	Design ID
PMMS-REQ-204 PMMS-REQ-205	The system shall display list of vendor information to Admin	SDD-REQ-201
PMMS-REQ-201	The system shall allow Admin to input new vendor or supplier details and click Confirm button to update the information list	SDD-REQ-202
PMMS-REQ-203	The system shall allow Admin to select previous vendor and click Delete button to remove the vendor details	SDD-REQ-203
PMMS-REQ-202	The system shall allow Admin to select previous vendor and edit vendor details and click Update button to update the information details	SDD-REQ-204

5.3 Manage Sales [SRS-REQ-300]

Muhammad Syakir Bin Hazli [CB20125]

Requirement ID	Description	Design ID
PMMS_REQ_301	The system shall allow Cashiers to enter homepage and choose action they intend to.	SDD-REQ-309
PMMS_REQ_302		
PMMS_REQ_303		
PMMS_REQ_304	The system shall allow Cashiers to add an opening sales and closing sales data.The system also shall allow cashiers to edit the opening and closing sales report form data.	SDD-REQ-310
		SDD-REQ-311
		SDD-REQ-312
		SDD-REQ-313
PMMS_REQ_305	The system shall allow Admin to enter the admin main page.	SDD-REQ-301
PMMS_REQ_306		
PMMS_REQ_307	The system shall allow Admin to be able to calculate everyday sales and can save the result of daily sales into the database. System also shall allow Admin to edit and view the sales result.	SDD-REQ-302
		SDD-REQ-303
		SDD-REQ-304
PMMS_REQ_308	The system shall allow Admin to able to generate mart reports for daily,weekly,monthly and yearly for PMMS	SDD-REQ-305
		SDD-REQ-306
PMMS_REQ_309	The system shall allow Admin to make a payment for vendors.	SDD-REQ-307
PMMS_REQ_310	The system shall allow Admin to make a salary payment for Cashiers.	SDD-REQ-308

5.4 Manage Schedule [SRS-REQ-400]

Siti Shazwanie Nurain Binti Mohd Shafiq [CB20008]

Requirement ID	Description	Design ID
PMMS_REQ_401	The system shall display the schedule page.	SDD_REQ_401
PMMS_REQ_402	Cashier shall be able to add schedules by clicking the add button.	SDD_REQ_402
PMMS_REQ_403	Cashiers shall be able to delete the registered slot by clicking the delete button.	SDD_REQ_403
PMMS_REQ_404	Cashiers shall be able to edit their details for the schedule by clicking the edit button.	SDD_REQ_404