POS Application Using C# .NET, WPF, MySQL, and MVVM Pattern

INTRODUCTION

The Point of Sale (POS) system is a user-friendly application designed to facilitate the efficient handling of sales transactions and inventory management for small to medium-sized retail businesses. This report outlines the development of a Point of Sale (POS) application tailored to accommodate the needs of today's retail operations. The POS system enables users to input product information, including cost and selling prices, and facilitates the recording of sales transactions during business operations. At the end of sales day, the application provides a summary of all transactions and calculates the total profit, delivering data to the business owner.

PROBLEM STATEMENT

Simple POS application.
User enters list of products and its cost and selling price during a sale user selects a product and enters quantity
when the user wants to look at the all sales at end of the say, should display all the sales and display profit made by the user.
We expect you to develop the application using C# .NET, WPF, MySQL and
follow the MVVM pattern.

TECHNOLOGY USED

Below is a comprehensive breakdown of the technologies employed in the project.

- ***C# .NET***

  C# is a object-oriented programming language developed by Microsoft, which provides a robust and secure framework for building applications. In the context of this POS application, C# serves as the backbone of the software, handling the business logic, including the management of product entries, processing of sales, and calculations of profits. The .NET framework offers a comprehensive class library and runtime environment that ensures the application runs efficiently and reliably.

- ***Windows Presentation Foundation (WPF)***

WPF is a UI framework for creating visually rich Windows desktop applications. For this POS system, WPF enabled us to design a user-friendly and responsive interface. It separates the UI from the business logic, allowing designers and developers to work concurrently. WPF's binding and templating features were particularly instrumental in presenting product and sales data dynamically.

- ***MySQL***

  MySQL is an open-source relational database management system that is known for its speed, reliability, and ease of use. Within this POS system, MySQL acts as the data repository, storing all the details of products, sales transactions, and profit calculations. It efficiently handles queries, updates, and retrieval of data, which is essential for the sales and inventory management features of the application.

DEVELOPMENT ENVIRONMENTS

- ***Visual Studio***

Visual Studio is an Integrated Development Environment (IDE) from Microsoft, supporting a multitude of programming languages and frameworks, including C# and .NET, which were central to this project. It offers powerful tools for developing, debugging, and compiling code, and has a vast ecosystem of extensions for enhanced functionality. For this POS application, Visual Studio was used to write and manage the C# codebase, design the WPF-based user interface, and coordinate version control. Its robust debugging tools were crucial for ensuring the reliability and performance of the application.

- ***MySQL Workbench***

MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. It provides data modelling, SQL development, and comprehensive administration tools for server configuration, user administration, and much more. In the development of this POS system, MySQL Workbench was employed to design, model, create, and manage the database. It allowed for the visualization of database structures and facilitated the writing, running, and optimization of SQL queries.

SYSTEM DESIGN

At its core, the POS system captures and retains product information, processes sales transactions, and computes profits. To achieve these functionalities, the design incorporates a modular approach consisting of several interconnected components, including the product entry module, sales transaction module, and reporting module.

ARCHITECTURE

The POS system is built on the Model-View-ViewModel (MVVM) architectural pattern, which promotes a clean separation of concerns and enhances maintainability and testability.

Model: This layer represents the data and business logic of the application. It includes classes for products and sales that encapsulate all the necessary properties and methods needed to handle data processing and business rules.

View: The view consists of the user interface designed with Windows Presentation Foundation (WPF). It presents the data to the user and captures user inputs with a user-centric layout, including text boxes for input, buttons for commands, and other WPF controls arranged in an intuitive layout.

ViewModel: Serving as the intermediary between the View and the Model, the ViewModel houses the presentation logic and binds the data from the Model to the View. It responds to commands and handles the complex logic of interfacing, ensuring that the user interface remains responsive and decoupled from the data layer.

The application uses a MySQL database as its persistent storage mechanism, with a ***DataAccess class*** facilitating communication between the database and the application logic. This class manages all database-related operations such as queries, updates, and transactions.

All interaction with the database is initiated by the ViewModel, which processes user actions, manipulates the data model, and updates the view with the latest information. Changes in the view are propagated back to the model through data binding, ensuring synchronization across the system.

IMPLEMENTATION

Database Setup and Integration

MySQL Database: A MySQL database was established to store product information and sales transactions. Tables for 'Products' and 'Sales' were created, with fields corresponding to product details and sales records, respectively. The database schema was designed to support efficient querying, updates, and scalability.

DataAccess Layer: The DataAccess layer in C# was implemented to manage all interactions with the MySQL database. It employed the ADO.NET framework for database connectivity, encapsulating all the SQL queries and operations.

Application Development

Models: Defined within the C# .NET environment, models for 'Product' and 'Sale' were created to mirror the database schema and serve as data structures throughout the application.

Views: The Views were constructed using WPF, with a focus on creating a responsive and intuitive user interface. Controls such as TextBoxes for data entry, Buttons for submitting data, and ListBoxes for displaying sales records were arranged in a user-friendly layout.

ViewModels: ViewModel classes were created to handle presentation logic and data binding. The MainViewModel class was the central component orchestrating the Views and Models, handling commands, and ensuring that user actions triggered the appropriate responses in the application.

Business Logic

Command Handling: The implementation of ICommand interface in the form of RelayCommand allowed the application to handle user actions such as adding a product or recording a sale. This decoupled the UI from the business logic, following the MVVM pattern principles.

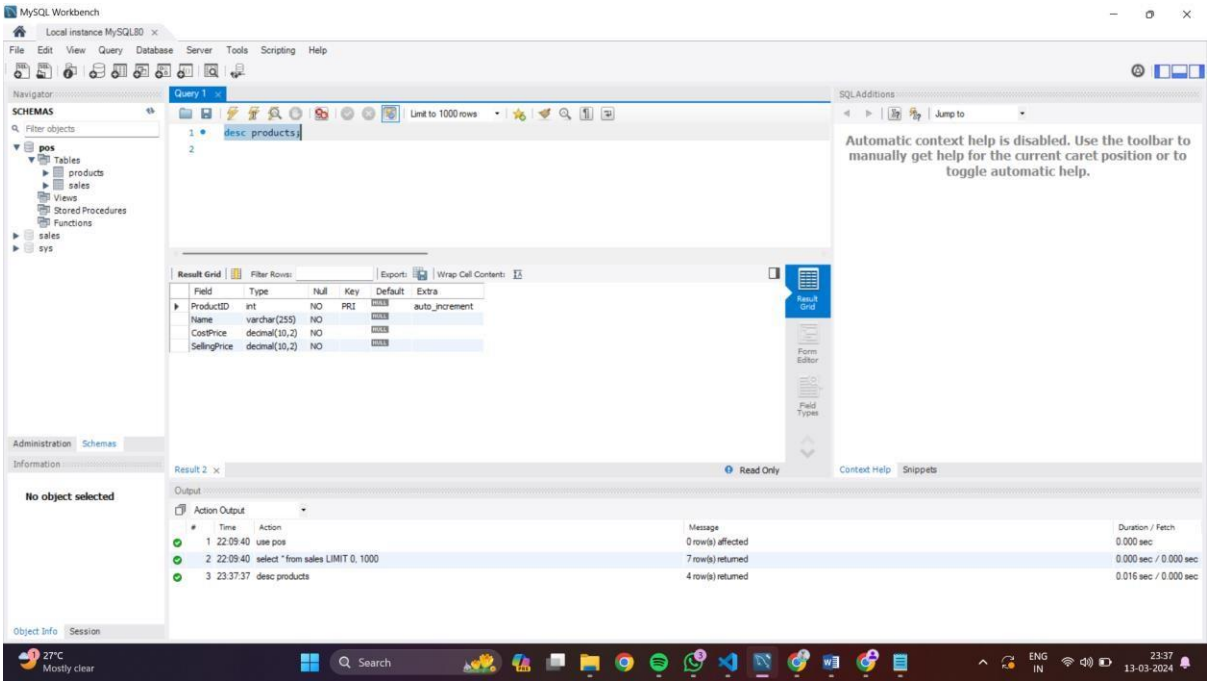
Profit Calculation: Logic to calculate profit was embedded within the ViewModel. Each sale transaction involved computing the profit by subtracting the cost price from the selling price and multiplying by the quantity sold.

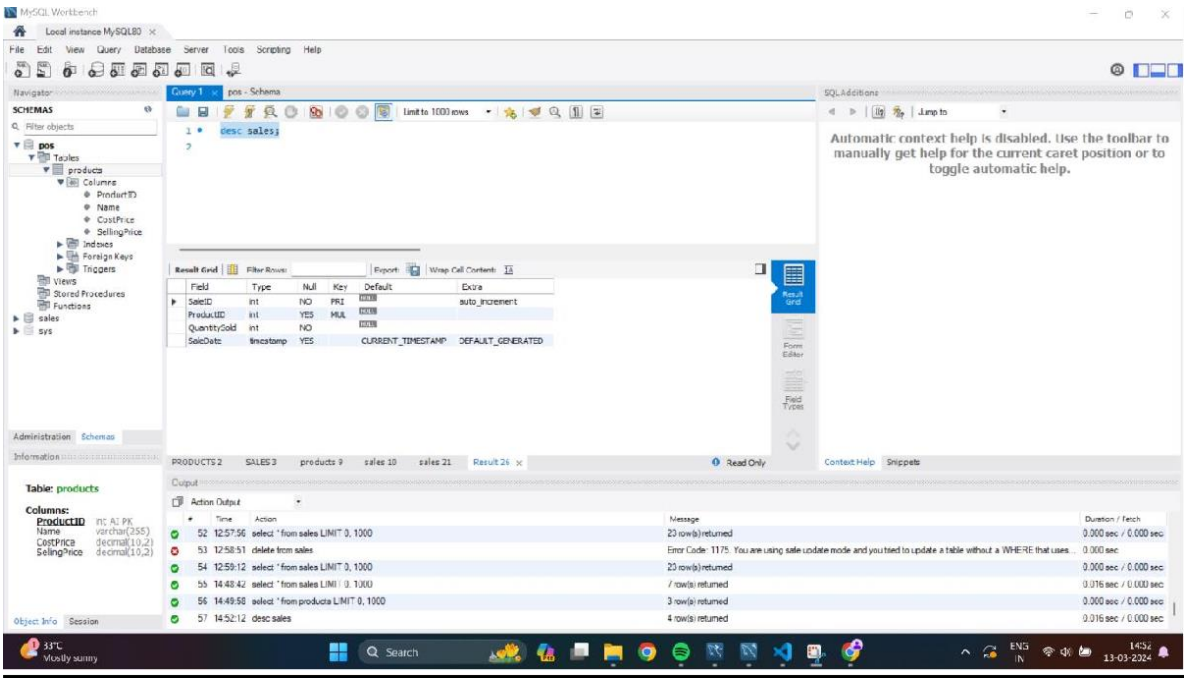
Testing and Debugging

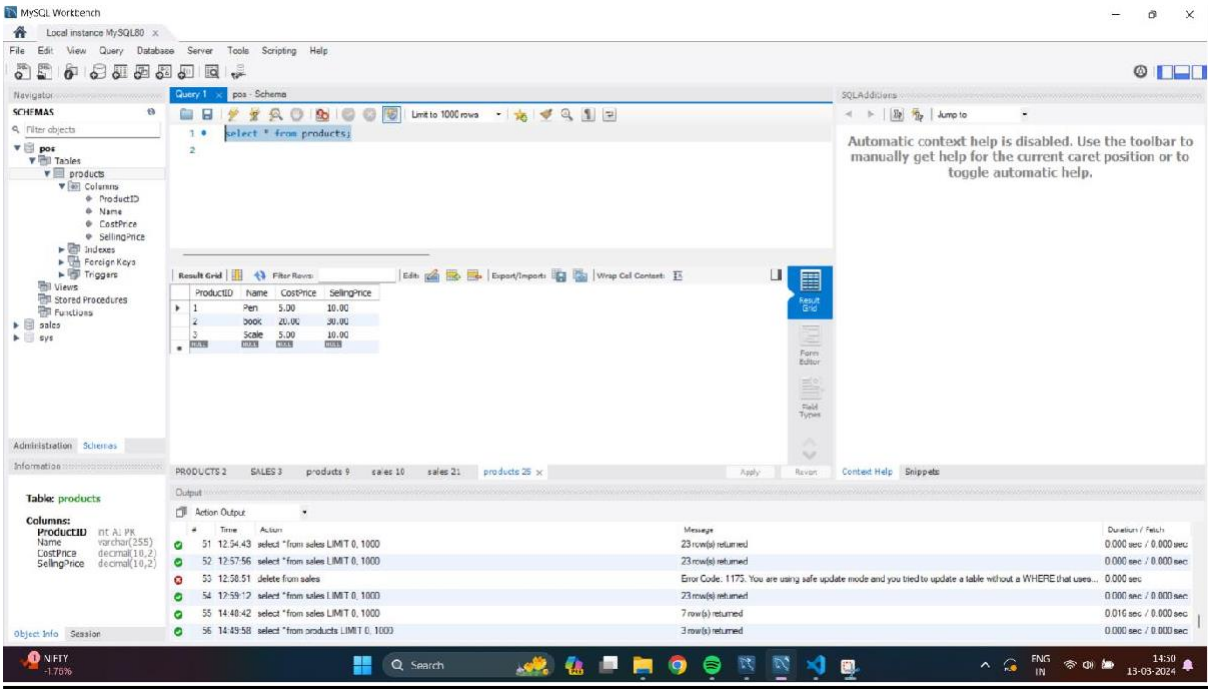
The application underwent rigorous debugging, with special attention to the data binding and command handling within the MVVM framework. Edge cases, especially within the profit calculation and database interaction routines, were methodically tested.

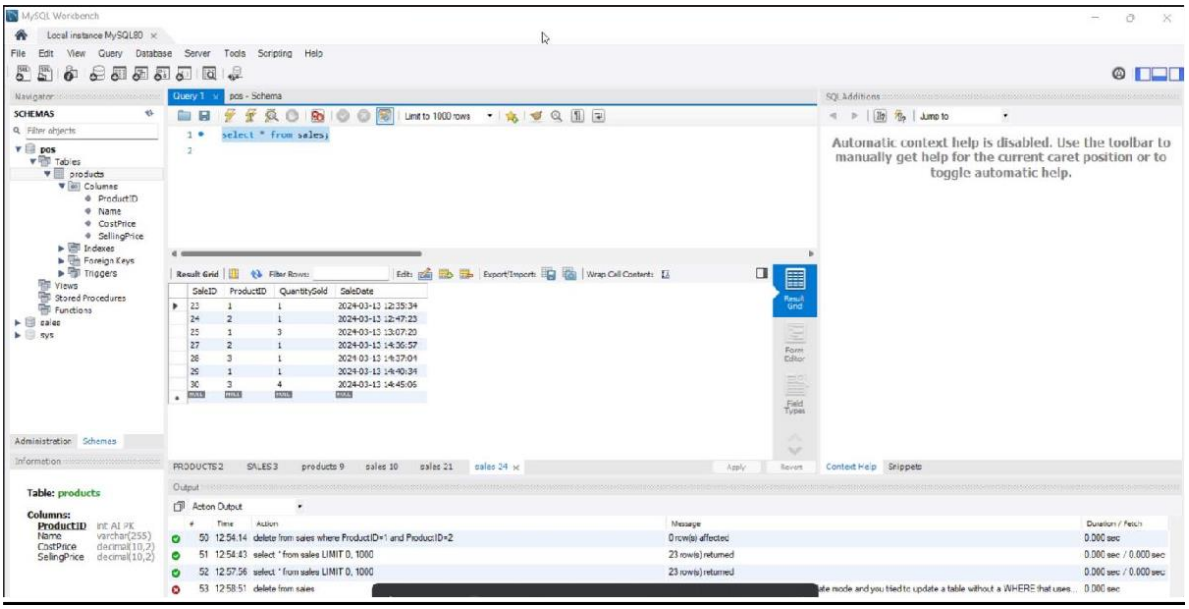
RESULTS(SNAPSHOTS)

Database:

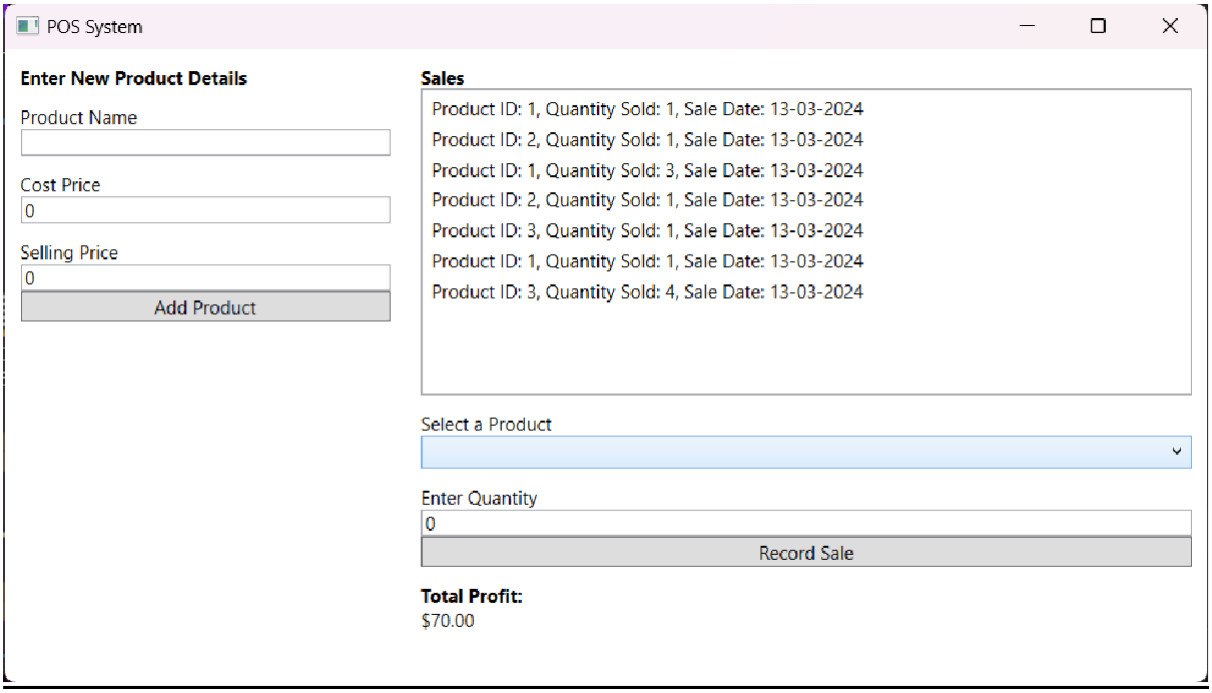


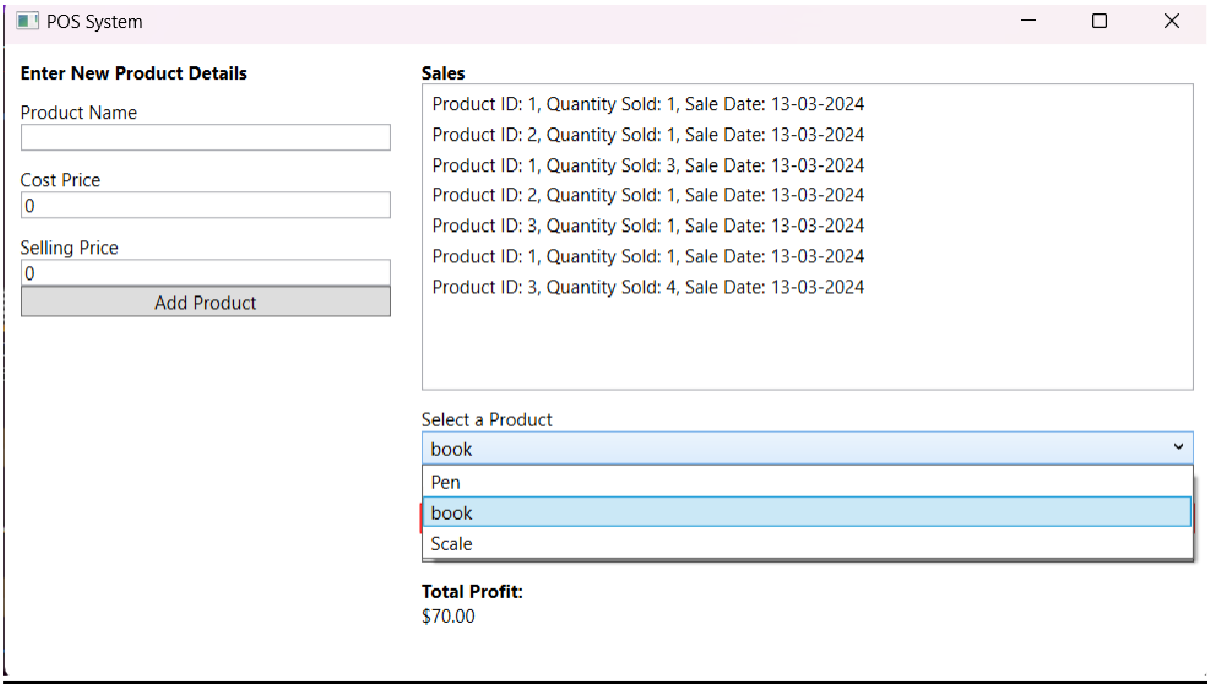



MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
pos
Tables
products
Columns
ProductID
Name
CostPrice
SellingPrice
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sales
sys
Query 1
pos - Schema
select * from products;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
ProductID Name CostPrice SellingPrice
1 Pen 5.00 10.00
2 book 20.00 30.00
3 Scale 5.00 10.00
Table: products
Columns:
ProductID int AI PK
Name varchar(255)
CostPrice decimal(10,2)
SellingPrice decimal(10,2)
Output
Action Output
51 12:54:43 select * from sales LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec
52 12:57:56 select * from sales LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec
53 12:58:51 delete from sales Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses... 0.000 sec
54 12:59:12 select * from sales LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec
55 14:48:42 select * from sales LIMIT 0, 1000 7 row(s) returned 0.016 sec / 0.000 sec
56 14:49:58 select * from products LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
NIFTY -1.76%
Search
ENG IN
14:50 13-03-2024


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
pos
Tables
products
Columns
ProductID
Name
CostPrice
SellingPrice
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sales
sys
Query 1
pos - Schema
select * from sales;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
SaleID ProductID QuantitySold SaleDate
23 1 1 2024-03-13 12:25:34
24 2 1 2024-03-13 12:47:23
25 1 3 2024-03-13 13:07:20
27 2 1 2024-03-13 14:06:57
28 3 1 2024-03-13 14:27:01
29 1 1 2024-03-13 14:40:34
30 3 4 2024-03-13 14:45:06
Table: products
Columns:
ProductID int AI PK
Name varchar(255)
CostPrice decimal(10,2)
SellingPrice decimal(10,2)
Output
Action Output
50 12:54:14 delete from sales where ProductID=1 and ProductID=2 0 row(s) affected 0.000 sec
51 12:54:43 select * from sales LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec
52 12:57:56 select * from sales LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec
53 12:58:51 delete from sales

POS Application:





Bitbucket link: https://bitbucket.org/varshitha-s/posapp/src/master/