Design and Implementation of Point of Sales Management System

by

Jakia Sultana
ID: CSE2103024013

Rasel Mia
ID: CSE2103024021

Hasanuzzaman
ID: CSE2103024063

Saiful Islam
ID: CSE2103024029

MD.Solaiman
ID: CSE2003021019

Supervised by
Arifur Rahaman

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

May 2025

APPROVAL

The project titled **"Design and Implementation of Point of Sales Management System"** submitted by Jakia Sultana (CSE2103024013), Rasel Mia (CSE2103024021), Saiful Islam (CSE2103024029), Hasanuzzaman (CSE2103024063), MD.Solaima (CSE2003021019), to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Arifur Rahaman
Assistant Professor & Coordinator
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 4

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 5

DECLARATION

We hereby declare that the work presented in this report is the outcome of our own efforts, carried out under the supervision of **Arifur Rahaman**, Assistant Professor and Coordinator, Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. This project has not been submitted elsewhere for the award of any degree or diploma.

Countersigned

(Arifur Rahaman)
Assistant Professor and Coordinator
Supervisor

Signature

Jakia Sultana
ID: CSE2103024013

Rasel Mia
ID: CSE2103024021

Hasanuzzaman
ID: CSE2103024063

Saiful Islam
ID: CSE2103024029

MD.Solaiman
ID: CSE2003021019

ABSTRACT

In today's competitive business environment, many small and medium-sized enterprises (SMEs) still rely on manual processes to manage daily operations such as sales, inventory, purchases, and expenses. These traditional methods are often inefficient, time-consuming, and prone to human error. To address these challenges, our team developed the Point of Sales Management System — a comprehensive web-based solution designed to automate and streamline business activities. The system manages product and stock information, generates professional invoices, tracks customer and supplier interactions, records expenses, and provides real-time reports. Built using modern technologies like Laravel, HTML5, CSS3, JavaScript, and MySQL, it offers a responsive, secure, and user-friendly experience. With features like role-based access control and a dynamic dashboard, the system helps reduce manual workload, improve accuracy, and enable better decision-making, ultimately enhancing the overall efficiency and productivity of small business operations.

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Arifur Rahaman**, Assistant Professor and Coordinator, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheet
DDL	Data Definition Language
DEY	Don’t Repeat Yourself
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model View Controller
ORM	Object Relational Mapping
POS	Point of Sale
RAM	Random Access Memory
RDBMS	Relational Database Management System
ROM	Read Only Memory
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
UX	User Experience
UAT	User Acceptance Testing
VAT	Value Added Tax
P&L	Profit and Loss
SKU	SKU Stock Keeping Unit
DBMS	DBMS Database Management System

TABLE OF CONTENTS

Title			Page No.
DECLARATION			i
ABSTRACT			ii
ACKNOWLEDGEMENT			iii
LIST OF ABBREVIATION			Iv
CHAPTER 1			1 – 3
INTRODUCTION TO POINT OF SALE (POS) SYSTEM			
1.1	Introduction		1
1.2	Project Objective		1
1.3	Project Planning		1
1.4	Problem Statement		2
1.5	Feasibility Study		2
1.6	Requirement Specification		3
	1.6.1	Hardware Requirements	3
	1.6.2	Software Requirements	3
CHAPTER 2			4-5
BACKGROUND OF POINT OF SALE SYSTEM & TECHNOLOGIES USED			
2.1	Background of POS Systems		4
2.2	Evolution of Sales Systems.		4
2.3	Technologies Used in the Project		4
	2.3.1	Laravel Framework	4
	2.3.2	HTML5	5
	2.3.3	CSS3	5
	2.3.4	JavaScript	5
	2.3.5	MySQL Database	5
	2.3.6	Blade Template Engine	5
2.4	Why These Technologies Were Chosen		5
2.5	Advantages of the Tech Stack		5
2.6	Related Projects / Existing Systems		5

CHAPTER 3 6-7

OBJECTIVES OF THE PROPOSED SYSTEM

3.1 Automation Goals……..... 6

3.2 Business Needs Addressed…… 6

3.3 Key Functional Goals……..... 6-7

3.4 Long-term Scalability…… 7

CHAPTER 4 8-12

SYSTEM ANALYSIS OF THE PROJECT

4.1 System Overview ……. 8

4.2 Functional Requirements ……..... 9

4.3 Non-Functional Requirements ……. 9

4.4 Use Case Overview ……..……. 9-10

4.5 System Workflow Diagram …….. 11

4.6 ER Diagram …….. 11-12

CHAPTER 5 13-17

SYSTEM DESIGN

5.1 System Architecture …….……..... 13

5.2 Module-Wise Design ……..... 14

5.2.1 User Module……. 14

5.2.2 Product & Inventory Module…… 14

5.2.3 Sales & Purchase Module……. 15

5.2.4 Expense Management Module……. 16

5.3 Data flow of POS Management …… 16-17

CHAPTER 6 18-20

IMPLEMENTATION

6.1 Backend Implementation – Laravel Controllers & Routes……..... 18

6.2 Frontend Implementation – Blade + HTML/CSS…….……..... 18

6.3 Database Integratio…… 18-19

6.4 Role-Based Access Control (RBAC)……. 19-20

6.5 Security Measures…….. 20

CHAPTER 7 21-33

TESTING AND RESULTS

7.1 Testing Strategy……. 21

7.2 Unit Testing and Bug Fixing……. 21-22

7.3 Integration and UAT ……. 22

7.4 Screenshots and Output Pages……... 23

7.5 Project Over View……. 24-33

CHAPTER 8 34-36

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion …..... 34

8.2 Limitations of the System…….. 34

8.3 Future Enhancements……. 34

REFERENCES…….. 35

LIST OF FIGURES

Figure No.	Title	Page No.
Fig 1.1	Project Planning	2
Fig 4.1	Software Development life cycle	8
Fig 4.2	USDP	10
Fig 4.3	Use case Diagram for pos	10
Fig 4.4	Pos system workflow	11
Fig 4.5	ER Diagram	12
Fig 5.1	System Architecture Diagram	13
Fig 5.2	Add Product	14
Fig 5.3	Sales Invoice	15
Fig 5.4	Purchase Entry Interface	15
Fig 5.5	Expense Entry Interface	16
Fig 5.6	Data Flow Diagram – Level 1	17
Fig 6.1	ER Diagram (Linked with Chapter 4)	19
Fig 6.2	User Role and Access Settings Page	20
Fig 7.1	Expense Report Output	23
Fig 7.2	Dashboard Overview	24
Fig 7.3	Profit Overview	24
Fig 7.4	Suppliers Overview	25
Fig 7.5	Customer Overview	25
Fig 7.6	Customer group Overview	25
Fig 7.7	Import Contacts Overview	26
Fig 7.8	List Purchases Overview	26
Fig 7.9	Add Product Overview	27
Fig 7.10	Print Labels Overview	27
Fig 7.11	Variations Overview	28
Fig 7.12	Import product Overview	28
Fig 7.13	Import Opening Stock	29
Fig 7.14	Selling Price Group	29
Fig 7.15	Units overview	29
Fig 7.16	Categories overview	30
Fig 7.17	Brands overview	30
Fig 7.18	List Purchase	30
Fig 7.19	Add Purchase	31

Fig 7.20	Purchase Return	31
Fig 7.21	Sales Overview	31
Fig 7.22	Expenses Overview	32
Fig 7.23	Invoice Setting Overview	32
Fig 7.24	Barcodes Setting Overview	32
Fig 7.25	Pinters Setting Overview	33
Fig 7.26	Tax Rates Setting Overview	33

CHAPTER 1

INTRODUCTION TO POINT OF SALE SYSTEM

1.1 Introduction

In the modern retail and service industry, efficiency, speed, and accuracy in business transactions are essential. Traditional paper-based and manual billing systems often lead to data loss, errors, and inefficiencies in inventory and expense tracking. These limitations drive the demand for intelligent, automated solutions.

Our team developed the Easy Sell Point of Sales System as a web-based solution to help businesses automate daily operations. This system offers features such as product and inventory management, customer and supplier handling, invoicing, sales tracking, and real-time reporting. Built using Laravel, HTML5, CSS3, JavaScript, and MySQL, the system ensures fast performance, security, and user-friendliness.

1.2 Project Objective

The main objective of the Easy Sell Point of Sales System is to simplify and automate sales and inventory management tasks, thereby increasing efficiency and accuracy in business processes.

Our system was developed with the following goals:

a) Automate the billing and invoicing process
b) Maintain up-to-date inventory records
c) Manage customer and supplier information
d) Record and categorize business expenses
e) Generate detailed sales and profit reports
f) Ensure multi-user access with role-based control
g) Enable secure, fast, and user-friendly interface

1.3 Project planning

We followed the Software Development Life Cycle (SDLC) model for structured development.
Our project was divided into the following phases:

No	Task Name	Duration	Start	End
1	Requirement Analysis	3 Month	06-01-2024	06-03-2024
2	System Design	3 Month	07-03-2024	16-06-2024
3	Development	3 Month	17-06-2024	09-09-2024
4	Testing and Debugging	3 Month	10-09-2024	30-12-2024
5	Deployment and Documentation	3 Month	31-12-2024	30-03-2025

Table 1.1 Project Plan

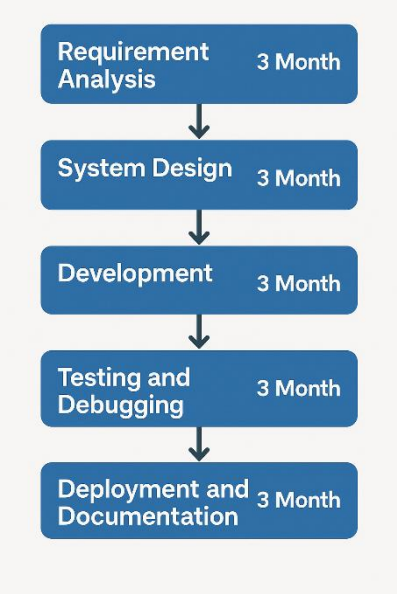


Fig 1.1 Project Planning

1.4 Problem Statement

Businesses using manual systems face several problems:

- Errors in billing and calculations
- Mismanaged stock levels
- Delays in generating reports
- No centralized platform for business operations
- Poor customer and supplier tracking
- Lack of user access control and data security

The Easy Sell POS System solves these problems by providing a single, integrated platform for managing all business activities.

1.5 Feasibility Study for a Hospital Management System (HMS)

Our team evaluated feasibility in three main areas:

- **Technical Feasibility:** The system was built using open-source technologies (Laravel, MySQL, etc.) making it cost-effective and scalable.
- **Operational Feasibility:** The system interface is designed to be user-friendly even for non-technical users.
- **Economic Feasibility:** The development cost was low and the return on investment is high due to time and effort saved in daily operations.

1.6 Requirement Specification

1.6.1 Hardware requirements

To develop this system, hardware requirement is important things to be required. Table 1.2 shows the hardware requirements to develop Hospital Management System (HMS).

Hardware	Description
Laptop with 16GB of RAM	Workstation for the project
Hard disk space (512GB)	To save data
Printer	A printer to test the report generating function.
Others	Keyboard and Microsoft Mouse or compatible pointing device and Android mobile etc.

Table 1.2 Hardware requirements

1.6.2 Software requirements

After hardware requirement, software requirement is the important things to be install.
Table 1.3 shows the software requirement to develop Hospital Management System (HMS) and Android studio.

Software	Description	Cost (RM)
PHP 8.x with Laravel Framework	Backend	Free
Frontend	HTML5, CSS3, JavaScript	Free
Java	Android version	Free
Microsoft Office 2007	Documentation Tools.	Free
Visual studio	Website coding editing tool	Free
MySQL	Database used for this system.	Free
Apache	Database used for this system.	Free

Table 1.3 Software Requirements

CHAPTER 2

BACKGROUND OF POINT OF SALE SYSTEM & TECHNOLOGIES USED

Although the system is code-based, the UI design was influenced by visual programming layout standards [7].

2.1 Background

Point of Sale (POS) systems are essential in today's retail and service industries. Previously, shopkeepers and businesses relied on handwritten invoices and physical ledgers. These systems were time-consuming and prone to error. Over time, POS systems evolved from basic desktop billing tools to modern, cloud-connected web applications.

Our team identified a common problem: small businesses often use either outdated software or manual methods. To solve this, we created the Easy Sell Point of Sale System, a browser-based solution designed for simplicity, accuracy, and real-time performance.

2.2 Evolution of Sales Systems

The evolution of sales systems can be divided into the following stages:

- **Manual Billing:** Ledger books, hand-written bills
- **Digital Spreadsheet Tools:** Excel, Tally (basic use)
- **Offline POS Applications:** Desktop-based POS systems
- **Online POS Systems:** Web applications like Odoo POS
- **Cloud-based Integrated Systems:** Fully hosted POS with mobile/web access and auto-backup

Our project belongs to the **online web application** category, which provides real-time access and role-based control.

2.3 Technologies Used in the Project

To develop this project efficiently, we selected the following technologies:

2.3.1 Laravel Framework

Laravel is a PHP framework that supports MVC design, built-in authentication, and clean routing. It helped us organize the codebase and build secure, scalable features. [1]

2.3.2 HTML5

HTML5 is used for building the structure of web pages. We used it for form layouts, buttons, tables, and overall content presentation. [2]

2.3.3 CSS3

CSS3 adds styling to the pages. Our team used it for making the interface responsive, visually clean, and mobile-friendly. [3]

2.3.4 JavaScript

JavaScript controls dynamic behavior such as filtering, form validation, dropdown toggles, and alert popups. [4]

2.3.5 MySQL Database

MySQL stores all system data — customers, products, transactions, invoices, users. It's reliable and integrates well with Laravel via Eloquent ORM. [5]

2.3.6 Blade Template Engine

Blade, Laravel's default template engine, allows reusable layout components and helps keep UI files clean and organized.

2.4 Why These Technologies Were Chosen

These tools and frameworks were selected because they are:

- Open-source (free to use)
- Widely documented and community-supported
- Easy to learn and scalable
- Secure and robust for business use
- Responsive across devices
- Easily deployed in local or cloud environments

2.5 Advantages of Using This Stack

- Laravel simplifies routing, validation, and security
- HTML5/CSS3 ensures a structured and responsive UI
- MySQL offers fast queries and backups
- Blade allows DRY (Don't Repeat Yourself) UI structure
- JavaScript makes the interface interactive

2.6 Related Projects / Existing Systems

Before developing Easy Sell POS, we reviewed several existing systems:

- PHP Point Of Sale
- Retail Cloud POS
- Odoo POS
- Tally ERP (partial POS)

These systems are either too complex or too costly for small businesses. Our system focuses on simplicity, essential features, and speed — the perfect match for growing businesses.

CHAPTER 3

OBJECTIVES OF THE PROPOSED SYSTEM

The Easy Sell POS System was developed with the aim to provide small and medium-sized businesses with a fast, accurate, and reliable platform for managing their daily operations.
Our team focused on solving real-world problems such as manual billing errors, lack of inventory control, customer dues mismanagement, and unorganized reporting.

3.1 Automation Goals

Our system aims to replace all manual sales, purchase, and stock management processes with a centralized digital platform.
The key automation goals include:

- Auto-generation of invoices
- Real-time product stock update
- Automatic due tracking
- Expense calculation
- Auto-calculated profit & loss reports
- One-click report generation

3.2 Business Needs Addressed

Our system was built based on the challenges businesses face in the retail and wholesale sectors.
It addresses the following business needs:

- **Sales Tracking:** Every transaction is recorded with timestamp and product details
- **Inventory Monitoring:** Automatic stock adjustment after every purchase/sale
- **Customer Management:** Save, update, and view full customer ledger and due
- **Supplier Tracking:** Record purchases and payment status to suppliers
- **Multi-user Support:** Allow different staff roles with permission control
- **Expense Entry:** Track daily business expenditures by category

3.3 Key Functional Goals

Our team decided on a set of functional goals for the system:

1. Invoice System**:**
    - Clean UI for sales/purchase invoice creation with payment mode selection.
2. Stock Management**:**
    - Opening stock setup, real-time stock alerts, product filtering

3. Payment Handling**:**
    - Multiple payment modes (cash, card, bank), full/partial payments, due
4. Reports:
    - Sales Report, Purchase Report, Expense Report, Profit & Loss Report, Customer Dues

5. User Role Management:
    - Admin creates staff accounts and assigns modules they can access/edit/view only
6. Accessibility:
    - Cloud or LAN-based setup with secure login

3.4 Long-term Scalability

We designed the system keeping in mind future upgrades.
Some long-term objectives include:

- Mobile version (Android/iOS)
- SMS/Email integration for invoice/due notification
- Barcode scanner support
- Cloud hosting with daily auto backup
- Export to Excel/PDF for accounting

CHAPTER 4

SYSTEM ANALYSIS AND REQUIREMENTS

System analysis helps in understanding and documenting the functional flow, structure, and core requirements of the proposed solution.

Our team performed a complete analysis to identify the pain points in traditional sales systems and ensure that the Easy Sell POS System covers all required features and usability needs.

The system development process followed a structured methodology involving data gathering, analysis, and design [9].

4.1 System Overview

The Easy Sell POS System is designed as a modular, web-based application that allows businesses to:
A system development life cycle is a logical process by which system analysts, software engineers, programmers, and end users build information systems and computer applications to solve business problems and needs.

- Manage inventory, sales, purchases, expenses, and reports
- Support multiple user roles with different permissions
- Provide a real-time dashboard for business summary
- Ensure secure login and structured data storage

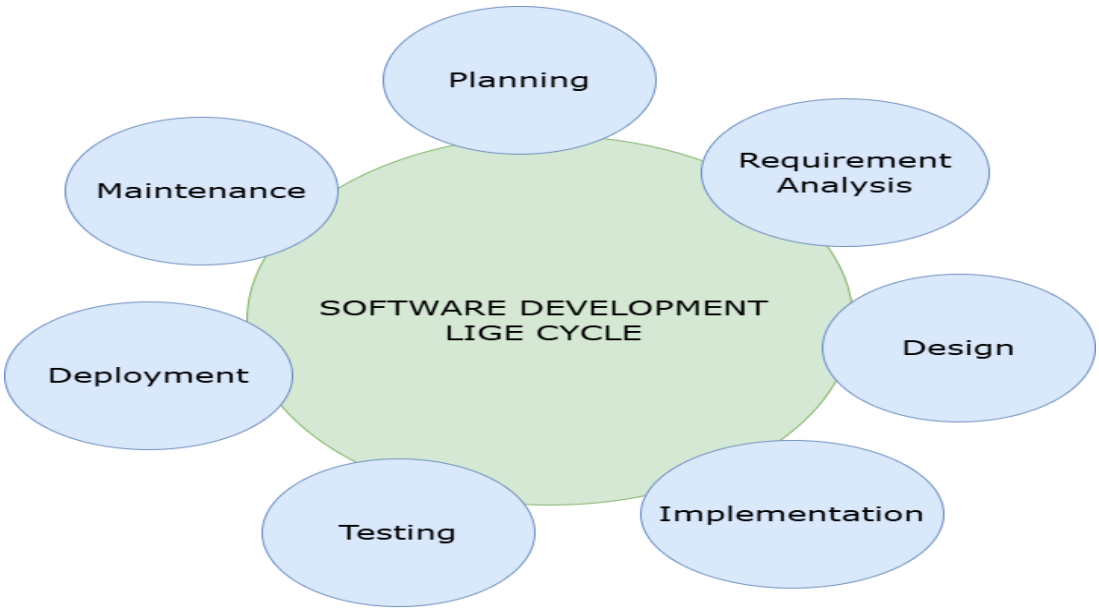


Figure 4.1: Software Development Life Cycle.

4.2 Functional Requirements

Functional requirements describe what the system should do.
The following features are considered core to our system:

- User Login & Role Access
- Customer & Supplier Management
- Add / Update / Delete Products
- Invoice Creation (Sales & Purchase)
- Real-time Stock Management
- Expense Recording and Categorization
- Multi-payment Support (Cash, Bank, Card)
- Report Generation (Sales, Purchase, Expense, P&L)
- Printable Invoice / Download PDF
- Data Filtering by Date, Party, Product

Each module is interconnected and works in real time to ensure smooth transactions and up-to-date reports.

4.3 Non-Functional Requirements

These are system properties that affect **how** the system performs:

- **Performance:** Fast loading time for dashboard and reports
- **Scalability:** Can be extended for multi-branch operations
- **Security:** Password encryption, CSRF protection, RBAC
- **Availability:** Works in LAN or Web Hosting environment
- **Backup:** MySQL database backup capability

4.4 Use Case Overview

A typical user's interaction with the system follows this path:

- Login as Admin/User
- Access Dashboard
- Add Products, Customers, Suppliers
- Record Sale / Purchase
- View Stock Update
- Enter Daily Expenses
- Generate Reports
- Logout

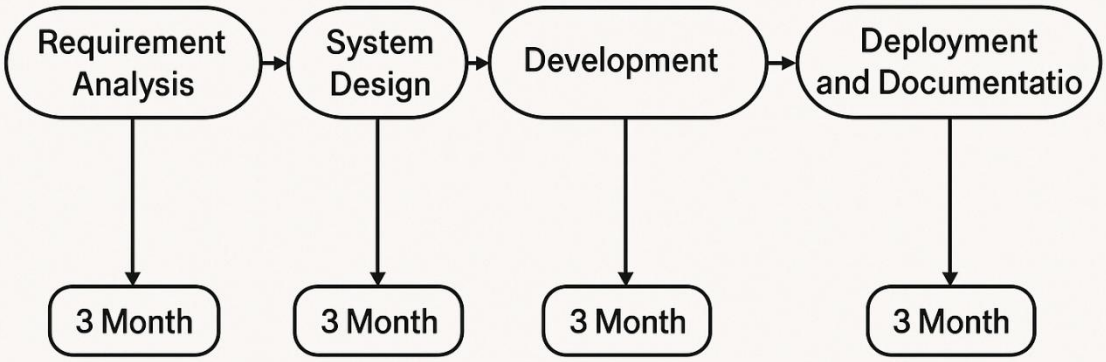


Fig 4.2: Unified Software Development Process (USDP)

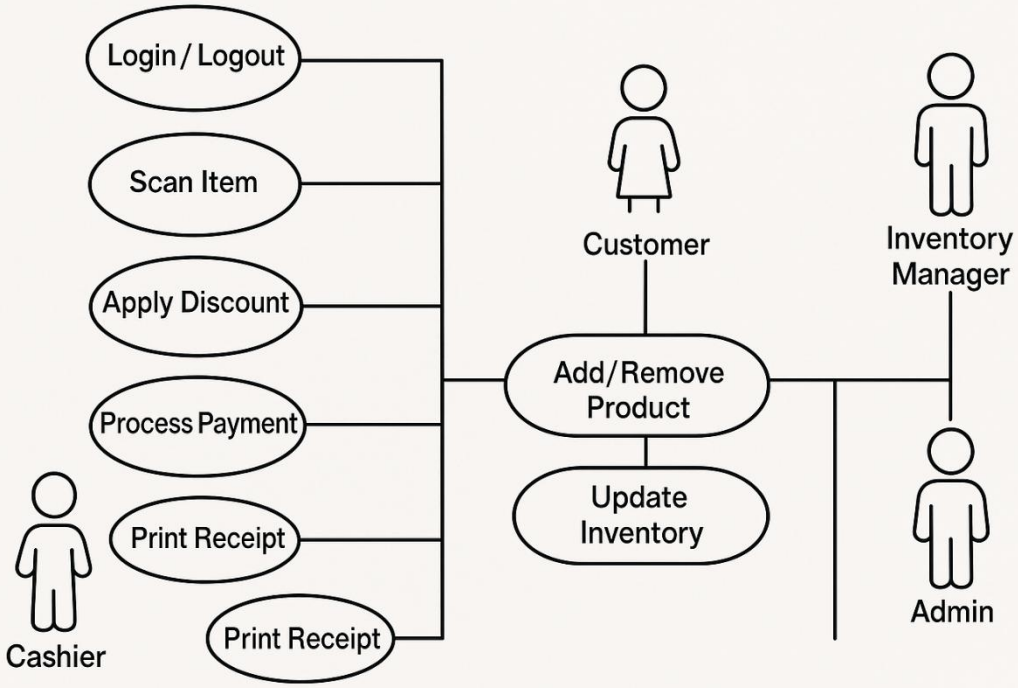


Fig 4.3: Use Case Diagram for POS Software

4.5 System Workflow Diagram

This flow explains how modules interact:

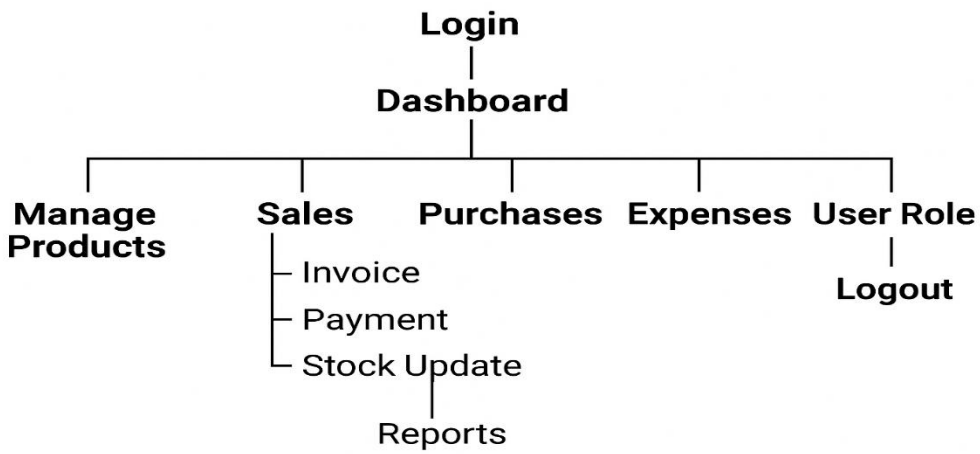


Fig 4.4: POS System Workflow Diagram

4.6 ER Diagram

The Entity Relationship Diagram (ERD) represents the database structure and shows how entities like customers, products, invoices, payments, etc., relate to each other.

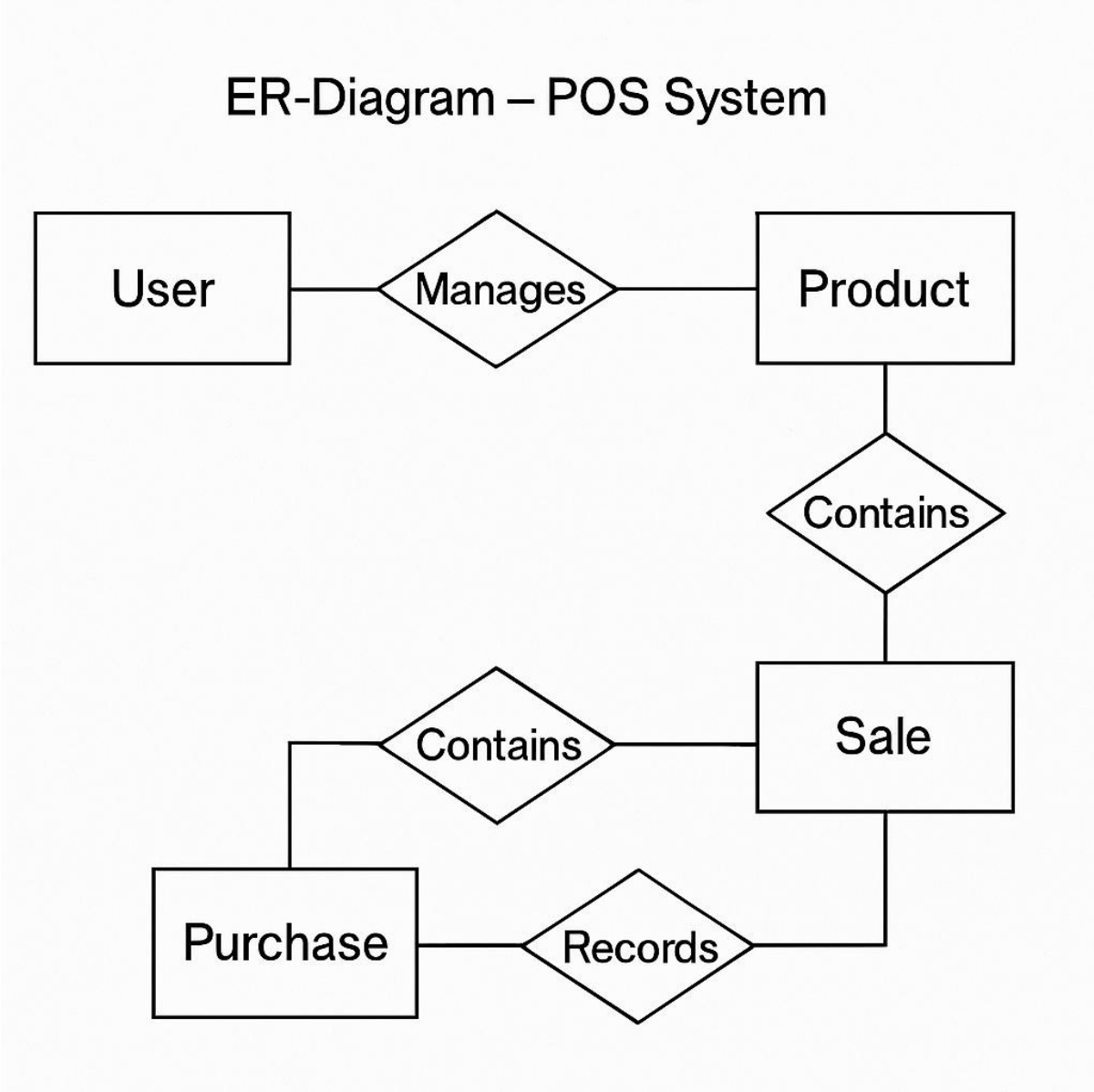


Fig 4.5: ER Diagram – POS System

Entities include:

- Users
- Customers
- Suppliers
- Products
- Sales Invoices
- Purchase Records
- Expense Entries

Each entity is linked via foreign keys and has attributes like `name`, `email`, `amount`, `date`, etc.

CHAPTER 5

SYSTEM DESIGN

System design is the blueprint for building the actual system. It describes the architectural layout, modules, data flow, and user interactions that bring the system to life. Our team followed modular design principles to ensure flexibility, maintainability, and scalability of the Easy Sell Point of Sales System.

UML class diagrams and ER diagrams were created to model the structure of the system database and flow [10]

5.1 System Architecture

The system uses a **Three-Tier Architecture**:

1. Presentation Layer:
   - Built with HTML5, CSS3, and Blade template engine
   - Handles UI, user inputs, and screen layouts
2. Application Layer:
   - Laravel (PHP Framework) handles routing, logic, validation
   - **Contains Controllers, Models, and Middleware**
3. Data Layer:
   - MySQL database stores structured information
   - Laravel Eloquent ORM interacts with DB

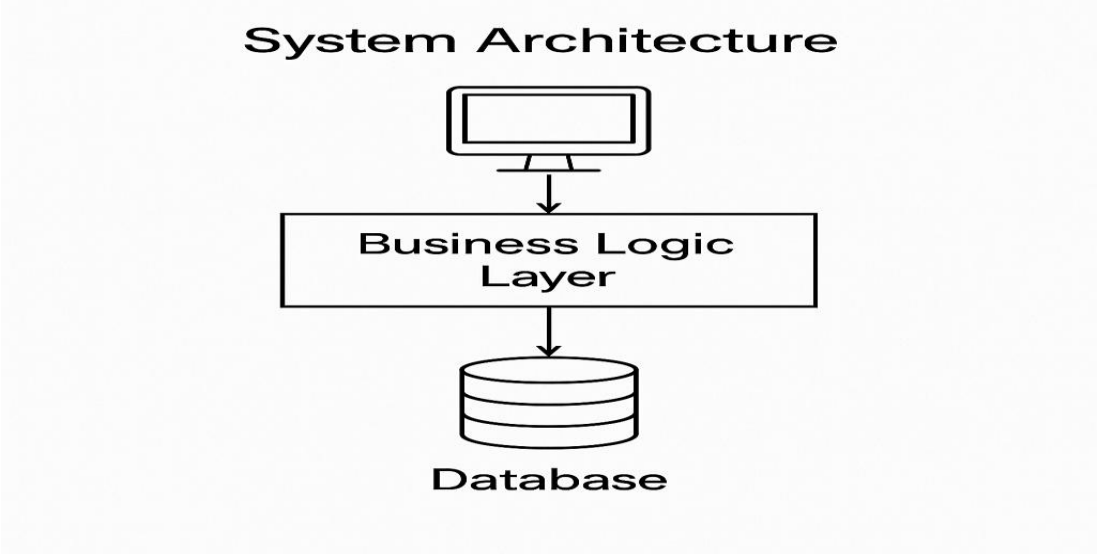


Fig 5.1: System Architecture Diagram

5.2 Module-Wise Design

We divided the system into core functional modules:

5.2.1 Module-Wise Design

- Admin can create users
- Assign specific roles (admin, staff, cashier)
- Manage user access (view, edit, delete)

5.2.2 Product & Inventory Module

- Add/Edit/Delete product details
- Assign SKU, Category, Tax, and Price
- Real-time stock updates after every sale/purchase
- Minimum stock alert notification

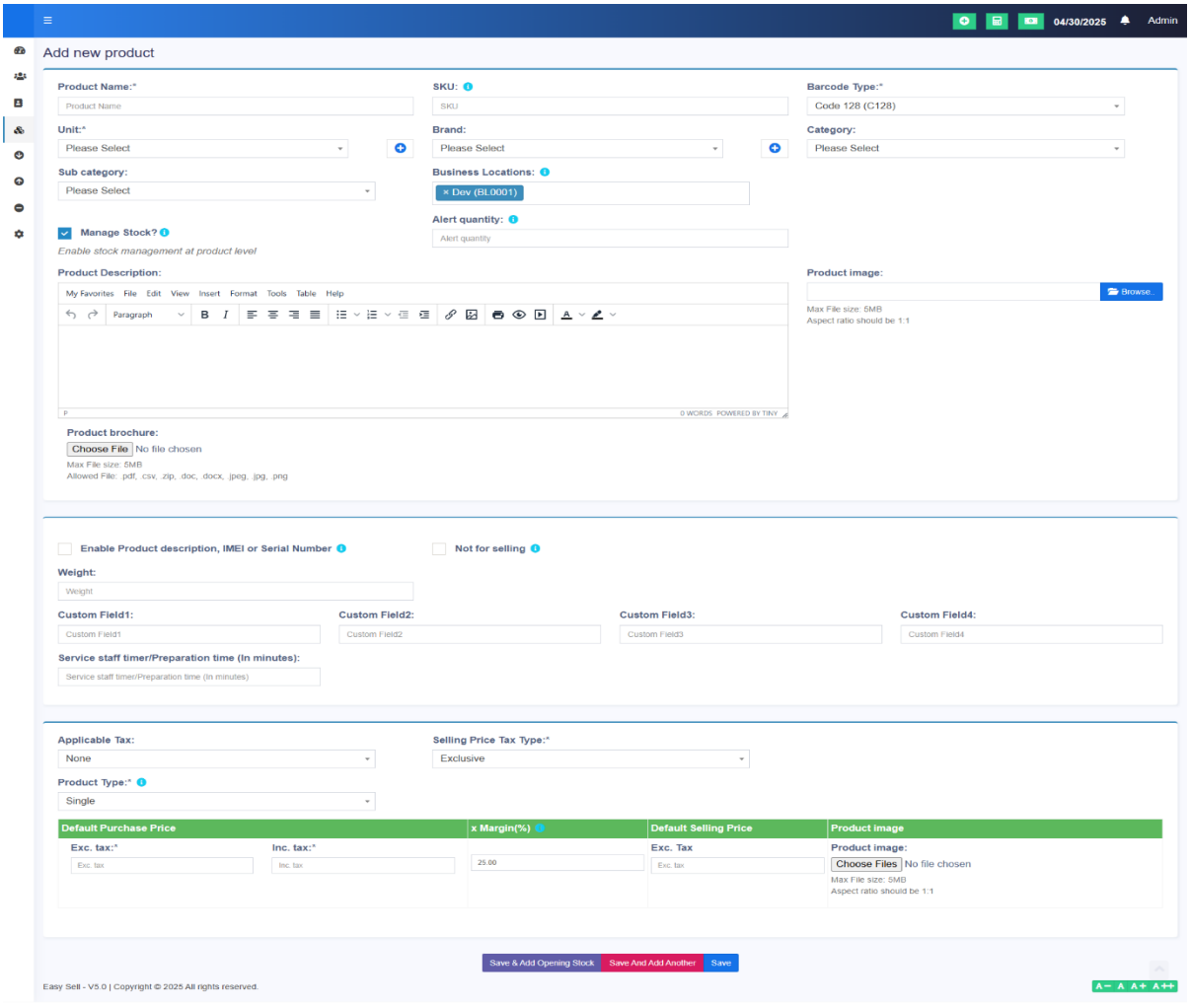


Fig 5.2: Add Product Form UI

5.2.3 Sales & Purchase Module

- Create sales and purchase invoices
- Select product, quantity, apply discount
- Support for multiple payment modes
- Track customer dues and supplier payments

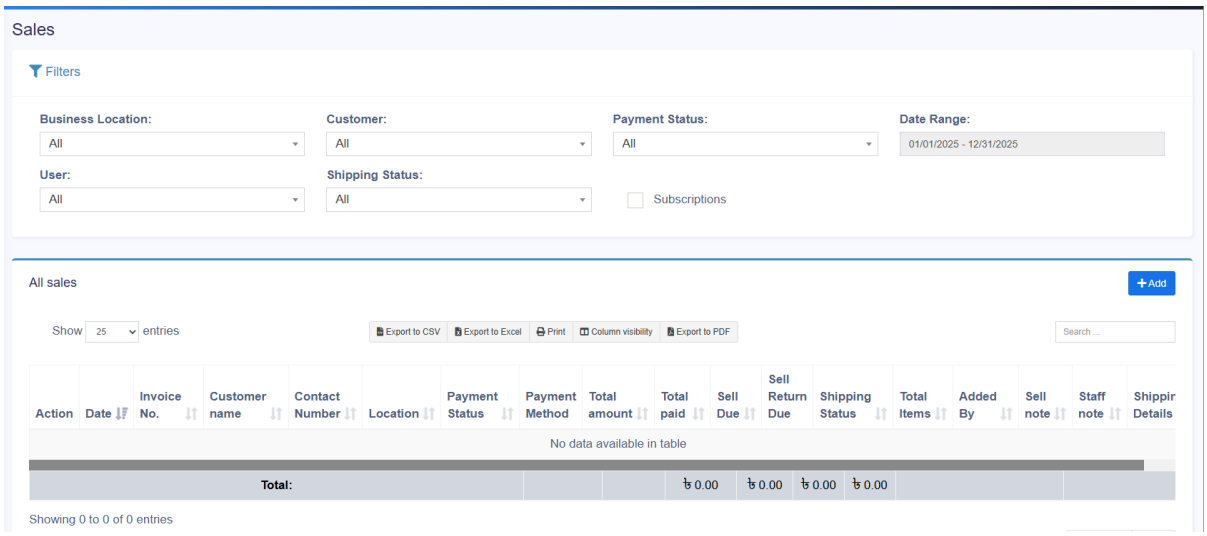


Fig 5.3: Sales Invoice Page

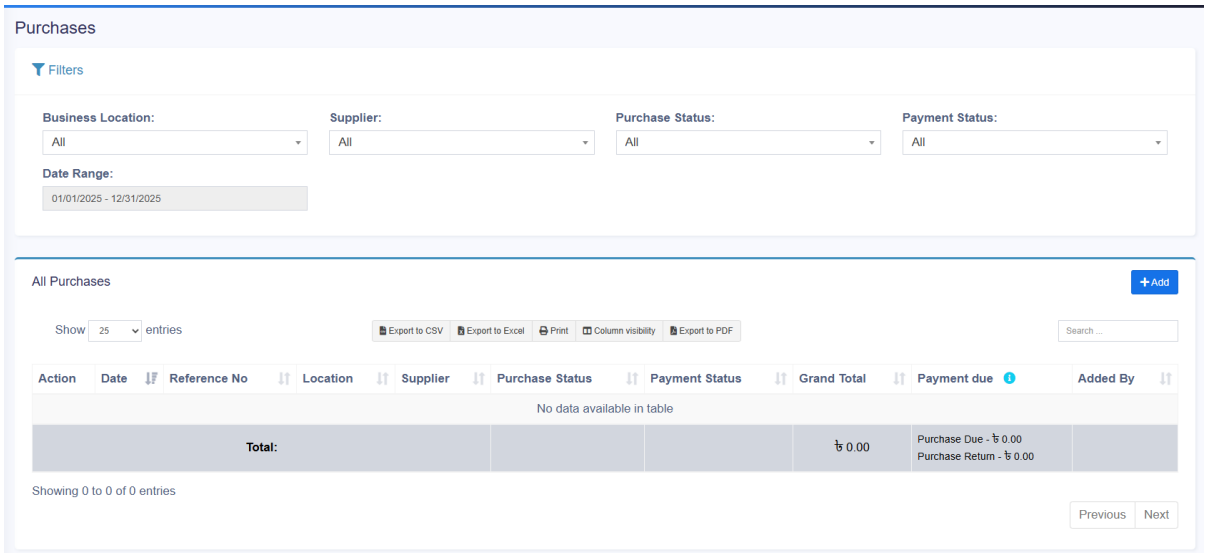


Fig 5.4: Purchase Entry Interface

5.2.4 Expense Management Module

- Add new expense categories (e.g., Rent, Transport)
- Record date-wise expenses
- Filter by category and date range
- Generate printable expense reports

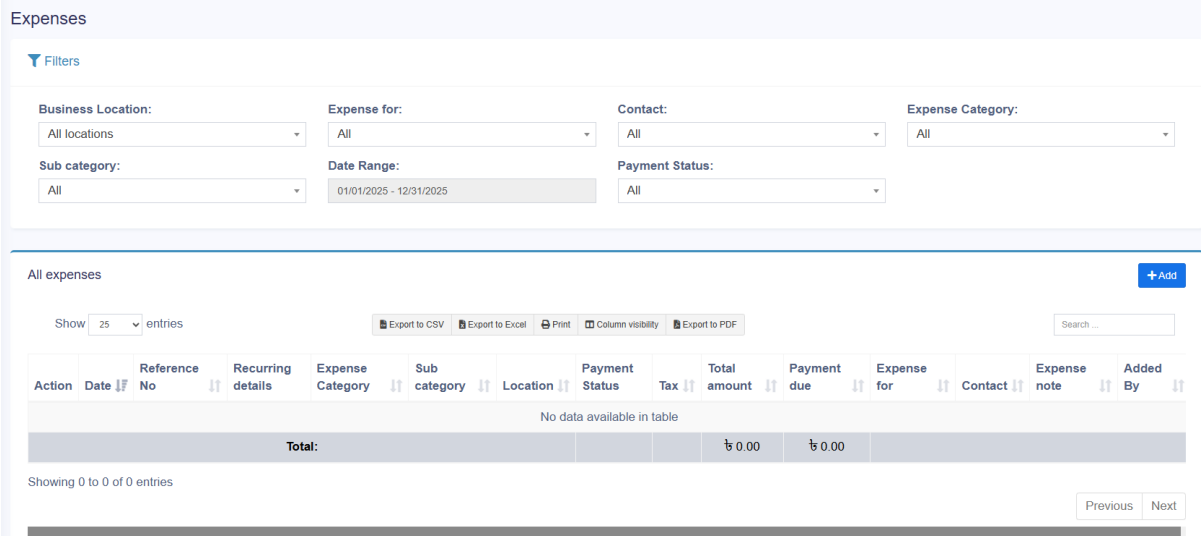


Fig 5.5: Expense Entry Interface

5.3 Data Flow Diagrams

We created both Level 0 and Level 1 Data Flow Diagrams (DFD) to visualize the system's movement of data.

- **Level 0 DFD** shows the interaction between the user and the system
- **Level 1 DFD** breaks down the major modules (e.g., product input, sale processing)

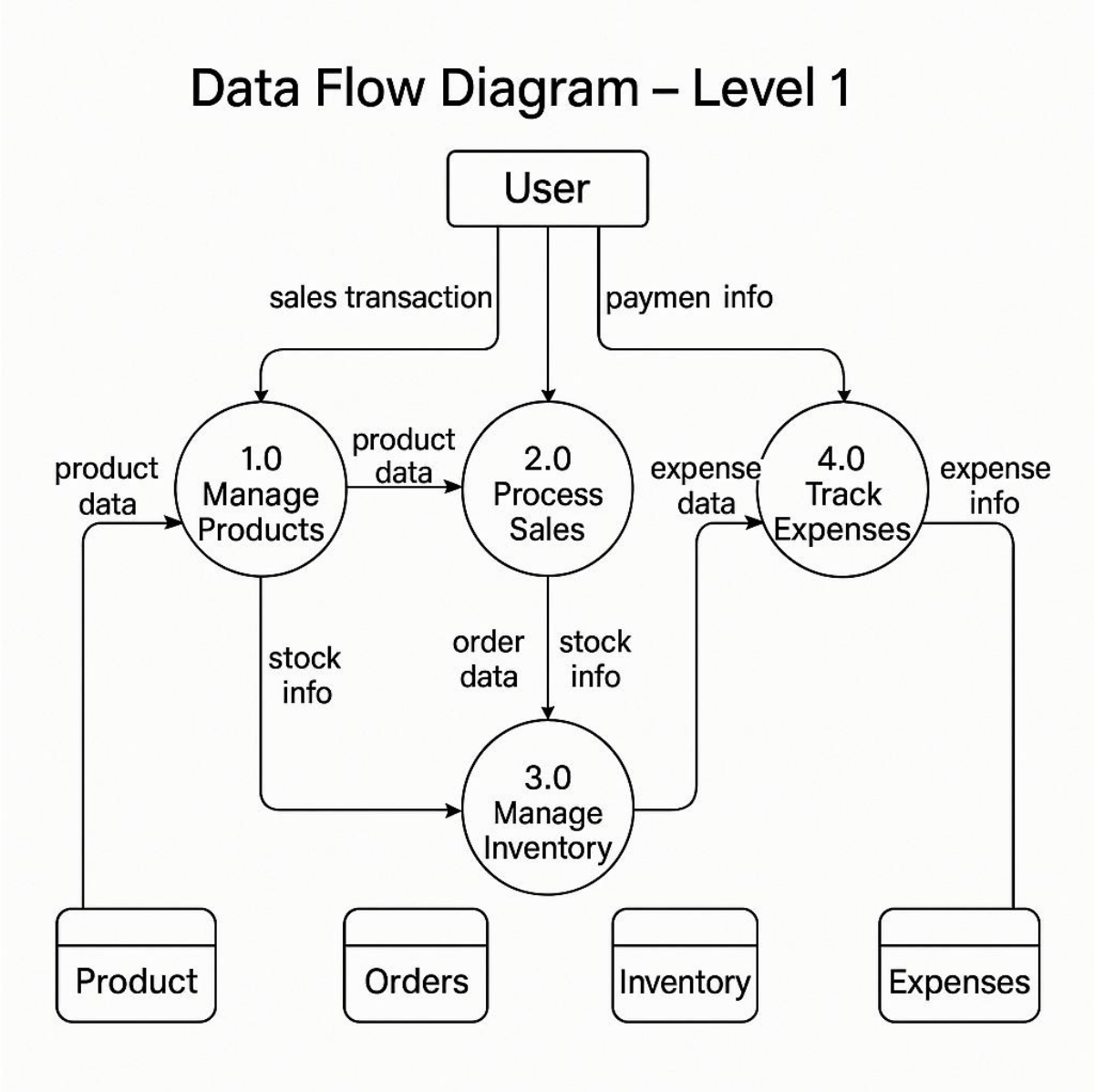


Fig 5.6: Data Flow Diagram – Level 1

All modules are connected to the central dashboard and database layer. The modular design ensures that future updates (like Barcode Scan, SMS Notification, etc.) can be added without disturbing core logic.

CHAPTER 6

IMPLEMENTATION

After completing system design, our team proceeded with implementation based on the Laravel MVC structure.
The implementation phase converts design into working code, with real-time features, database integration, and proper security layers.

6.1 Backend Implementation – Laravel Controllers & Routes

We used **Laravel Framework** (PHP 8.x) as the backend engine. It offers routing, validation, session management, and ORM features. Common Laravel issues such as route errors and session handling bugs were resolved using online community resources [8].

- Controllers were created for each module: ProductController, SalesController, ExpenseController, etc.
- Routes were registered in web.php for user navigation
- Eloquent ORM managed all database operations (CRUD)
- Middleware protected admin routes from unauthorized access

Example:

```
Route::middleware(['auth', 'role:admin'])->group(function () {
   Route::get('/dashboard', [DashboardController::class, 'index']);
});
```

6.2 Frontend Implementation – Blade + HTML/CSS

The frontend was built using:

- **Blade Templates:** For dynamic views and UI consistency
- pages (forms, tables, cards)
- **CSS3:** For visual styling and responsive design
- **JavaScript/JQuery:** For interactivity, validation, filtering, and AJAX

UI Highlights:

- Clean dashboard layout
- Responsive product table
- Date pickers for reports
- Success/error alert popups

6.3 Database Integration

The system uses **MySQL** as the RDBMS.

Key tables:

- **users**
- **products**
- **customers**

- **suppliers**
- **sales**
- **purchases**
- **expenses**
- **payments**

Laravel's **migration & seeding** features were used to create and populate the database.

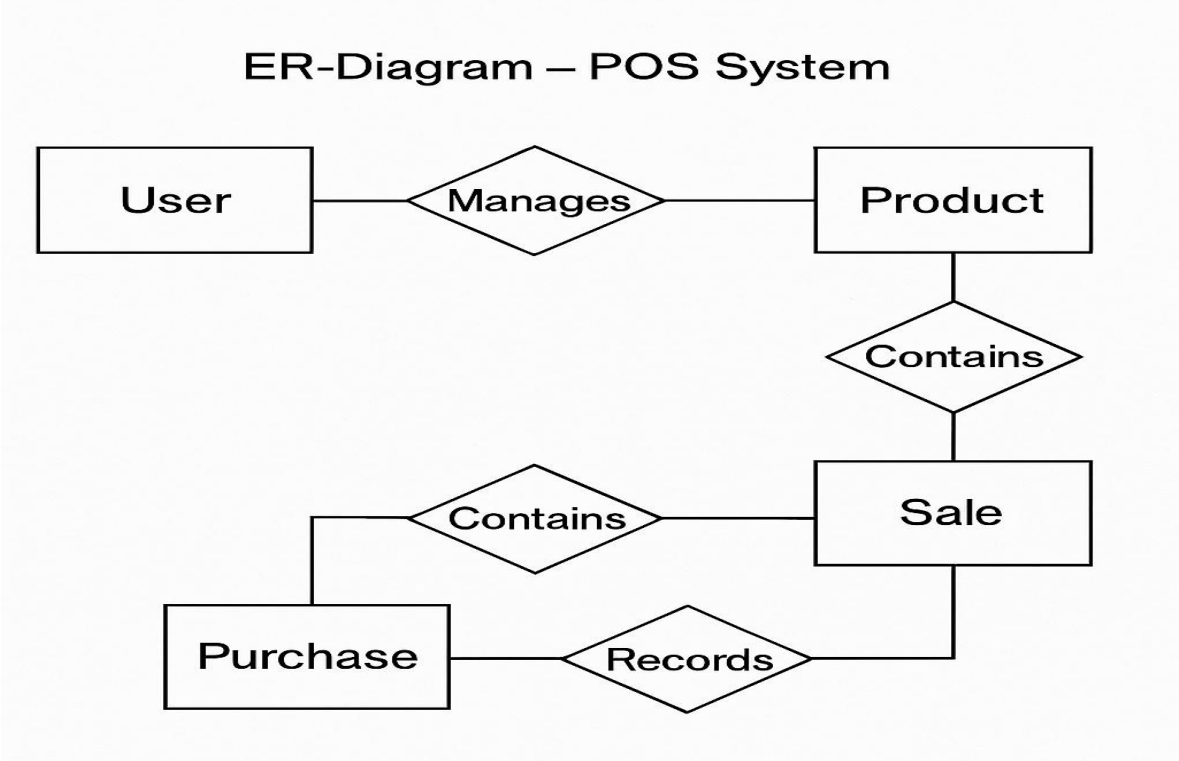


Fig 6.1: ER Diagram (Linked with Chapter 4)

6.4 Role-Based Access Control (RBAC)

The system supports multiple user roles:

- **Admin:** Full access to all modules
- **Staff:** Restricted access (no delete)
- **Cashier:** Can only create sales invoices

Role permissions are managed via middleware and stored in the `roles` table.

Benefits:

- Better security
- Task separation
- Admin-level monitoring

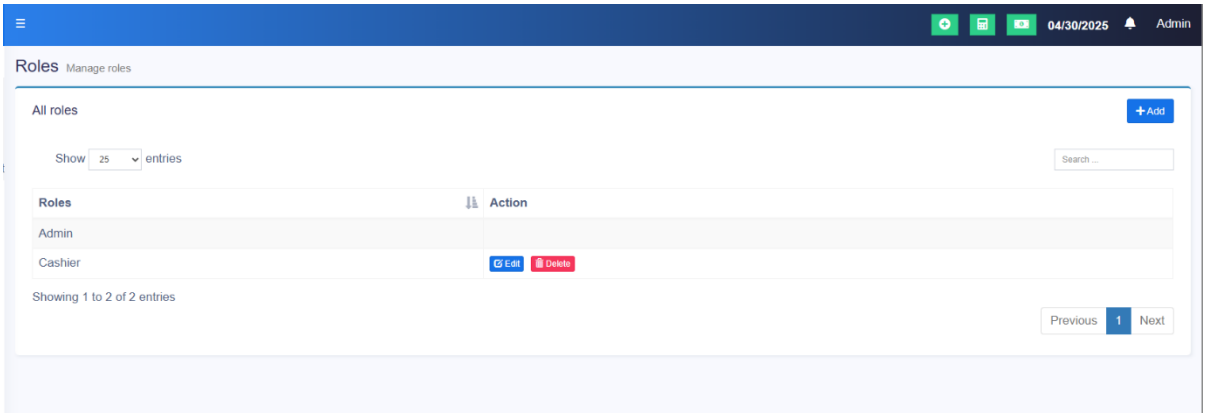


Fig 6.2: User Role and Access Settings Page

6.5 Security Measures

Our system implements key security practices:

- Password hashing using bcrypt()
- CSRF protection via Laravel tokens
- Route-level authentication and permission checks
- Input validation on server and client side
- Restricted file uploads and access points

Security in login, transactions, and data storage was strictly maintained.

CHAPTER 7

TESTING AND RESULTS

Testing is an essential phase of any software development life cycle.
Our team conducted multiple levels of testing to ensure the Easy Sell POS System is stable, secure, and ready for real-world use.

Bug tracking and solutions were often derived from popular developer forums like Stack Overflow [8].

7.1 Testing Strategy

We followed the following testing methodology:

- **Unit Testing:** Testing individual modules
- **Integration Testing:** Testing interactions between modules
- **System Testing:** Complete end-to-end testing
- **User Acceptance Testing (UAT):** Real scenario validation
- Testing involved verifying the structure and consistency of JSON responses from the server [6].

Tools used:

- Manual testing in browser
- Debugging with Laravel logs
- MySQL data validation
- Browser console and error tracking

7.2 Unit Testing and Bug Fixing

Each module (e.g., Sales, Product, Expenses) was tested individually:

- Product creation form validation
- Invoice calculation accuracy
- Stock update after sales/purchase
- Expense filtering by date/category
- Customer due report accuracy

Bug ID	Module	Description	Status	Fixed By
001	Invoice	Total amount incorrect after discount	Fixed	Jakia Sultana
002	Product Stock	Stock not updating after sale	Fixed	Rasel Mia
003	Expenses	Expense not saving properly	Fixed	Saiful
004	Report Filter	Date filter not applying	Fixed	Hasan
005	User Roles	Staff able to access admin page	Fixed	Solaiman

7.1: Bug Log Sheet Screenshot (Text Table Style)

Invoice Output (Text Summary Style):
Test Case: Create Sales Invoice
Steps: Add 2 products → Apply 10% discount → Select cash payment
Expected Result:
- Invoice generated successfully
- Total calculation is accurate
- Stock reduced accordingly
- Printable invoice available
Result: ☑ Passed

We found bugs like:

- Incorrect total after discount
- **Role permission bypass**
- UI layout breaking on mobile view

All bugs were logged and resolved before final deployment.

7.3 Integration and UAT

After unit testing, we tested how modules interact:

- Creating a sale and seeing stock reduction
- Creating a purchase and validating supplier ledger
- Payment entries affecting due balances
- Generating reports after sales/purchases

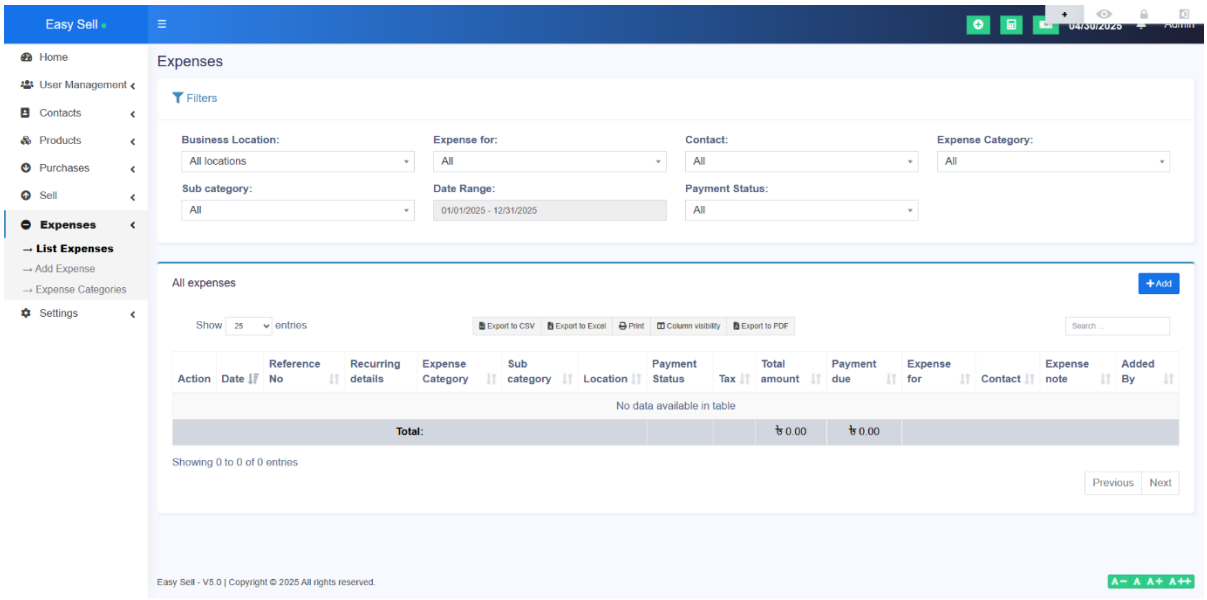
In **User Acceptance Testing**, dummy data was used:

- Admins created products and invoices
- Salespersons used sales module
- Reports were generated and cross-verified with manual calculations

7.4 Screenshots and Output Pages

We captured various output pages as part of testing proof:

- Sales invoice success page
- Payment summary
- Dashboard after transactions
- Filtered reports with proper totals
- Profit & loss graph and tables



7.1: Expense Report Output

Each result was documented with timestamps and validated.

7.5 Project Overview

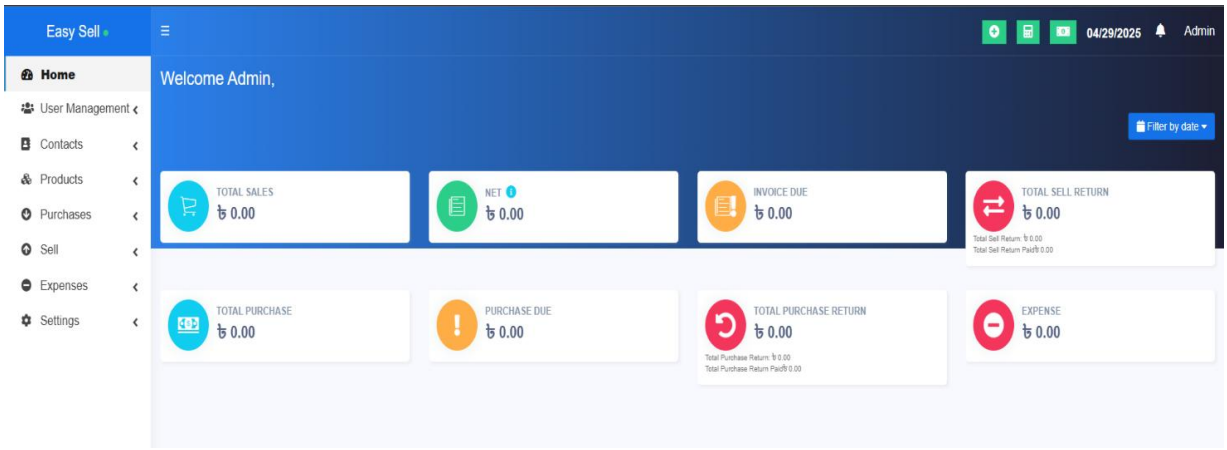


Fig 7.2: Dashboard Overview

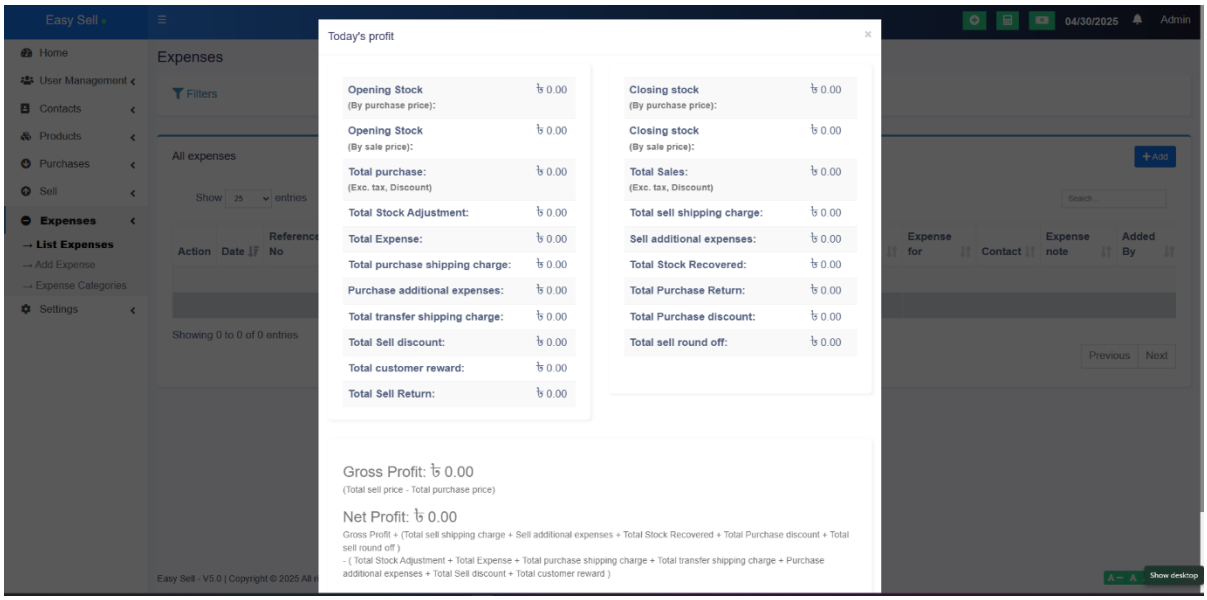


Fig 7.3 Profit Overview

Contacts overview:

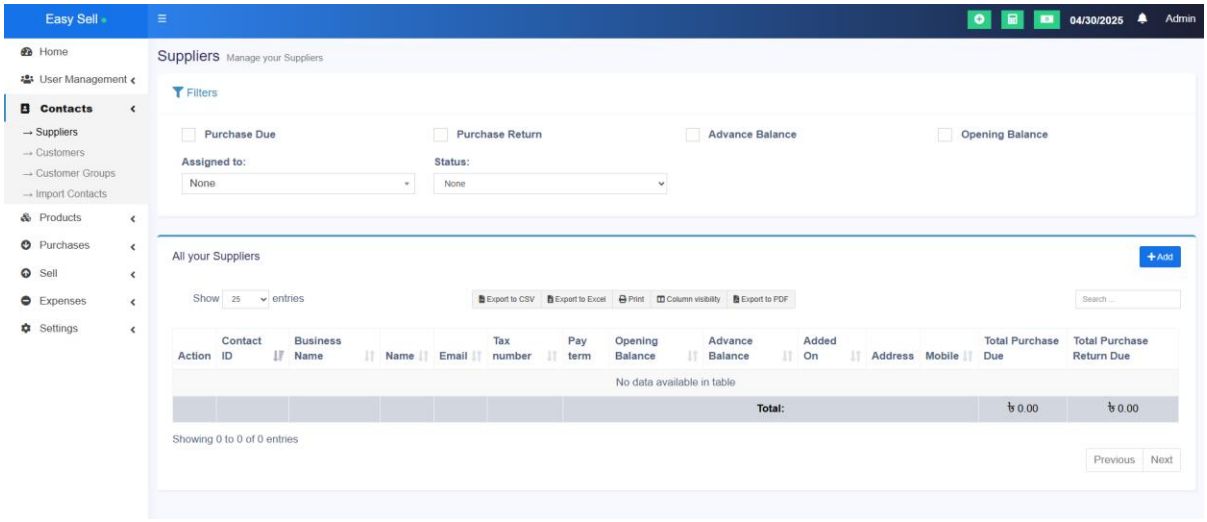


Fig 7.4:Suppliers Overview

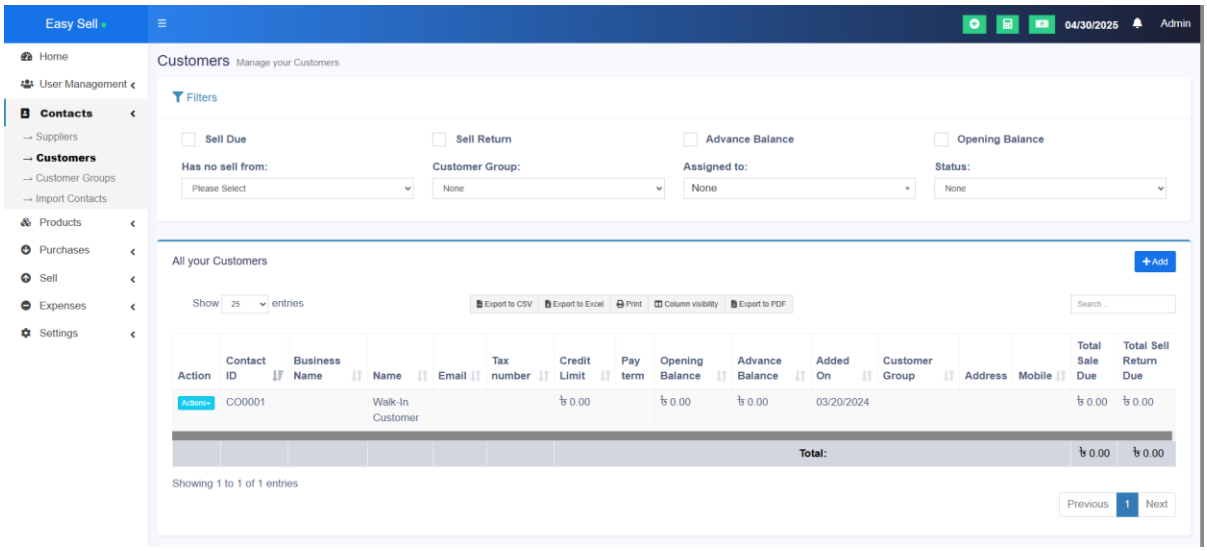


Fig 7.5:Customer Overview

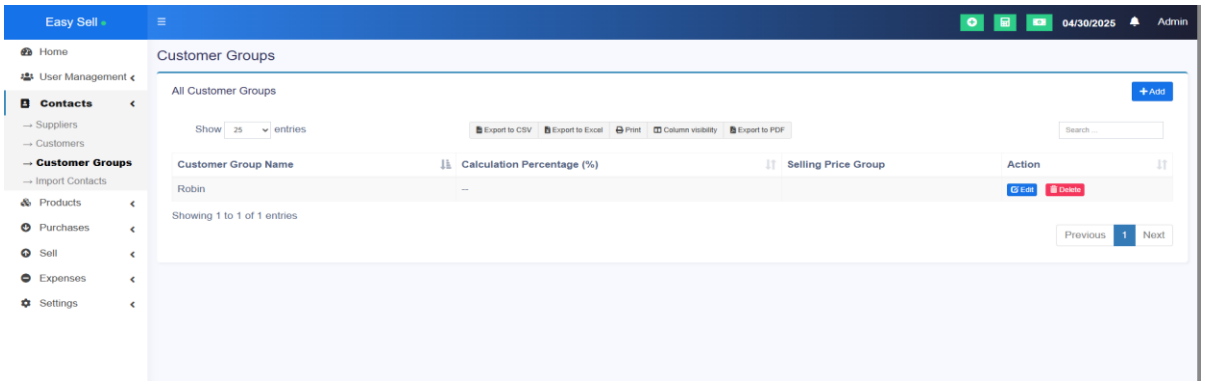


Fig 7.6:Customer group Overview

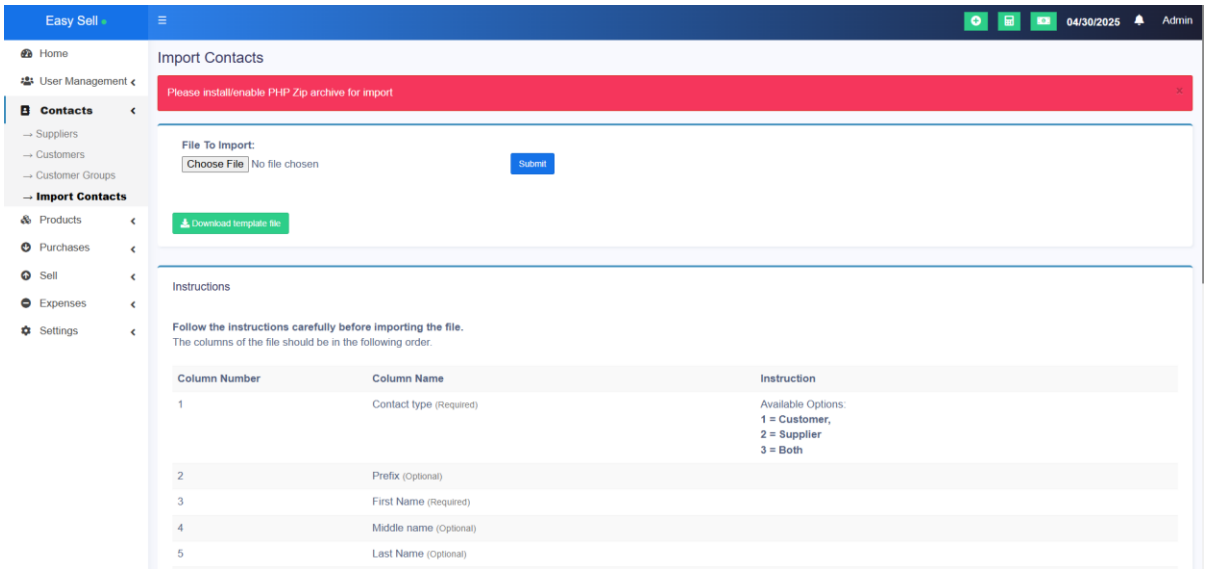


Fig 7.7:Import Contacts Overview

Purchases overview:

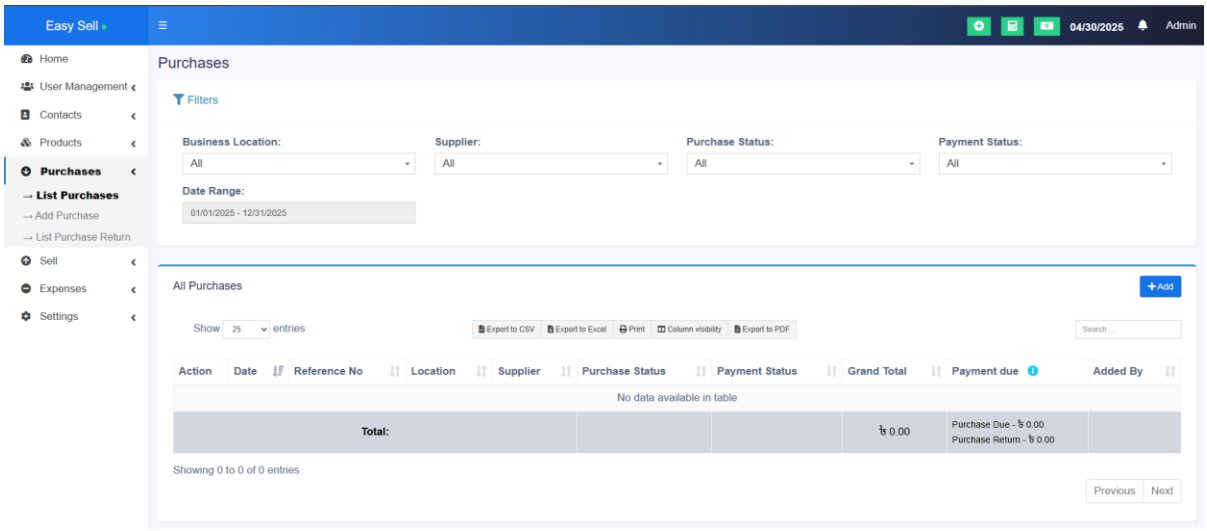


Fig 7.8: List Purchases Overview

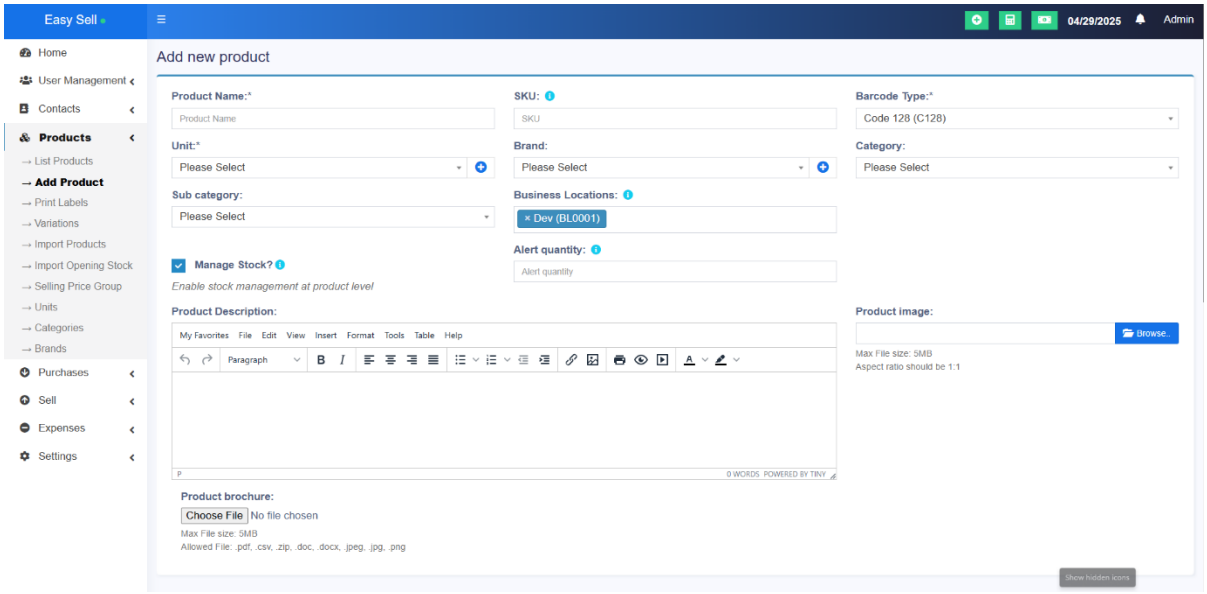


Fig 7.9: Add Product Overview

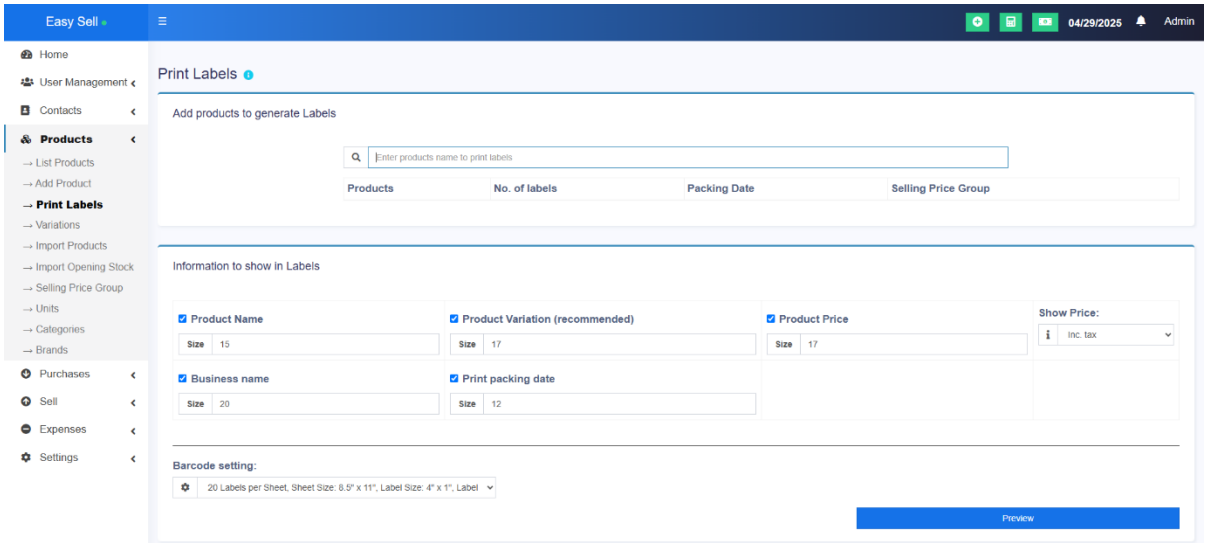


Fig 7.10: Print Labels Overview

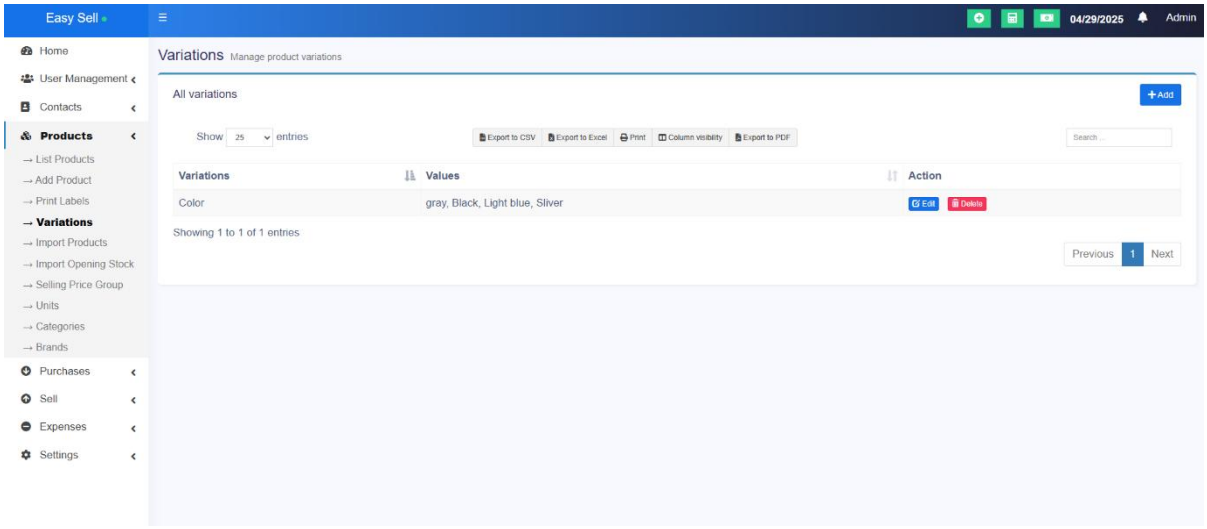


Fig 7.11: Variations Overview

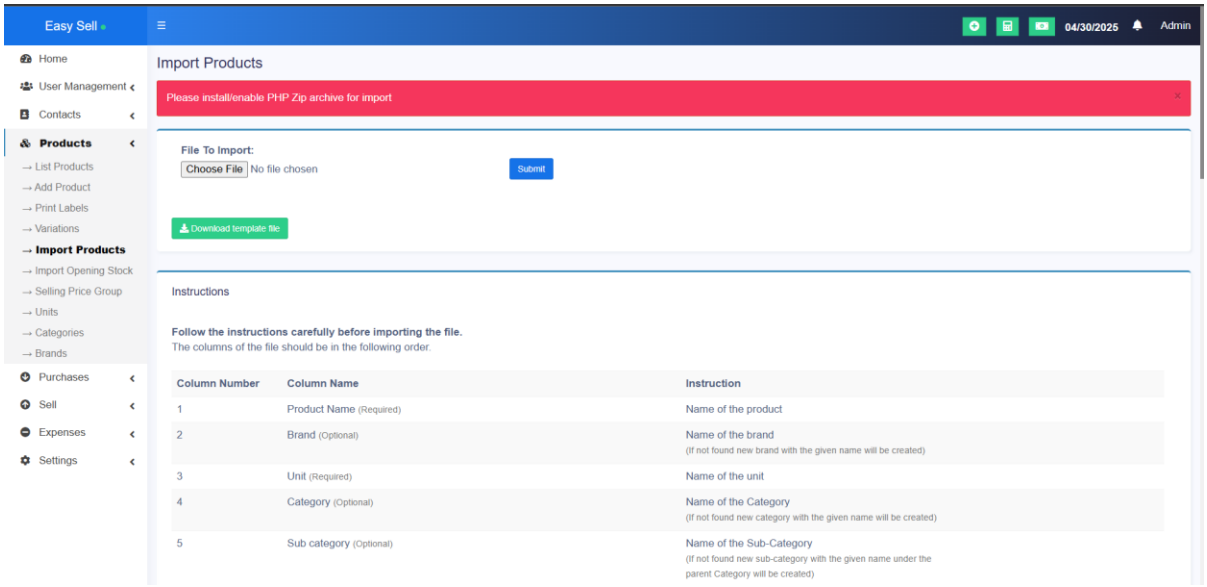


Fig 7.12: Import product Overview

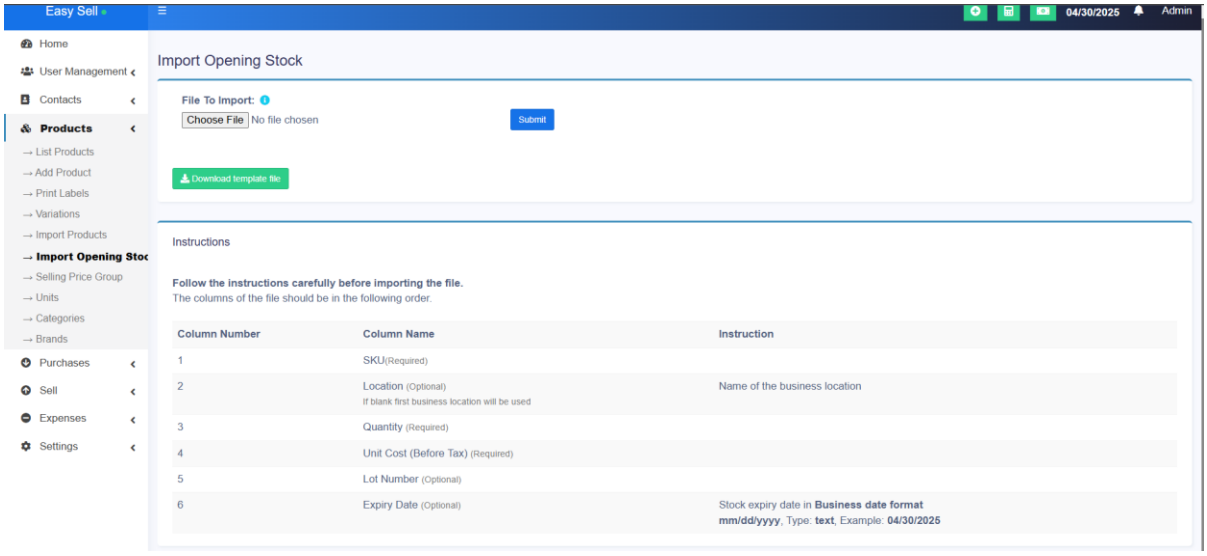


Fig:7.13: Import Opening Stock

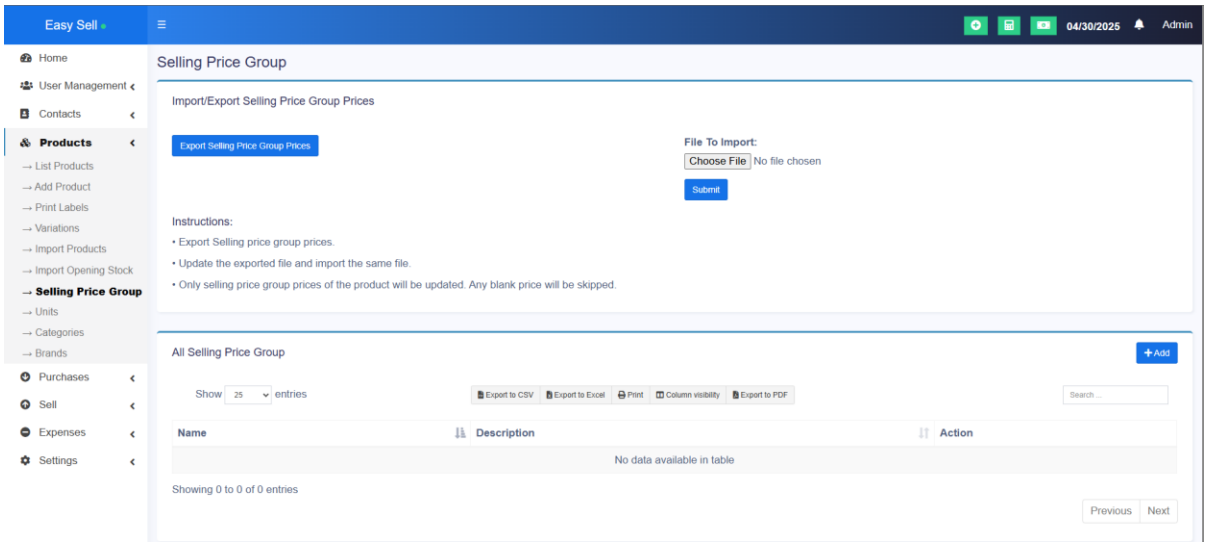


Fig:7.14: Selling Pricee Group

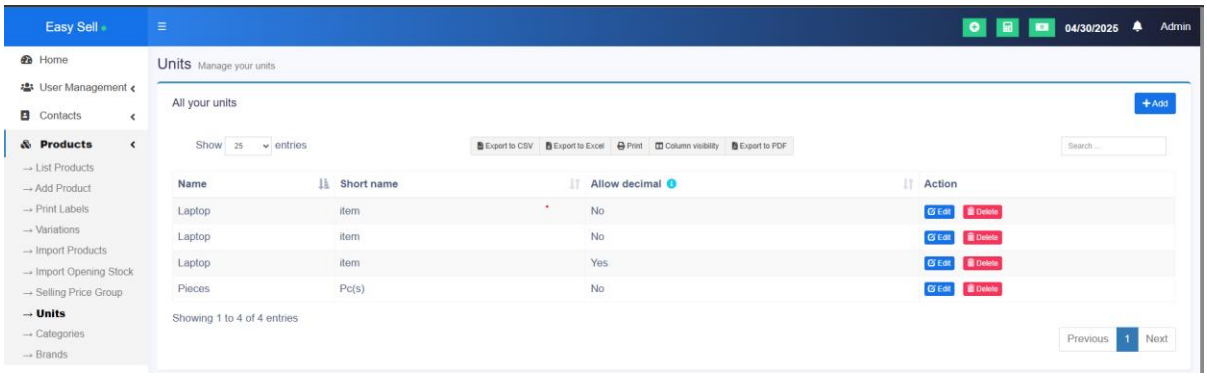


Fig:7.15: Units overview

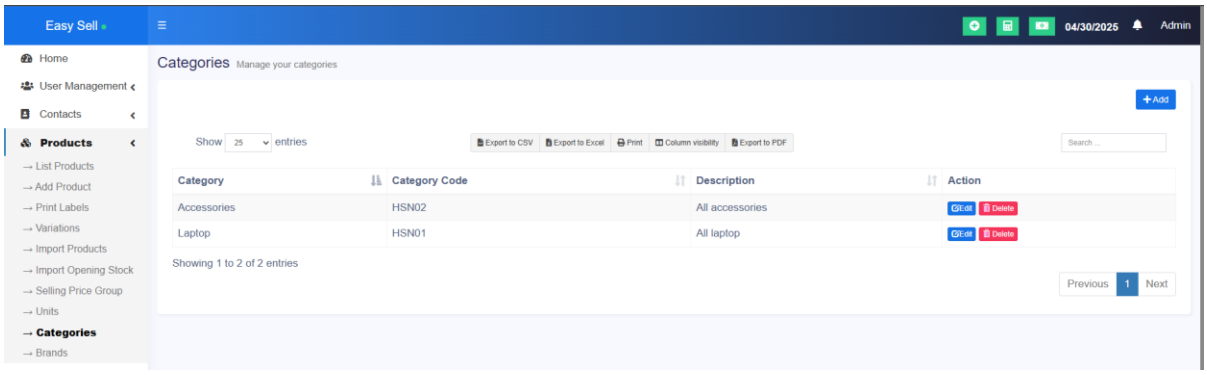


Fig:7.16: Categories overview

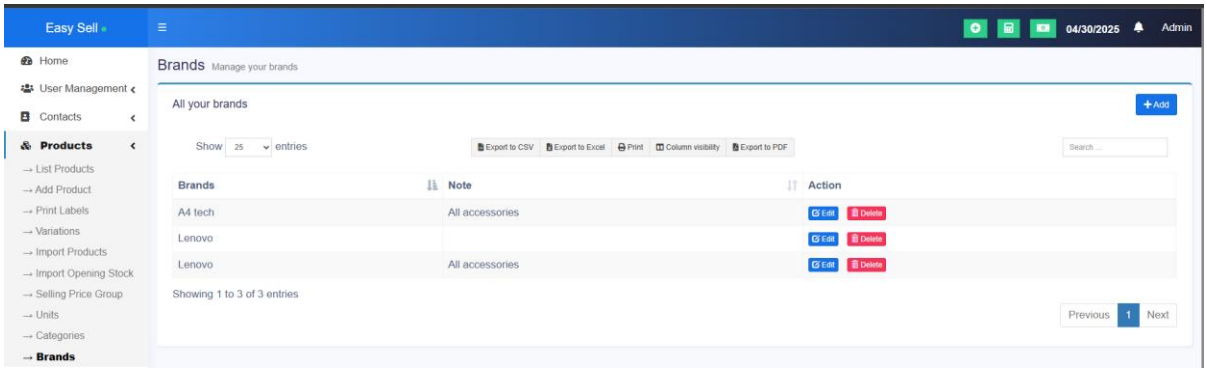


Fig:7.17: Brands overview

Purchases:

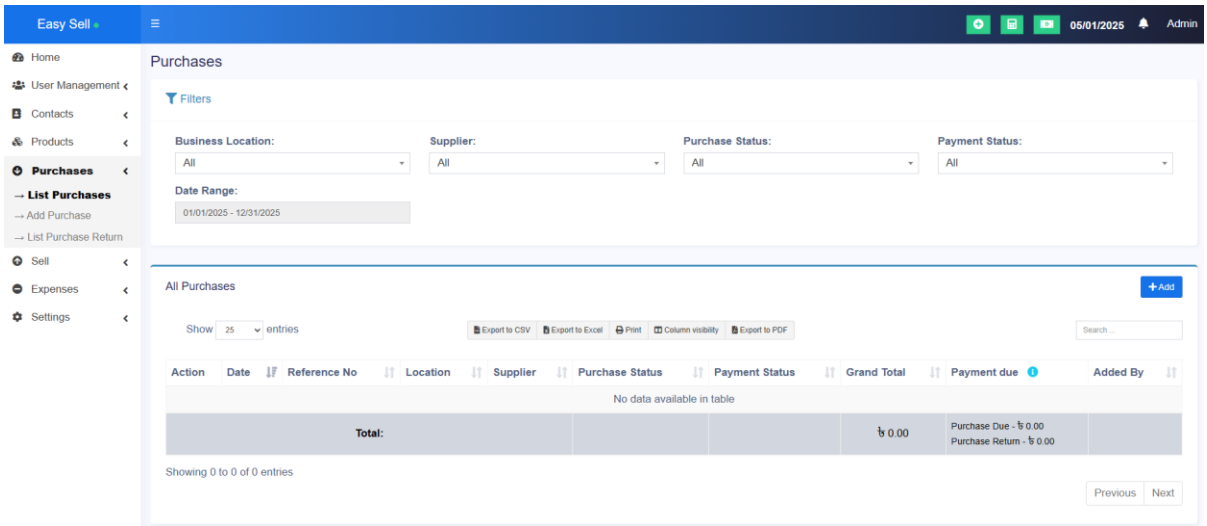


Fig:7.18: List Purchase

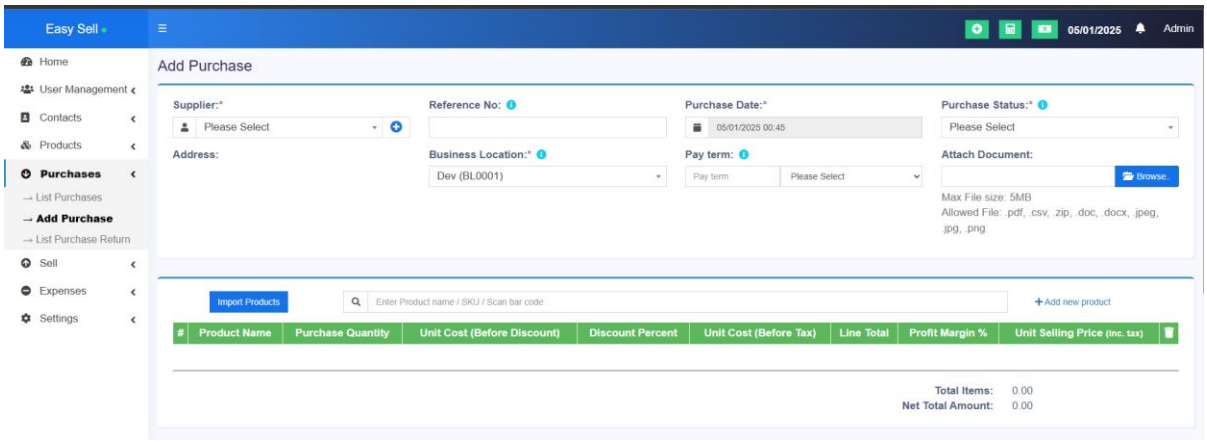


Fig:7.19: Add Purchase

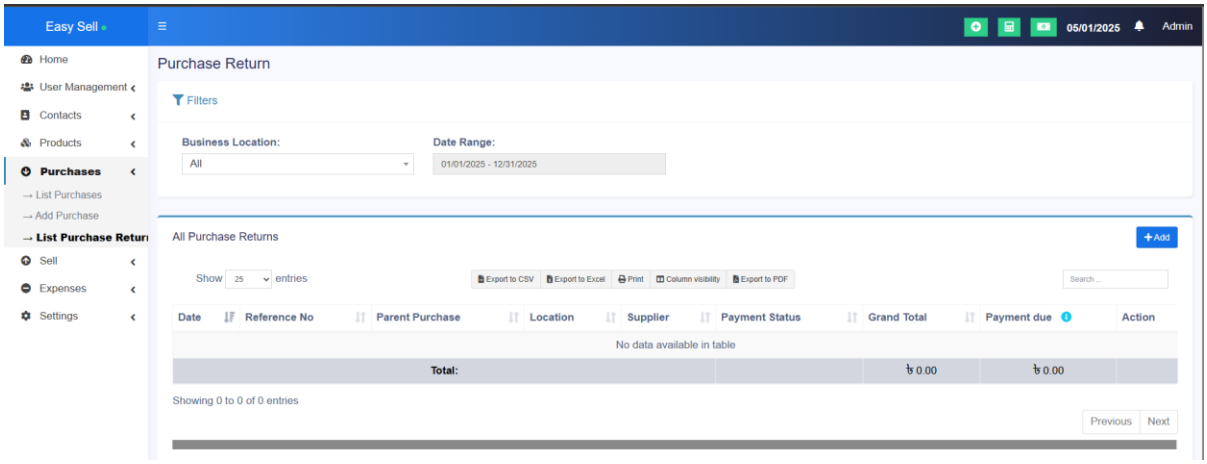


Fig:7.20: Purchase Return

Sales:

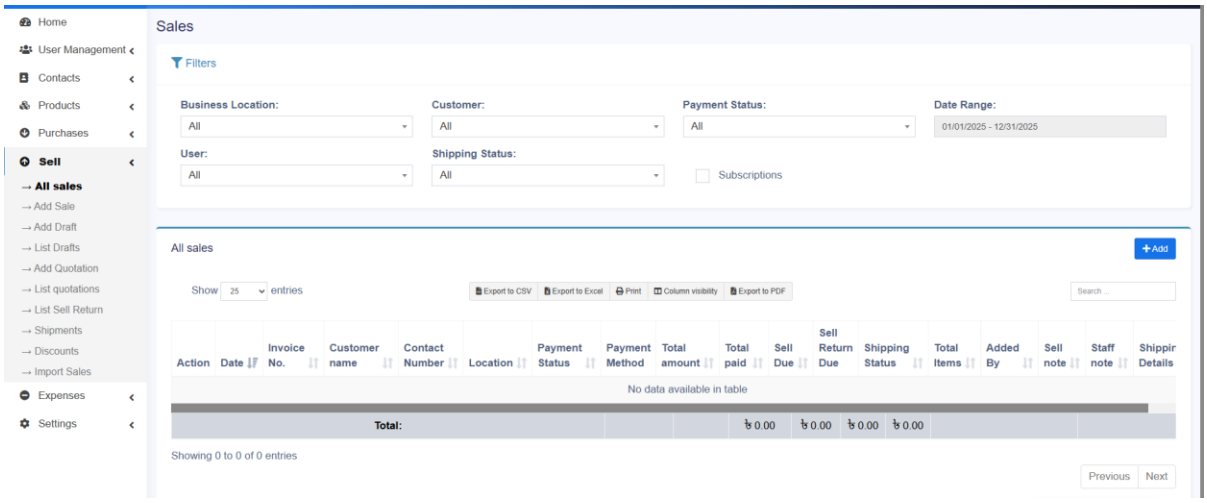


Fig:7.21: Sales Overview

Expenses:

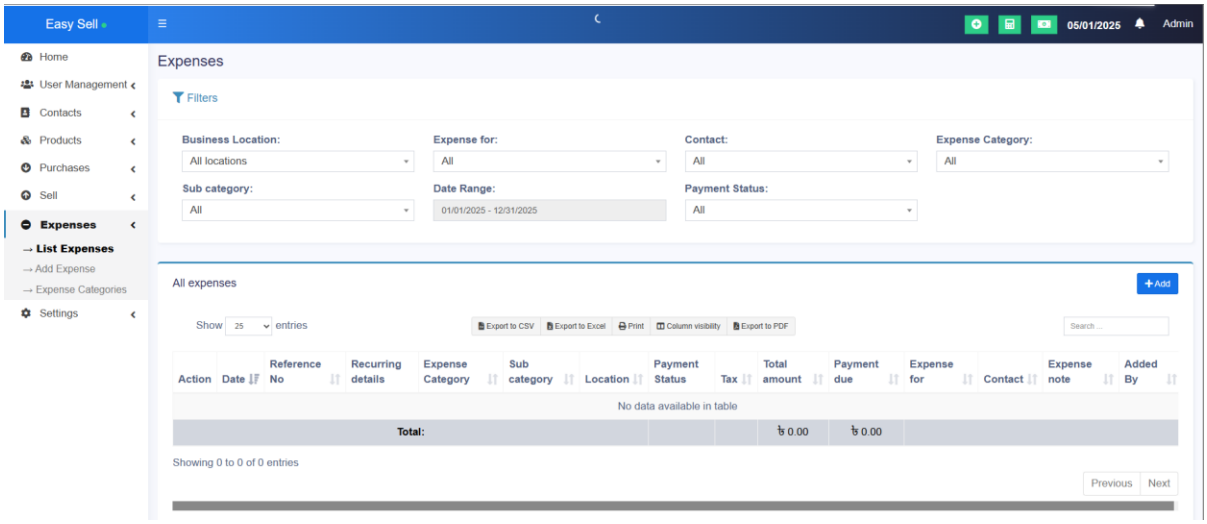


Fig:7.22: Expenses Overview

Settings:

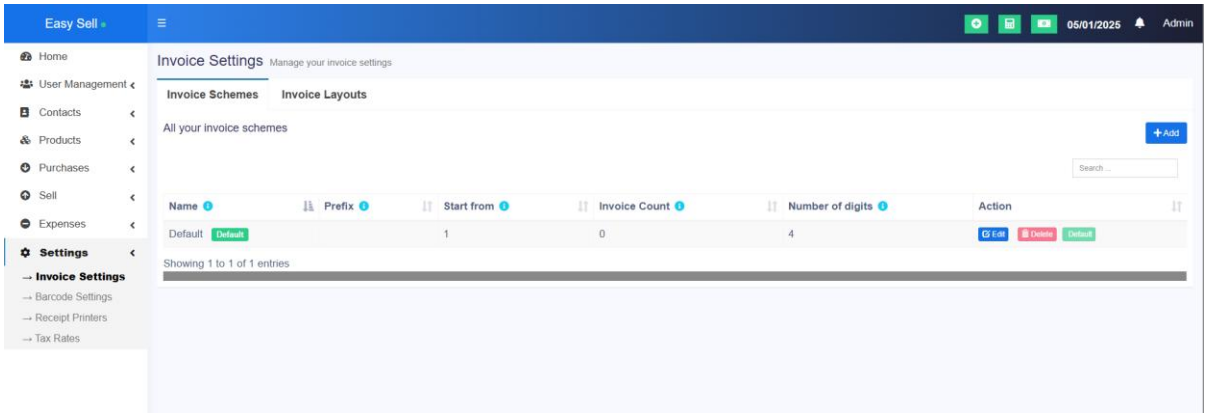


Fig:7.23: Invoice Setting Overview

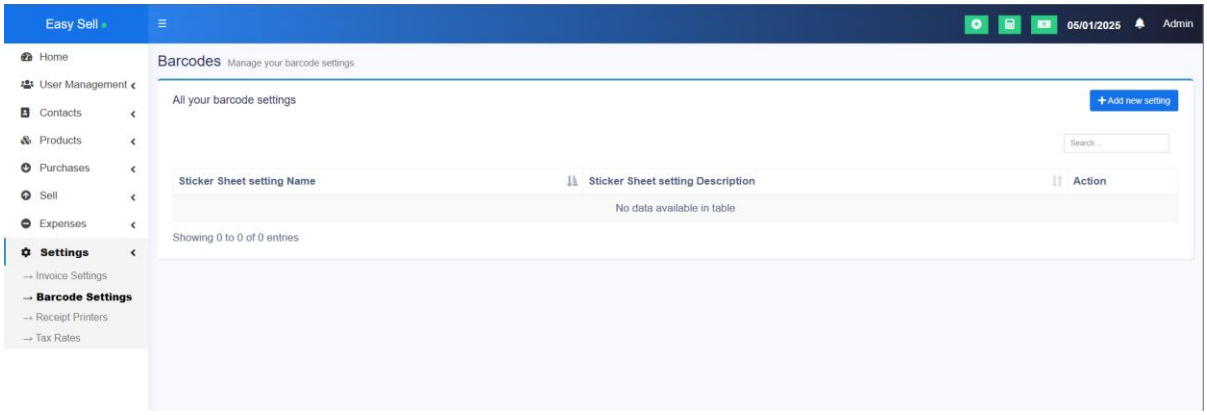


Fig:7.24: Barcodes Setting Overview

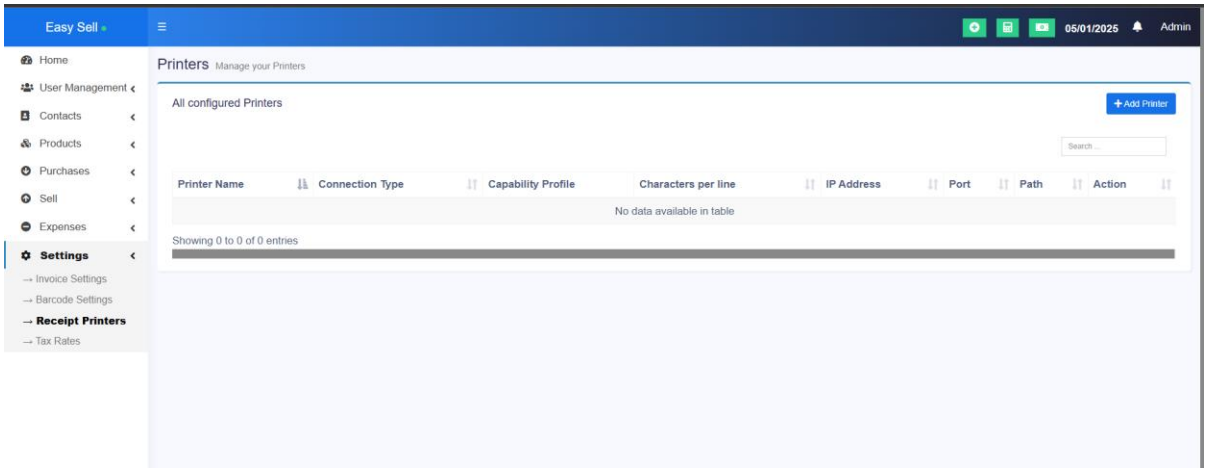


Fig:7.25: Pinters Setting Overview

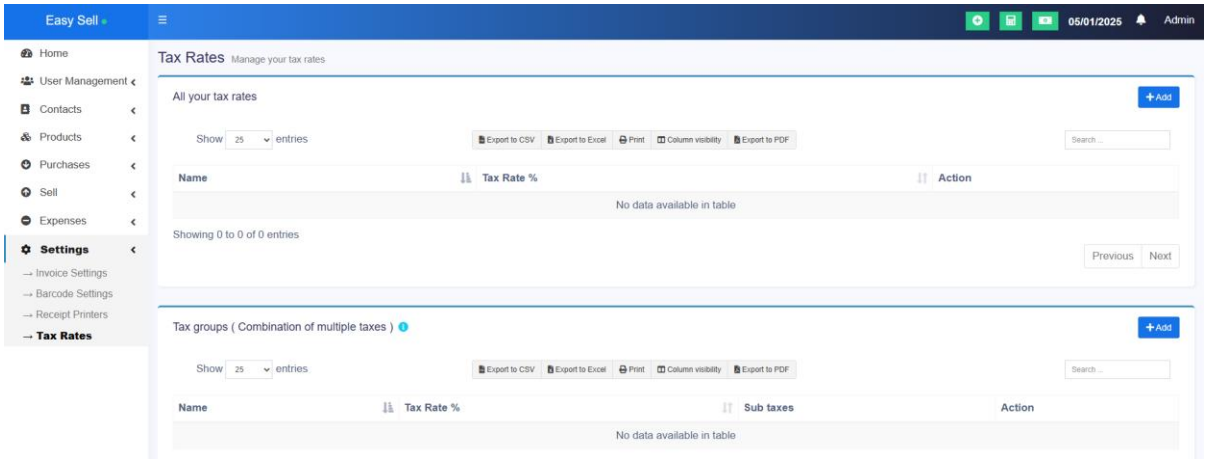


Fig:7.26: Tax Rates Setting Overview

CHAPTER 8

CONCLUSION

8.1 Conclusion

After several weeks of research, planning, development, and testing, our team successfully completed the Easy Sell POS System. This project allowed us to turn a real-world problem into a functioning solution that can help small and medium businesses manage their sales, purchases, inventory, expenses, and reports from a single, user-friendly platform. Through this system, we were able to reduce manual errors, save time, and make business processes more organized and efficient. It was a complete team effort where we collaborated on designing the system, writing the code using Laravel and MySQL, creating user interfaces with HTML and CSS, and validating features through testing. We also learned how to handle data securely, implement role-based access control, and generate real-time reports that provide clear insights for business decision-making.

8.2 Limitations of the System

Even though our system works properly and meets all main objectives, there are still a few things it cannot do right now:

- It works only for one shop or branch at a time
- It does not have a mobile app (Android/iOS)
- It does not send SMS or email notifications for invoices or dues
- It does not support barcode scanning
- It requires internet connection or LAN to work — offline mode is not supported yet

These are not big problems, but they can be improved in future versions.

8.3 Future Enhancements

In the future, we plan to add more features to make this system even better.
Some of the ideas we have are:

- Make it work for multiple branches or warehouses
- Develop a mobile app version so users can check reports from phone
- Add barcode scanning for fast product entry
- Add SMS/email notifications for invoices, due alerts, or low stock
- Add cloud backup and restore to keep data safe
- Add export options for PDF or Excel reports

These improvements will make our Easy Sell POS System more powerful and helpful for more businesses.

REFERENCES

[1] Otwell, T. (2020). Laravel: The PHP Framework for Web Artisans. [Online]. Available: https://laravel.com/docs

[2] W3Schools. (2023). HTML5 Tutorial. [Online]. Available: https://www.w3schools.com/html/

[3] W3Schools. (2003) CSS3 Tutorial. [Online] Available: https://www.w3schools.com/css/

[4] Crockford, D. (2008). JavaScript: The Good Parts. O'Reilly Media, Inc

[5] Letkowski, J. (2015). Doing database design with MySQL. Journal of Technology Research, 6, p.1.

[6] Lv, T., Yan, P. & He, W. (2018). Survey on JSON data modelling. Journal of Physics: Conference Series, 1069(1), p.012101.

[7] Boshernitsan, M., & Downes, M.S. (2004). Visual programming languages: A survey. Computer Science Division, University of California.

[8] Stack Overflow Community. (2024). Common Laravel Issues and Solutions. [Online Forum]. Available: https://stackoverflow.com/questions/tagged/laravel

[9] Bhattacharyya, D.K. (2006). Research Methodology. Excel Books India

[10] Ravi, S.Y., Jindal, R., & Anand, S. (2020). A Study on Comparison of UML and ER Diagram. International Research Journal of Engineering and Technology (IRJET), 7(5), pp.7575–7580.