 Faculty of Computing

**Documentation For Application Development**

**London Metropolitan University, Faculty of Computing**

**CS6004ES Application Development**

**Module: CS6004ES Setter: P. K. Vimarshana**

# **Title: SEP 2024 - AD - CW 01**

## Student ID: 00238397

**Introduction**

The BookHaven Management System is a desktop application designed to help bookstore administrators and sales clerks manage various aspects of bookstore operations, including user management, customer management, supplier management, book inventory, sales, orders, and reporting.

**Key Features and Scope**

**Overview**

The **BookHaven Management System** is a **Windows Forms-based desktop application** developed in **C#** to provide a **comprehensive, efficient, and scalable** solution for bookstore operations. Designed for **bookstore administrators and sales clerks**, the system integrates essential functionalities, including **user management, customer management, supplier management, book inventory control, sales tracking, order processing, and reporting**. By streamlining these processes, BookHaven reduces manual workload, improves operational efficiency, and ensures secure, well-organized bookstore management. The system enables seamless inventory tracking, real-time stock updates, and automated sales processing, allowing businesses to manage books, customers, and suppliers effectively. Additionally, robust **reporting and analytics** capabilities provide valuable insights into sales trends, customer preferences, and inventory status, supporting informed decision-making. With a **user-friendly interface**, **role-based access control**, and **scalability**, BookHaven is designed for long-term usability, ensuring adaptability to evolving business needs while maintaining security and efficiency.

**Objectives**

* **Provide a user-friendly bookstore management system.**
* **Improve inventory tracking and order processing.**
* **Automate sales transactions and customer management.**
* **Generate detailed reports for business insights.**
* **Ensure security with role-based access and data protection.**

**Functional Scope**

The system includes the following core features:

* Authentication and Authorization: – Secure authentication with role-based access (Admin, Sales Clerk).
  + Users must log in to access the system.
  + Role-based access control ensures that only authorized users can perform specific actions.
* Book Inventory Management: – Add, update, delete, and search books with details (title, author, genre, ISBN, price, stock).
  + Admins can manage the bookstore's inventory (add, update, delete books).
  + Books are categorized by genre, author, and ISBN.
* Customer Management: – Manage customer profiles, contact details, and purchase history.
  + Admins and sales clerks can manage customer information (add, update, delete).
  + Customers are associated with sales transactions.
* Sales Management: – Point-of-sale (POS) module to process purchases, apply discounts, and generate receipts.
  + Admins and sales clerks can manage sales transactions.
  + Sales include details like customer, total amount, discount, and sale date.
* Order Management: – Track orders.
  + Admins can manage book orders from suppliers.
  + Orders include details like order date, total amount, and status (e.g., Pending, Completed).
* Supplier Management: – Maintain supplier details.
  + Admins can manage supplier details (add, update, delete).
  + Suppliers are linked to book orders.
* Dashboard: – View overall bookstore performance metrics.
  + Provides an overview of key metrics (e.g., total sales, orders, and inventory).
  + Includes charts for visualizing data (e.g., monthly sales, revenue by genre).
* Reporting & Analytics: – Generate reports on sales and orders.
  + Generate reports for sales and orders.
  + Reports include details like monthly sales, top-selling books, and revenue by genre.
* Security & Data Protection: – Role-based access, data encryption, and validation.

**Out of Scope**

* Online Integration:
  + The system does not include e-commerce functionality or integration with online platforms.
* Advanced Analytics:
  + The system provides basic reporting but does not include advanced data analytics or machine learning capabilities.
* Multi-User Concurrency:
  + The system is designed for single-user or small-team use and does not handle concurrent access by multiple users.
* Mobile Support:
  + The system is a desktop application and does not support mobile devices.

**Target Users**

* Admin:
  + Full access to all features (user management, customer management, supplier management, book inventory, orders, sales, and reporting).
* Sales Clerk:
  + Limited access to customer management, sales management, and basic reporting.

**Technology Stack**

* Programming Language: C#
* Framework: .NET Windows Forms
* Database: (SQL Server)
* Charts: LiveCharts for data visualization
* Logging: Basic logging for error tracking
* Development Tools: Visual Studio 2022
* Architecture: Multi-layered (Data Layer, Business Logic, Presentation Layer)

**Deliverables**

* Functional Desktop Application:
  + A fully functional application with a user-friendly interface for managing bookstore operations.
* Documentation:
  + UML diagrams (class diagrams, use case diagrams).
  + User manuals for admins and sales clerks.
* Source Code:
  + Well-structured, modular, and documented code adhering to best practices.
  + Repository Link: https://github.com/kasunvimarshana/BookHaven

**Installation Guide  
  
System Requirements**

Before installing BookHaven, ensure your system meets the following requirements:

* Operating System: Windows 10 or later (64-bit recommended)
* Processor: Intel Core i3 or higher
* RAM: 4GB minimum (8GB recommended)
* Disk Space: At least 500MB of free space
* Framework: .NET 7.0 or later
* Database: SQL Server (or compatible database)
* Screen Resolution: 1024x768 or higher

**Installation Steps**

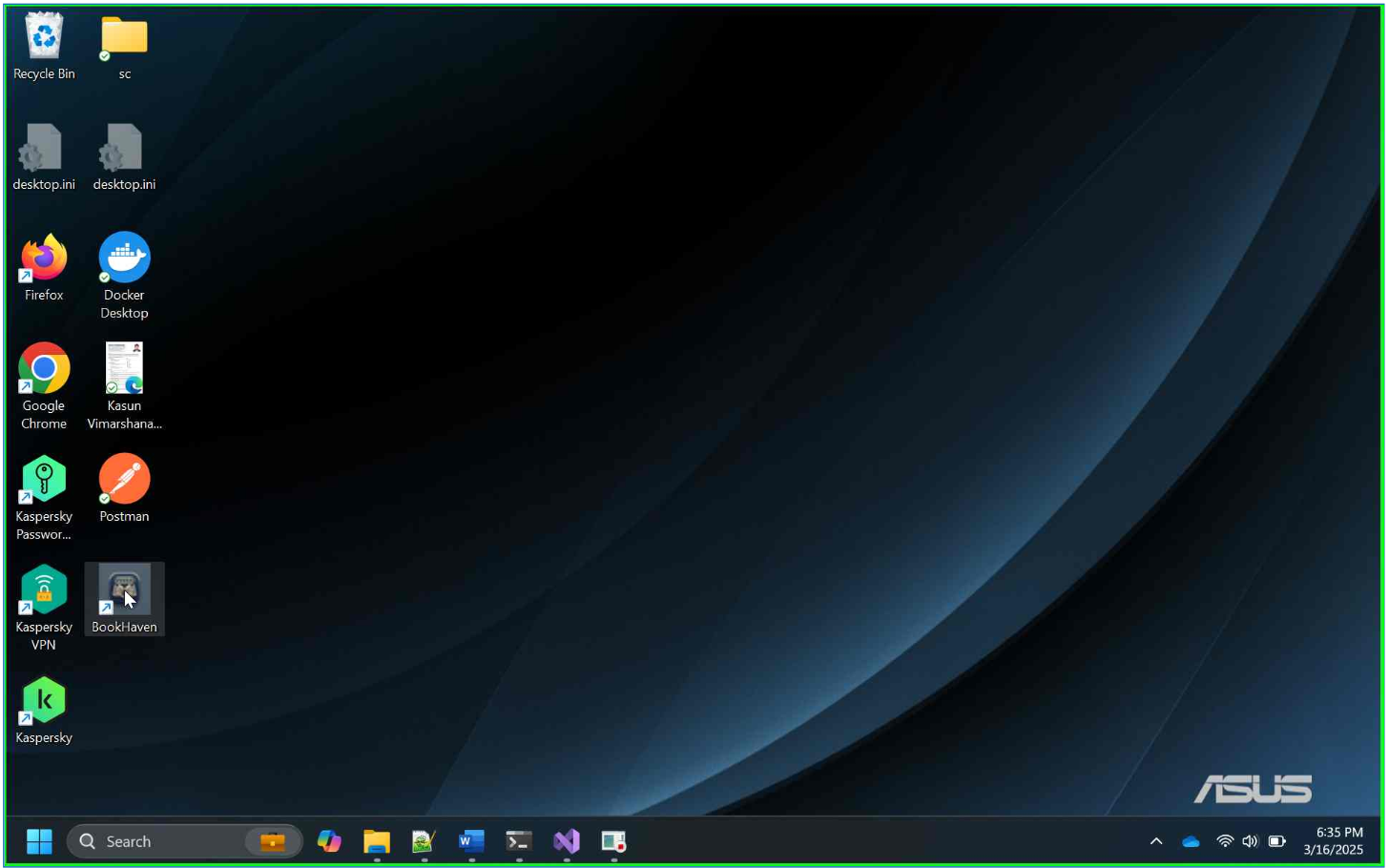
Follow these steps to install BookHaven on your computer:

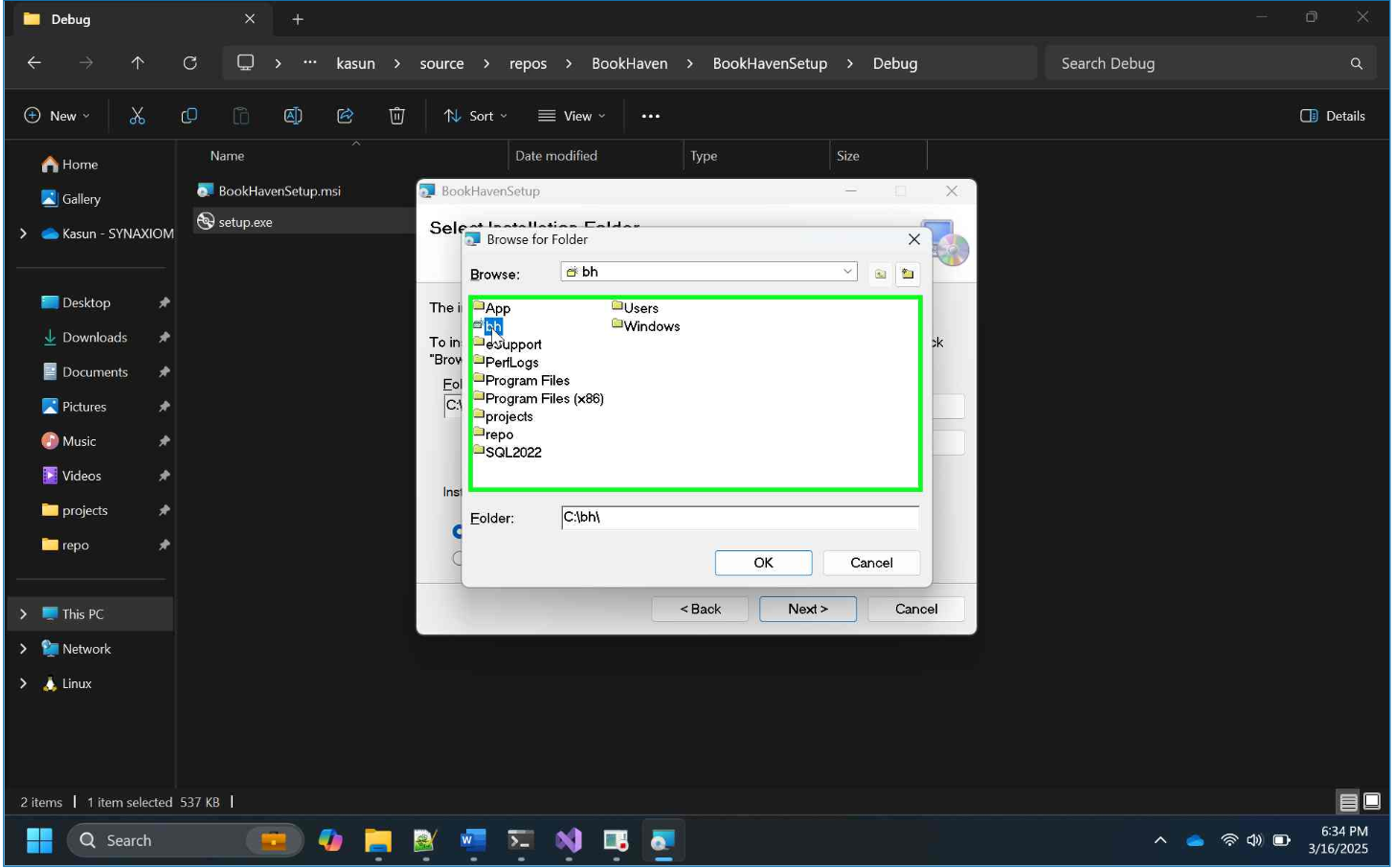
* Locate the Setup File
  + Download the BookHavenSetup.exe file from the official source.
  + Navigate to the folder where the setup file is saved.
* Start the Installation
  + Double-click on BookHavenSetup.exe to launch the installer.
  + The installer welcome screen will appear.
  + Click on Next to proceed.
* Accept the License Agreement
  + Read the end-user license agreement carefully.
  + If you agree to the terms, select I Agree and click Next.
  + If you do not agree, click Cancel to exit the installation.
* Select Installation Scope And Installation Location
  + Choose one of the following options:
    - Everyone (Install for all users on the system)
    - Just Me (Install only for the current user)
  + By default, BookHaven will install in C:\Program Files\BookHaven.
  + To change the installation location, click Browse and select your preferred folder.
  + Click Next to continue.
* Begin Installation
  + Review your selected options.
  + Click Next to start the installation process.
* Complete Installation
  + Once the installation is complete, a confirmation message will appear.
  + Click Close to close the installer.
  + If prompted, restart your computer to apply changes.

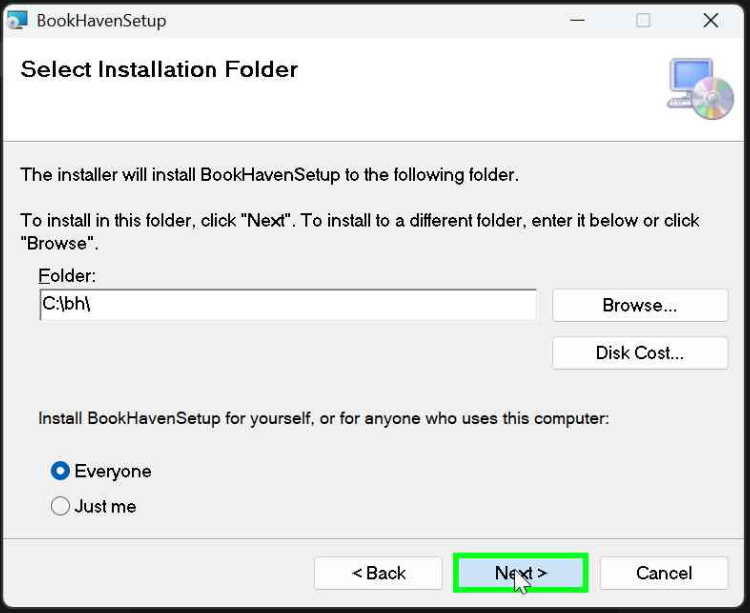
**Post Installation Steps**

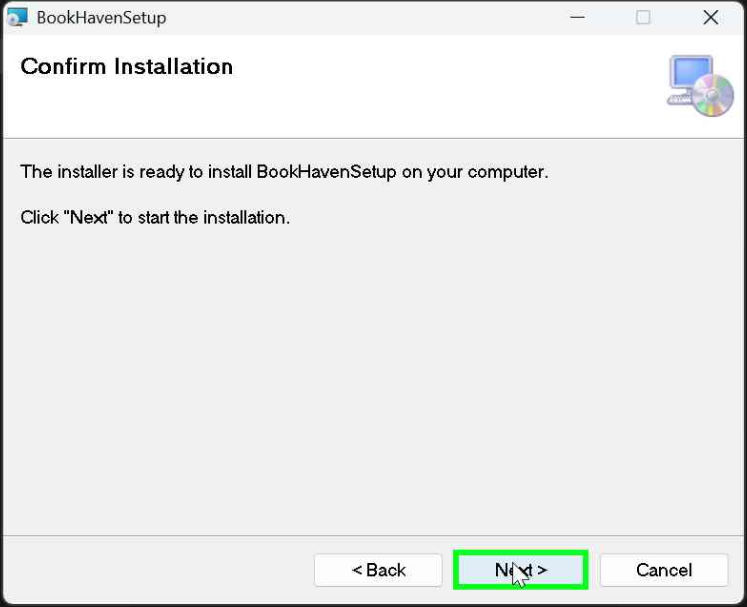
* Launch BookHaven from the Start Menu or Desktop shortcut.
* Log in using the default user credentials:
  + Username: admin
  + Password: admin

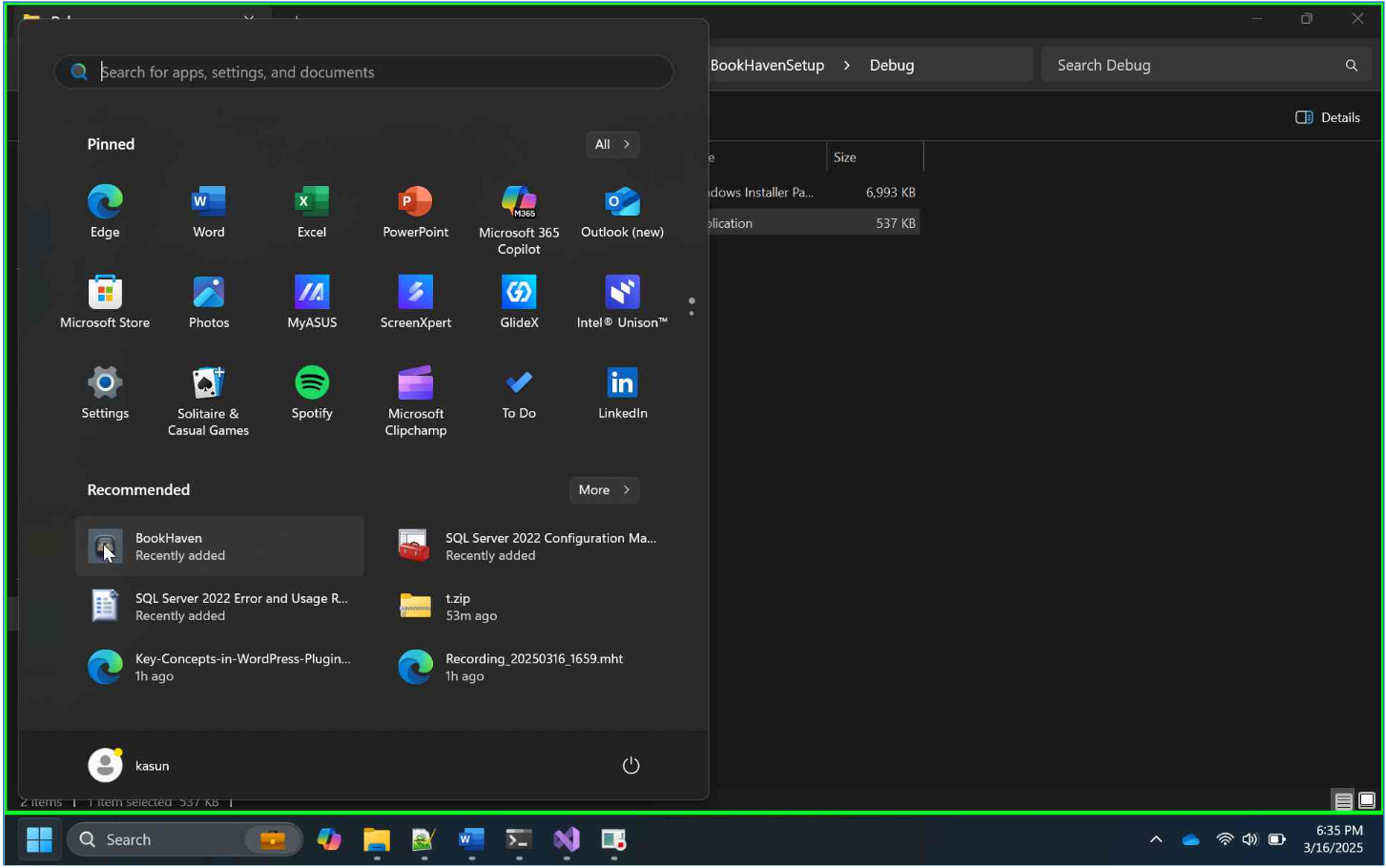
For troubleshooting or further assistance, refer to the User Manual or contact support.











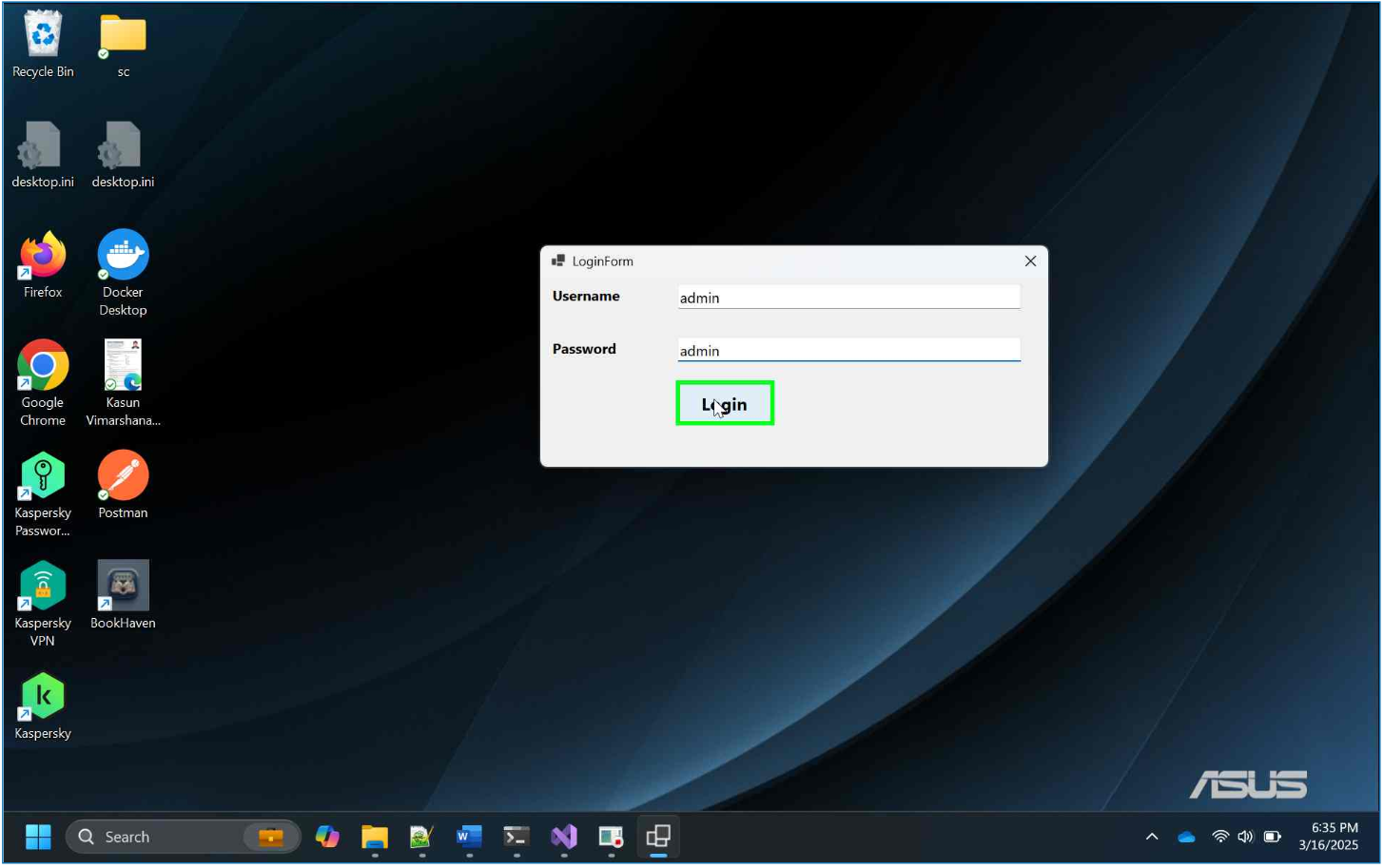
**User Manual for BookHaven Management System**

This user manual provides step-by-step instructions on how to use the system effectively.

**Getting Started**

1. **Login**

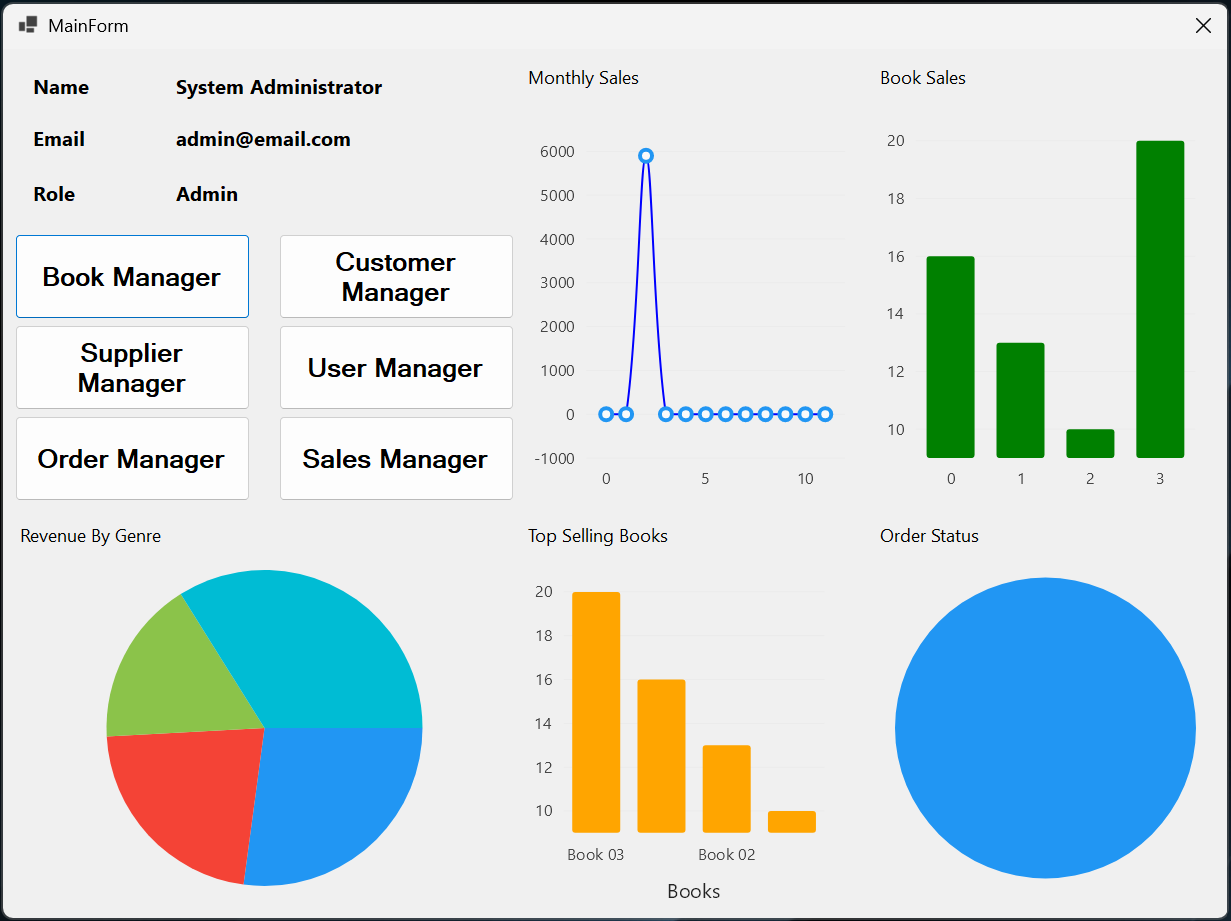
* Launch the application.
* Enter your Username and Password.
  + Default Username: admin
  + Default Password: admin
* Click Login.
  + Admins have access to all features.
  + Sales Clerks have limited access (customer management, sales, and basic reporting).

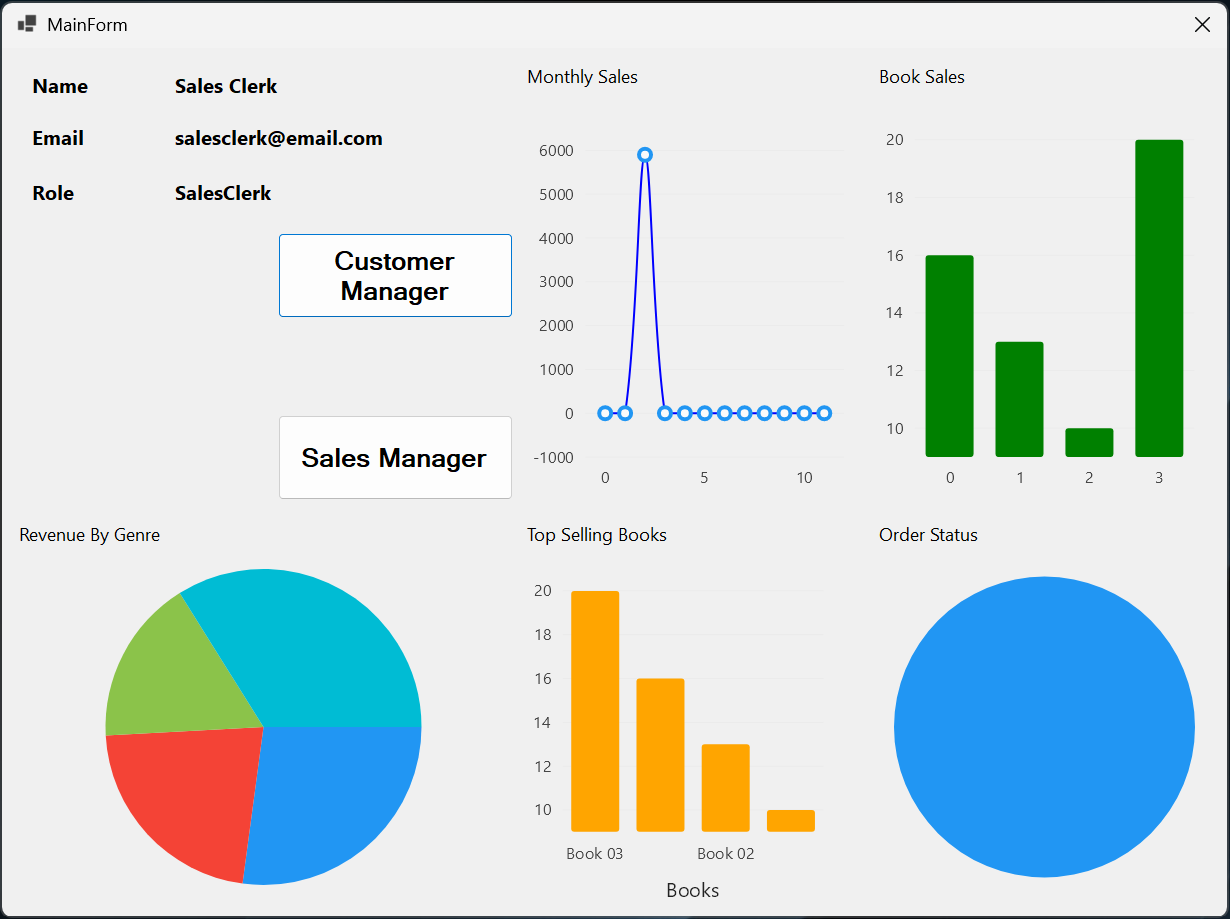


**Main Features**

1. **Dashboard**

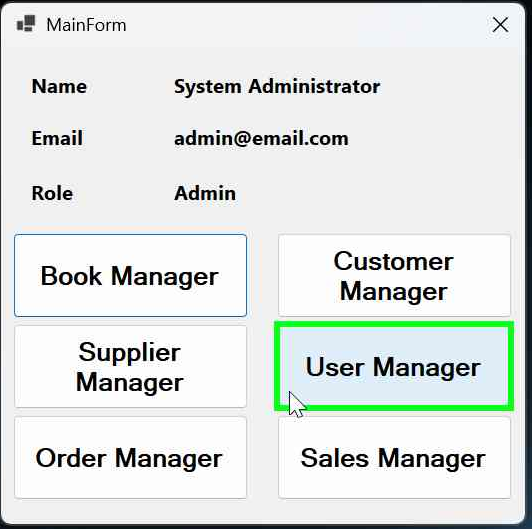
* Purpose: Provides an overview of key metrics and visualizations.
* Features:
  + View total sales, orders, and inventory.
  + Visualize data using charts (e.g., monthly sales, revenue by genre).

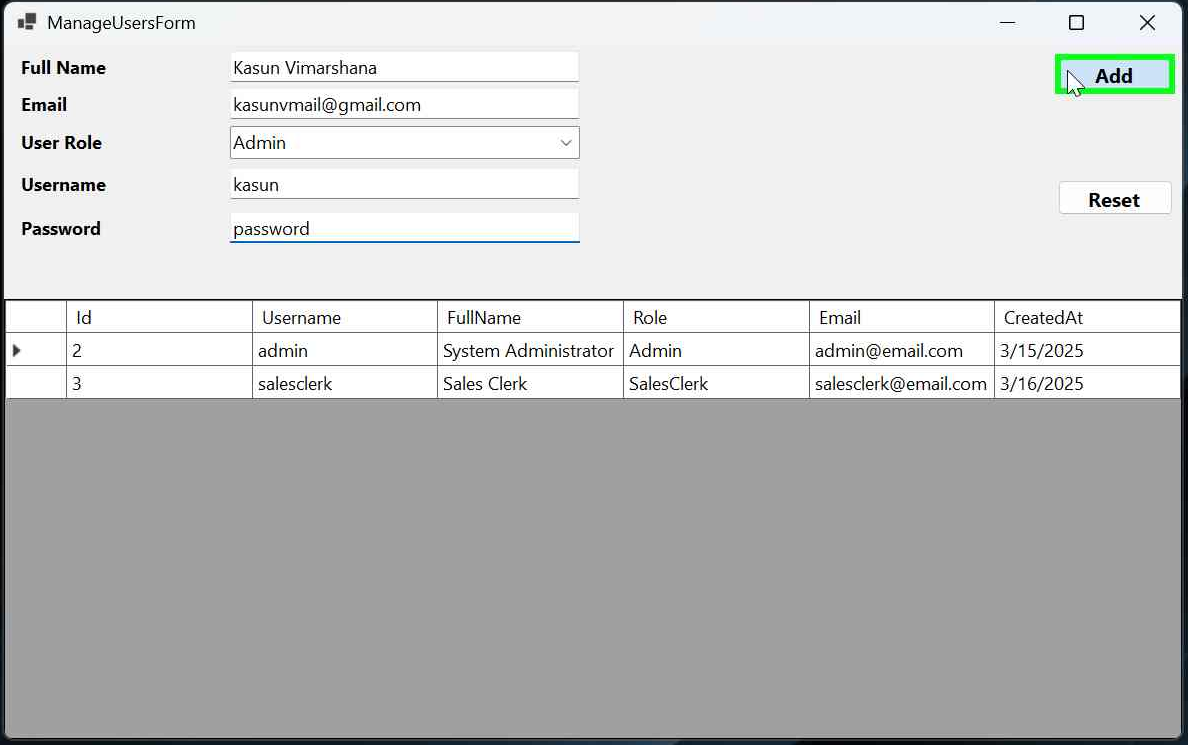


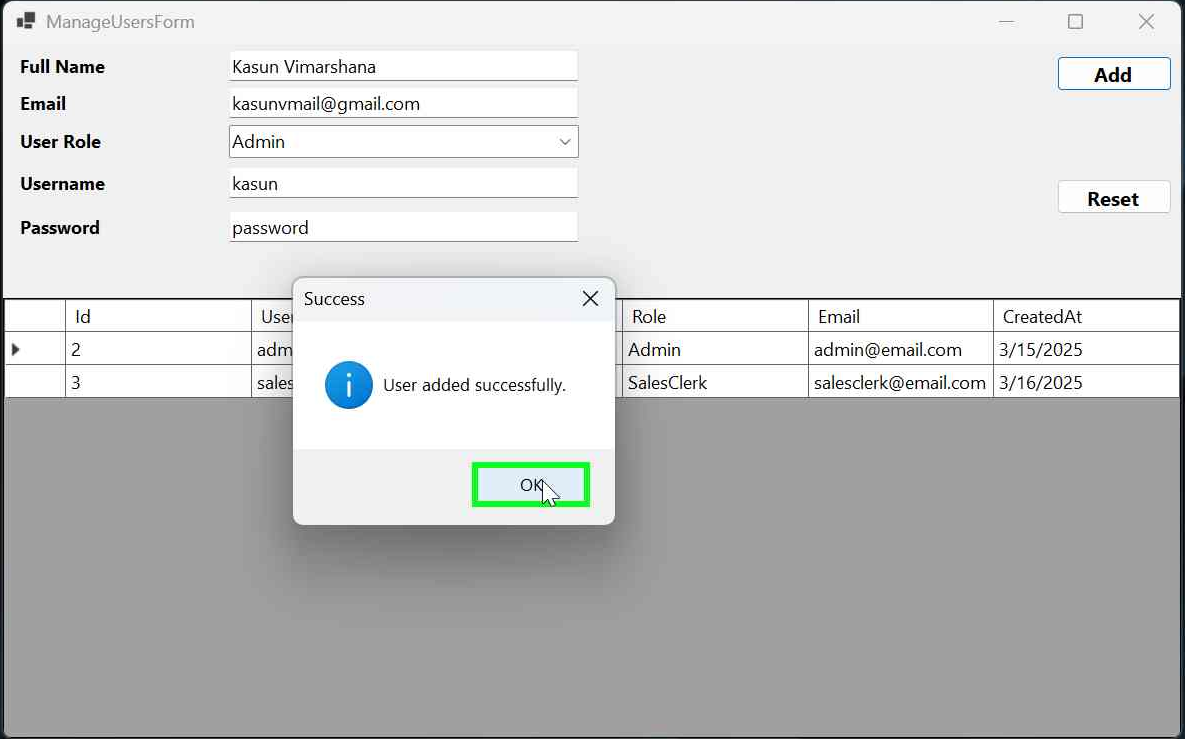


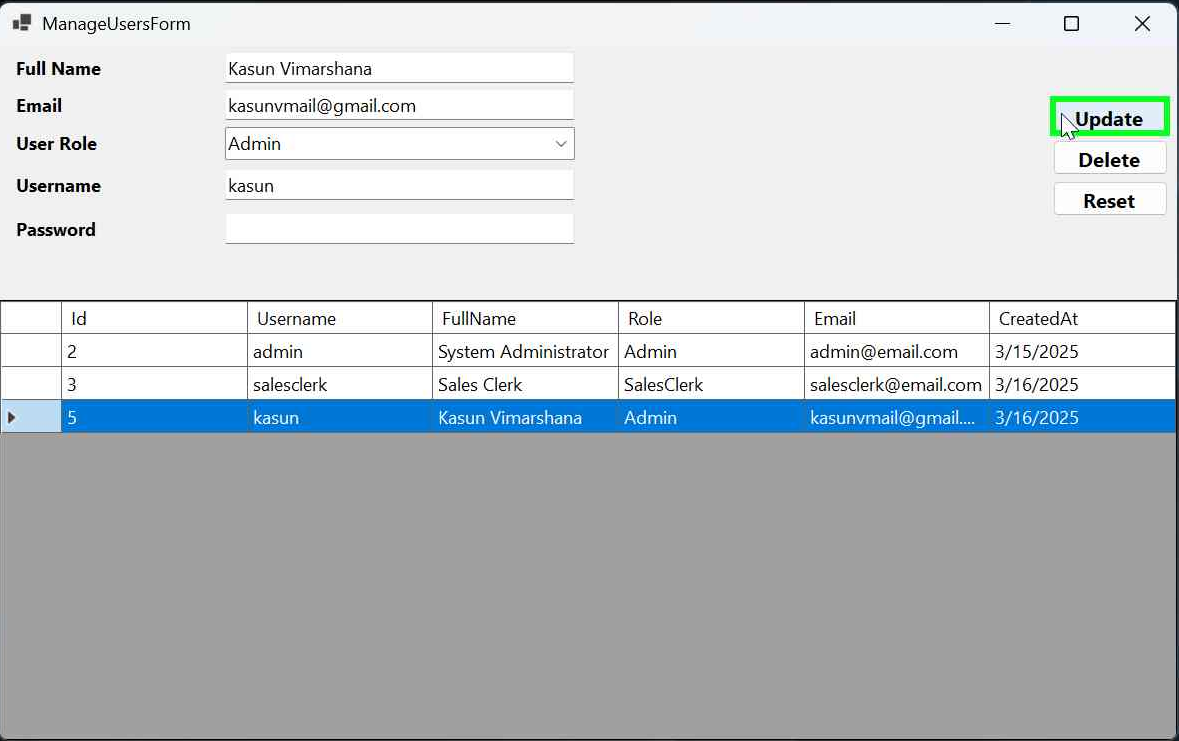
1. **User Management**

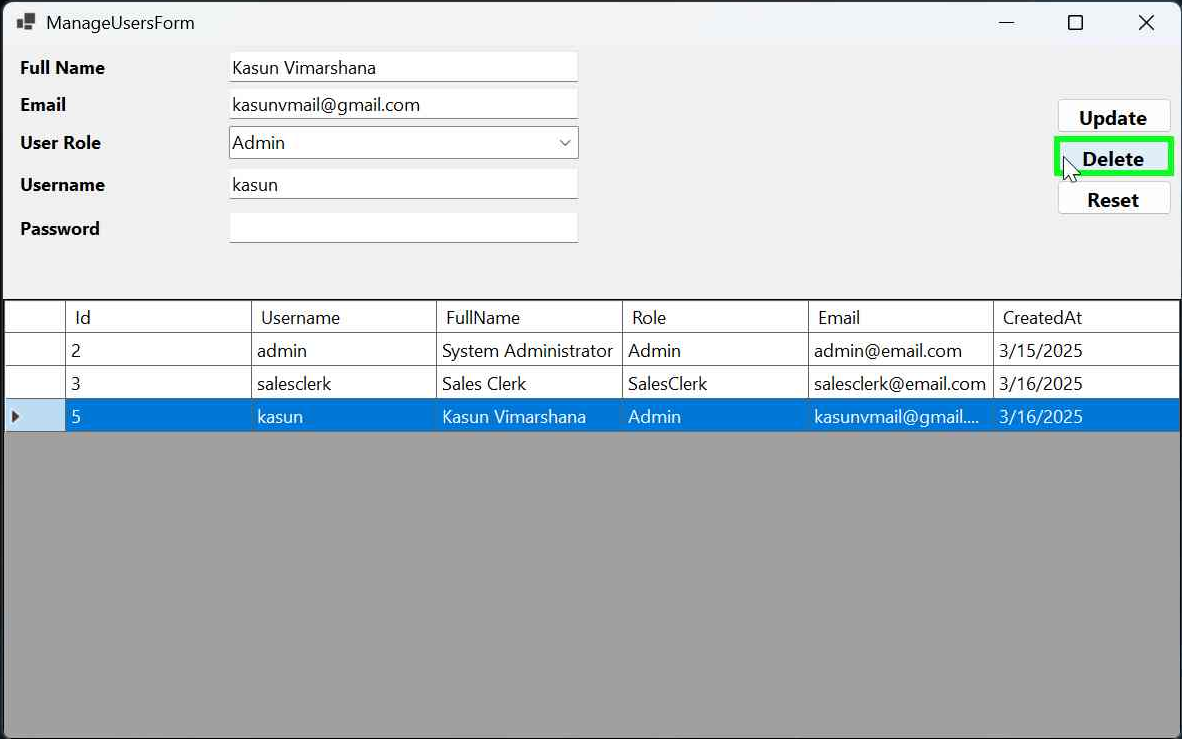
* Access: Admin only.
* Purpose: Manage user accounts (create, update, delete).
* Steps:
  + Navigate to User Management from the main menu.
  + To add a user:
    - Fill in the details (Username, Password, Full Name, Email, Role).
    - Click Save.
  + To update a user:
    - Select a user from the list
    - Modify the details.
    - Click Update.
  + To delete a user:
    - Select a user from the list.
    - Click Delete.
    - Confirm the deletion.

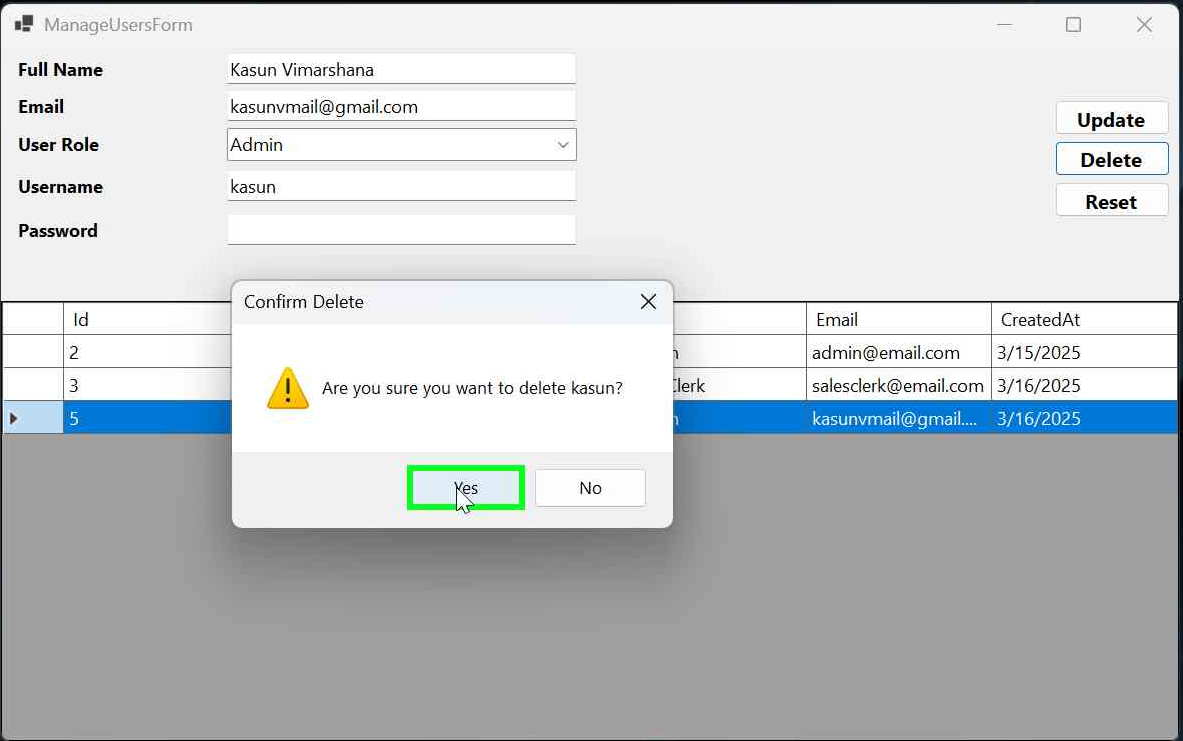


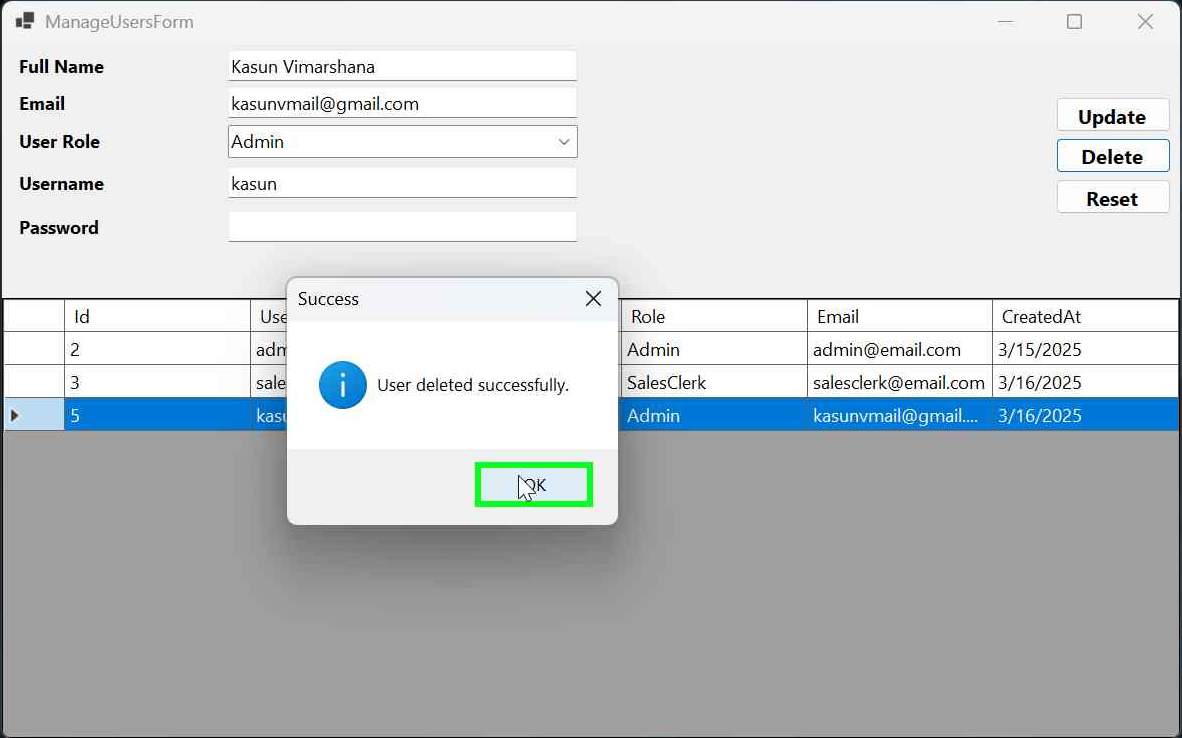






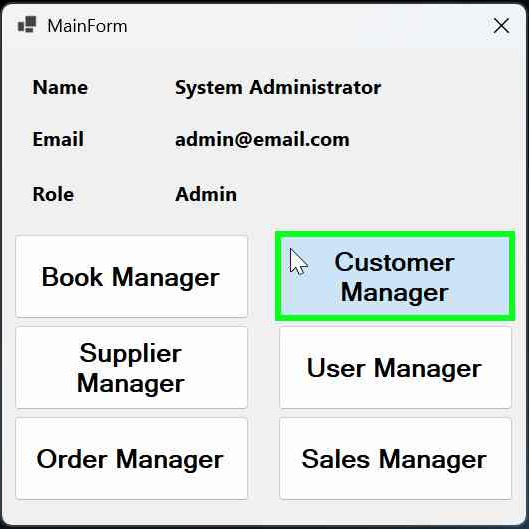


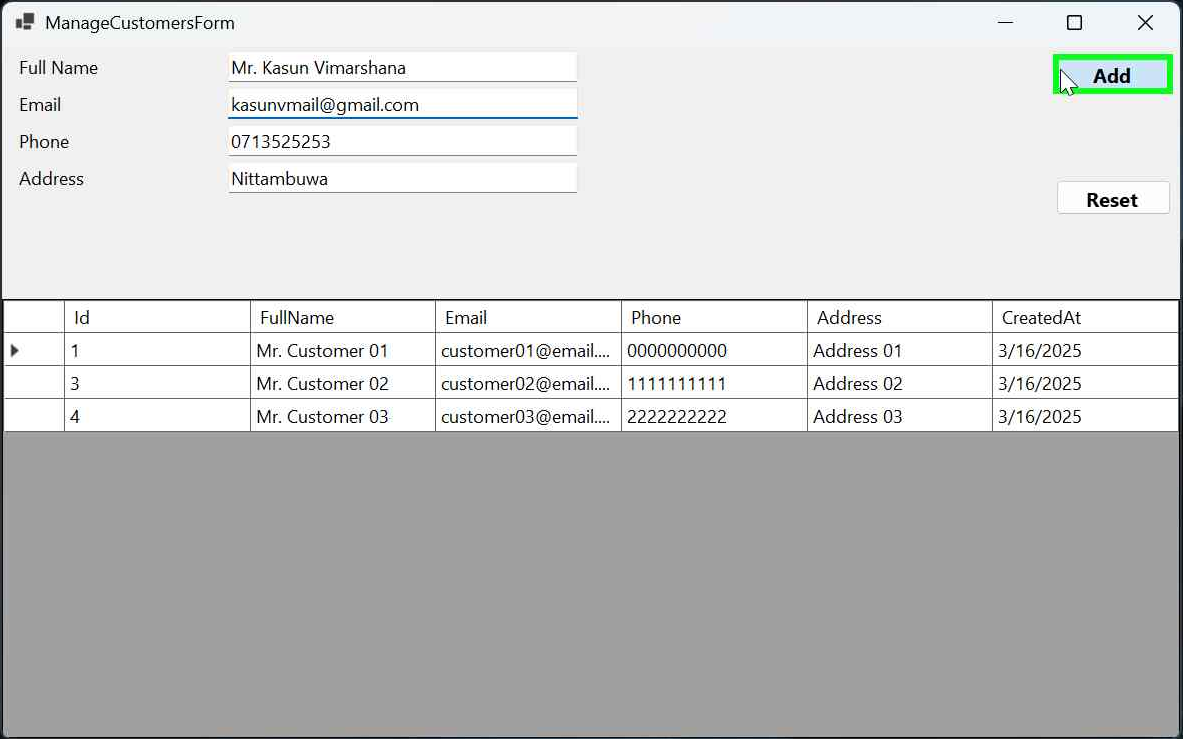


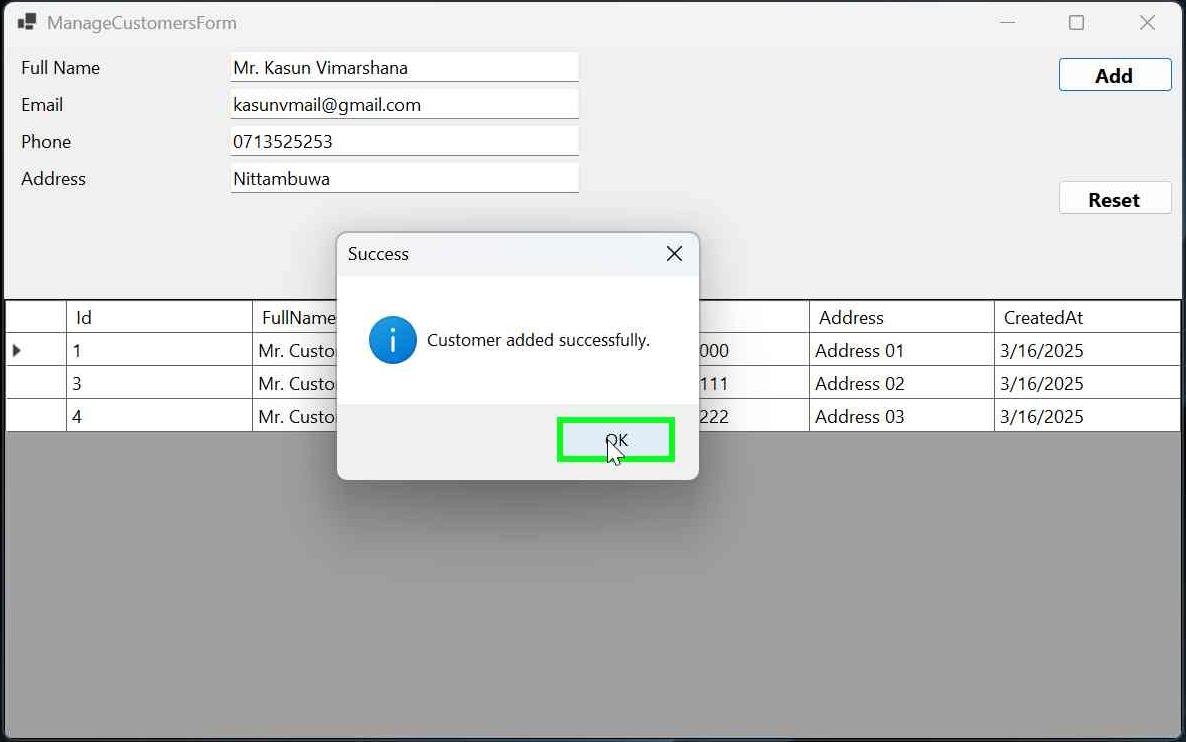


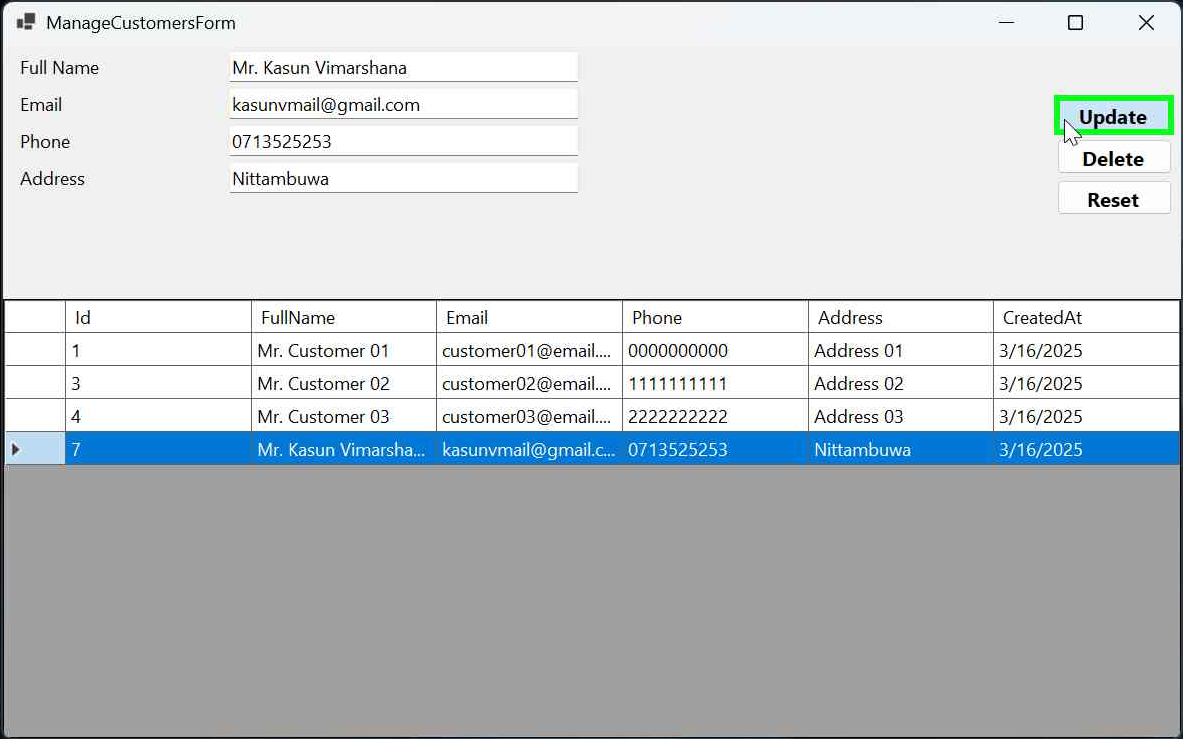
1. **Customer Management**

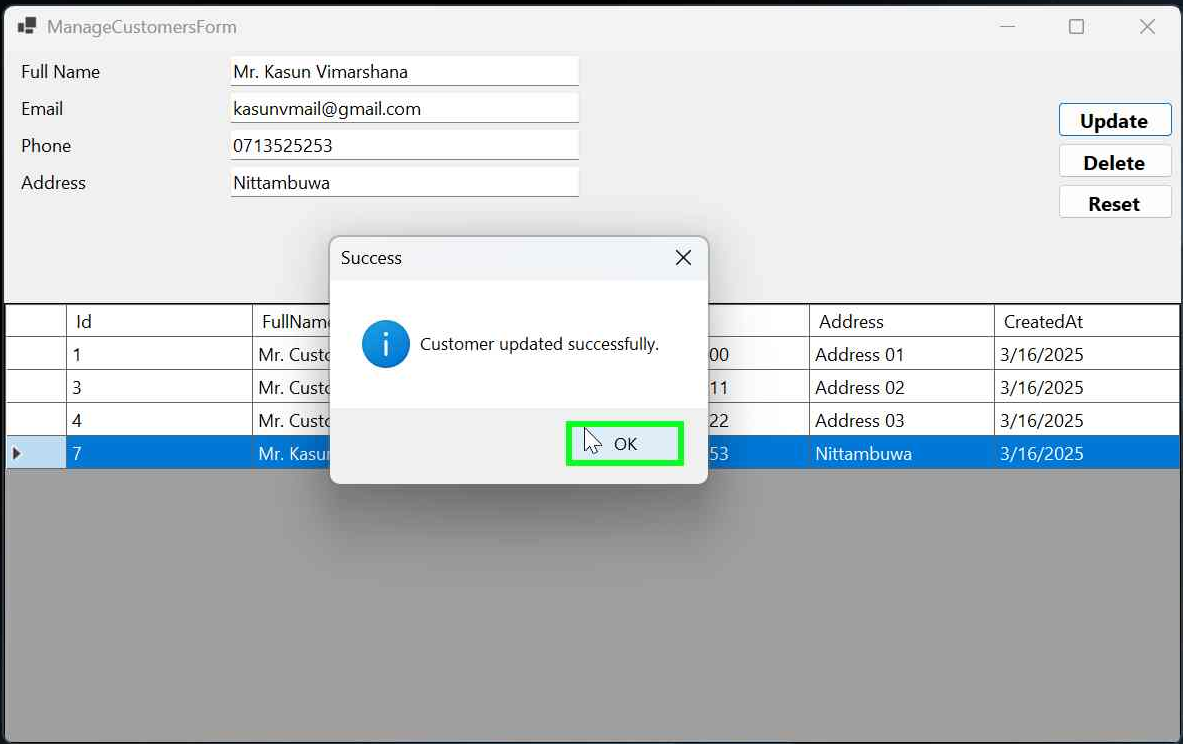
* Access: Admin and Sales Clerk.
* Purpose: Manage customer information.
* Steps:
  + Navigate to Customer Management from the main menu.
  + To add a customer:
    - Fill in the details (Full Name, Email, Phone, Address).
    - Click Save.
  + To update a customer:
    - Select a customer from the list.
    - Modify the details.
    - Click Update.
  + To delete a customer:
    - Select a customer from the list.
    - Click Delete.
    - Confirm the deletion.

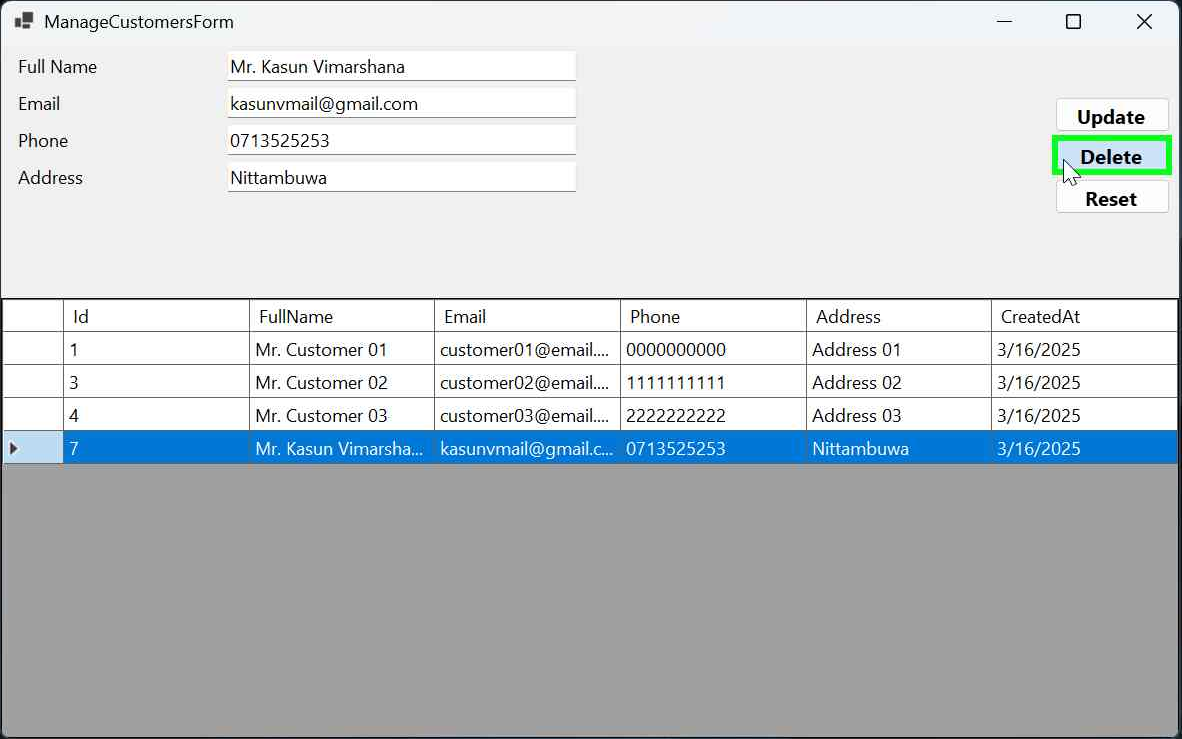


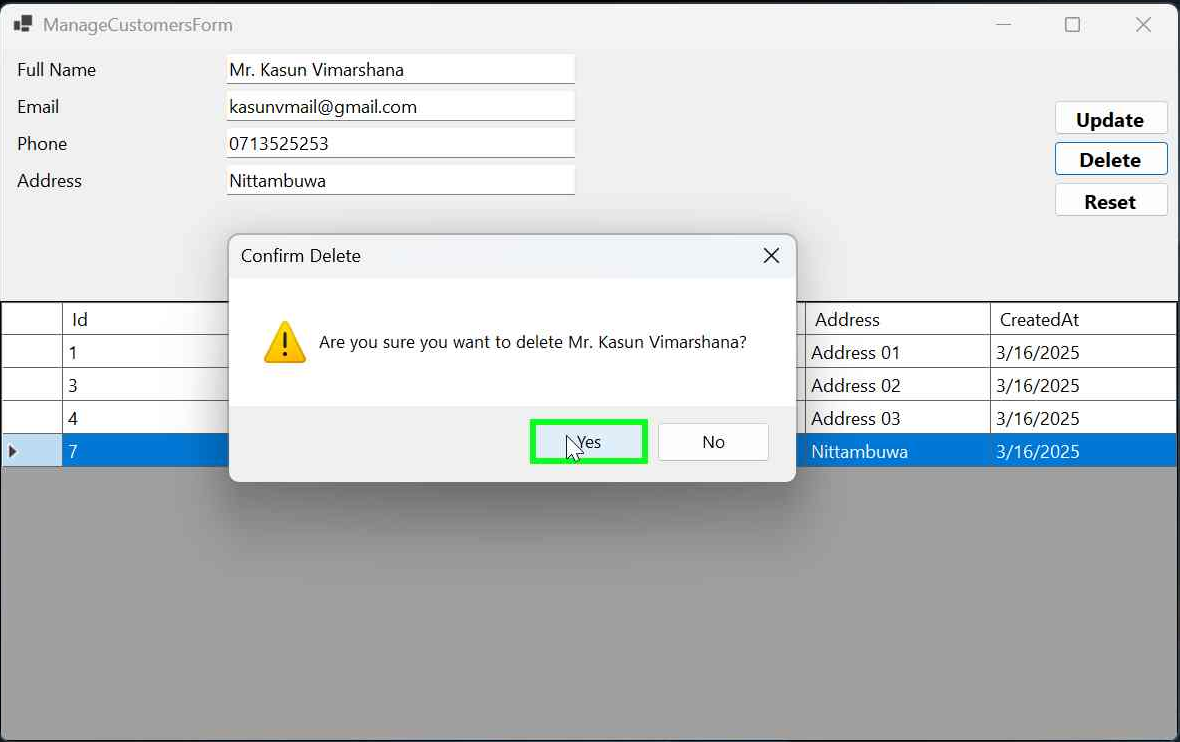


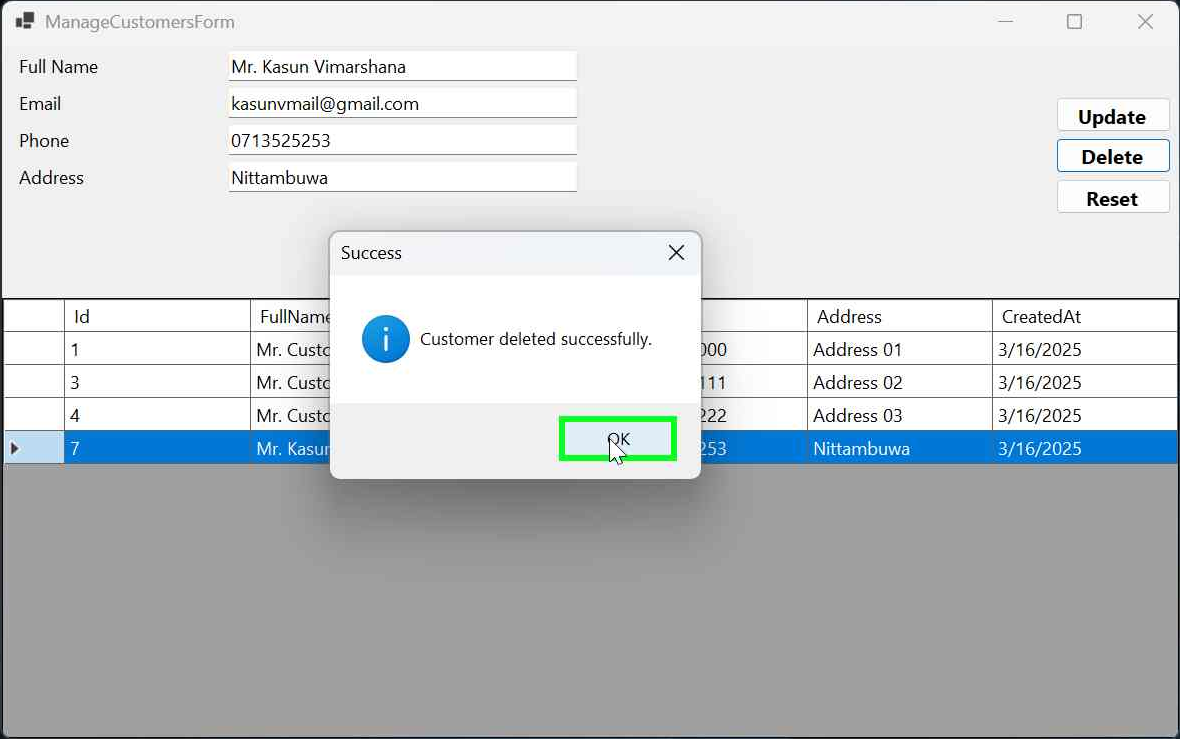












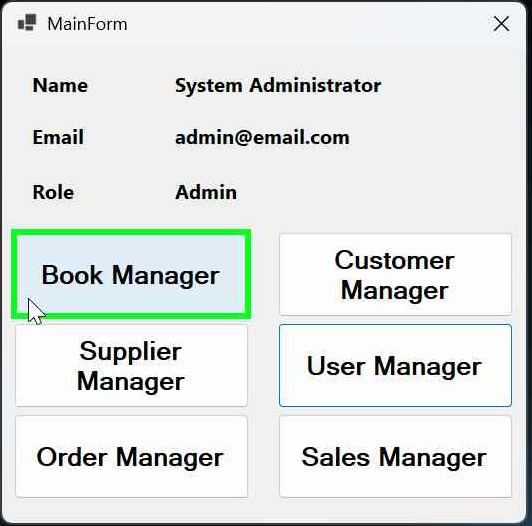
1. **Supplier Management**

* Access: Admin only.
* Purpose: Manage supplier details.
* Steps:
  + Navigate to Supplier Management from the main menu.
  + To add a supplier:
    - Fill in the details (Name, Contact Person, Phone, Email, Address).
    - Click Save.
  + To update a supplier:
    - Select a supplier from the list.
    - Modify the details.
    - Click Update.
  + To delete a supplier:
    - Select a supplier from the list.
    - Click Delete.
    - Confirm the deletion.



1. **Book Management**

* Access: Admin only.
* Purpose: Manage book inventory.
* Steps:
  + Navigate to Book Management from the main menu.
  + To add a book:
    - Fill in the details (Title, Author, Genre, ISBN, Price, Stock Quantity).
    - Click Save.
  + To update a book:
    - Select a book from the list.
    - Modify the details.
    - Click Update.
  + To delete a book:
    - Select a book from the list.
    - Click Delete.
    - Confirm the deletion.



1. **Order Management**

* Access: Admin only.
* Purpose: Manage book orders from suppliers.
* Steps:
  + Navigate to Order Management from the main menu.
  + To add an order:
    - Select a supplier and fill in the details (Order Date, Total Amount, Status).
    - Click Save.
  + To update an order:
    - Select an order from the list.
    - Modify the details.
    - Click Update.
  + To delete an order:
    - Select an order from the list.
    - Click Delete.
    - Confirm the deletion.



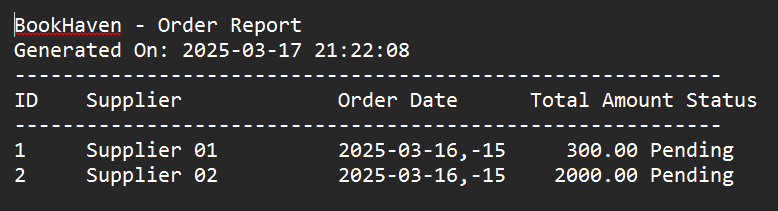
1. **Sales Management**

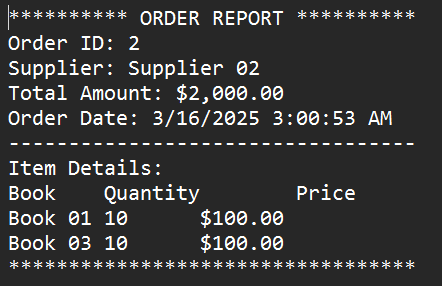
* Access: Admin and Sales Clerk.
* Purpose: Manage sales transactions.
* Steps:
  + Navigate to Sales Management from the main menu.
  + To add a sale:
    - Select a customer and fill in the details (Sale Date, Total Amount, Discount).
    - Click Save.
  + To update a sale:
    - Select a sale from the list.
    - Modify the details.
    - Click Update.
  + To delete a sale:
    - Select a sale from the list.
    - Click Delete.
    - Confirm the deletion.

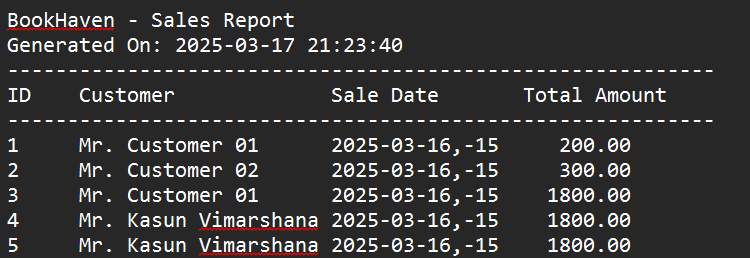


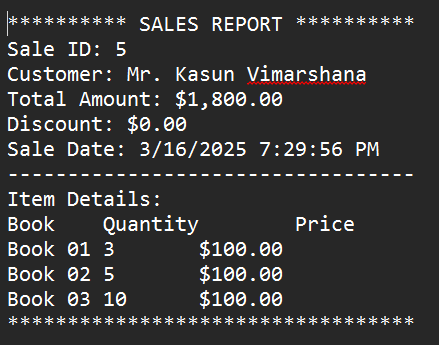
1. **Reporting**

* Access: Admin and Sales Clerk.
* Purpose: Generate reports for sales and orders.
* Steps:
  + Navigate to Reports from the main menu.
  + To generate a sales report:
    - Click Generate Sales Report.
    - The report will be saved as a text file and opened automatically.
  + To generate an order report:
    - Click Generate Order Report.
    - The report will be saved as a text file and opened automatically.









**Architecture Of System**

**BookHaven** application follows a well-structured **multi-layered architecture**, adhering to the **MVC (Model-View-Controller) pattern**. Below is a breakdown of each layer, its responsibilities, and how they interact with each other.

:.

| .gitignore

| BookHaven.sln

| README.md

|

+---BookHaven

| | App.config

| | BookHaven.csproj

| | BookHaven.csproj.user

| | db.mdf

| | db\_log.ldf

| | Program.cs

| |

| +---BLL

| | AuthenticationService.cs

| | BookService.cs

| | CustomerService.cs

| | OrderDetailService.cs

| | OrderService.cs

| | SalesDetailService.cs

| | SalesService.cs

| | SupplierService.cs

| | UserService.cs

| |

| +---DAL

| | BookRepository.cs

| | CustomerRepository.cs

| | DatabaseHelper.cs

| | IDatabaseHelper.cs

| | OrderDetailRepository.cs

| | OrderRepository.cs

| | SalesDetailRepository.cs

| | SalesRepository.cs

| | SupplierRepository.cs

| | UserRepository.cs

| |

| +---Enums

| | Month.cs

| | OrderStatus.cs

| | UserRole.cs

| |

| +---Helpers

| | LoggedInUserSession.cs

| |

| +---Models

| | Book.cs

| | Customer.cs

| | Order.cs

| | OrderDetail.cs

| | Sale.cs

| | SalesDetail.cs

| | Supplier.cs

| | User.cs

| |

| +---Properties

| | \---PublishProfiles

| +---Reports

| | OrderDetailReportService.cs

| | OrderReportService.cs

| | SalesDetailReportService.cs

| | SalesReportService.cs

| |

| +---Seeders

| | UserSeeder.cs

| |

| +---UI

| | \---Forms

| | | LoginForm.cs

| | | LoginForm.Designer.cs

| | | LoginForm.resx

| | | MainForm.cs

| | | MainForm.Designer.cs

| | | MainForm.resx

| | |

| | +---Book

| | | ManageBooksForm.cs

| | | ManageBooksForm.Designer.cs

| | | ManageBooksForm.resx

| | |

| | +---Customer

| | | ManageCustomersForm.cs

| | | ManageCustomersForm.Designer.cs

| | | ManageCustomersForm.resx

| | |

| | +---Order

| | | ManageOrdersForm.cs

| | | ManageOrdersForm.Designer.cs

| | | ManageOrdersForm.resx

| | |

| | +---OrderDetail

| | | ManageOrderDetailsForm.cs

| | | ManageOrderDetailsForm.Designer.cs

| | | ManageOrderDetailsForm.resx

| | |

| | +---Sale

| | | ManageSalesForm.cs

| | | ManageSalesForm.Designer.cs

| | | ManageSalesForm.resx

| | |

| | +---SalesDetail

| | | ManageSalesDetailsForm.cs

| | | ManageSalesDetailsForm.Designer.cs

| | | ManageSalesDetailsForm.resx

| | |

| | +---Supplier

| | | ManageSuppliersForm.cs

| | | ManageSuppliersForm.Designer.cs

| | | ManageSuppliersForm.resx

| | |

| | \---User

| | ManageUsersForm.cs

| | ManageUsersForm.Designer.cs

| | ManageUsersForm.resx

| |

| \---Utilities

| Logger.cs

| PasswordHelper.cs

|

**Models Layer (Entities)**

* Directory: BookHaven/Models/
* Purpose: Defines the data structure used in the application, representing real-world entities like books, customers, orders, etc.
* Key Characteristics:
  + Classes in this layer define the properties of entities.
  + Used throughout the application in Business Logic (BLL) and Data Access Layer (DAL).

**namespace BookHaven.Models**

**{**

**class Book**

**{**

**public int Id { get; set; }**

**public string Title { get; set; }**

**public string Author { get; set; }**

**public string Genre { get; set; }**

**public string ISBN { get; set; }**

**public decimal Price { get; set; }**

**public int StockQuantity { get; set; }**

**public int? SupplierId { get; set; }**

**public virtual Supplier Supplier { get; set; }**

**public DateTime CreatedAt { get; set; } = DateTime.Now;**

**}**

**}**

**Data Access Layer (DAL)**

* Directory: BookHaven/DAL/
* Purpose:Handles direct interaction with the database.
* Key Characteristics:
  + Provides CRUD (Create, Read, Update, Delete) operations.
  + Encapsulates all database logic.
  + Each repository handles database operations for a specific entity.

**namespace BookHaven.DAL**

**{**

**class BookRepository**

**{**

**private readonly DatabaseHelper \_dbHelper = new DatabaseHelper();**

**public int CreateBook(Book book, SqlTransaction transaction = null)**

**{**

**try**

**{**

**string query = @"**

**INSERT INTO Books (Title, Author, Genre, ISBN, Price, StockQuantity, SupplierId, CreatedAt)**

**OUTPUT INSERTED.Id**

**VALUES (@Title, @Author, @Genre, @ISBN, @Price, @StockQuantity, @SupplierId, @CreatedAt)";**

**SqlParameter[] parameters = {**

**new SqlParameter("@Title", SqlDbType.NVarChar) { Value = book.Title },**

**new SqlParameter("@Author", SqlDbType.NVarChar) { Value = book.Author },**

**new SqlParameter("@Genre", SqlDbType.NVarChar) { Value = book.Genre ?? (object)DBNull.Value },**

**new SqlParameter("@ISBN", SqlDbType.NVarChar) { Value = book.ISBN },**

**new SqlParameter("@Price", SqlDbType.Decimal) { Value = book.Price },**

**new SqlParameter("@StockQuantity", SqlDbType.Int) { Value = book.StockQuantity },**

**new SqlParameter("@SupplierId", SqlDbType.Int) { Value = book.SupplierId ?? (object)DBNull.Value },**

**new SqlParameter("@CreatedAt", SqlDbType.DateTime) { Value = book.CreatedAt }**

**};**

**// Execute the query and return the inserted ID**

**object result = \_dbHelper.ExecuteScalar(query, parameters, transaction);**

**return result != null ? Convert.ToInt32(result) : -1;**

**}**

**catch (Exception ex)**

**{**

**Logger.LogError("CreateBook failed: " + ex.Message);**

**return -1;**

**}**

**}**

**}**

**}**

**Business Logic Layer (BLL)**

* Directory: BookHaven/BLL/
* Purpose: Contains core business logic and acts as an intermediary between the UI and the Data Access Layer.
* Key Characteristics:
  + Calls DAL methods and applies necessary validations, transformations, and business rules.
  + Encapsulates business-related operations to ensure maintainability.
  + Calls corresponding repository methods from the DAL.
  + Ensures business rules like "Stock cannot be negative" are applied.
  + Handles transactions when multiple operations need to be executed together.

**namespace BookHaven.BLL**

**{**

**class BookService**

**{**

**public int CreateBook(Book supplier, SqlTransaction transaction = null)**

**=> \_bookRepo.CreateBook(supplier, transaction);**

**}**

**}**

**User Interface Layer (UI)**

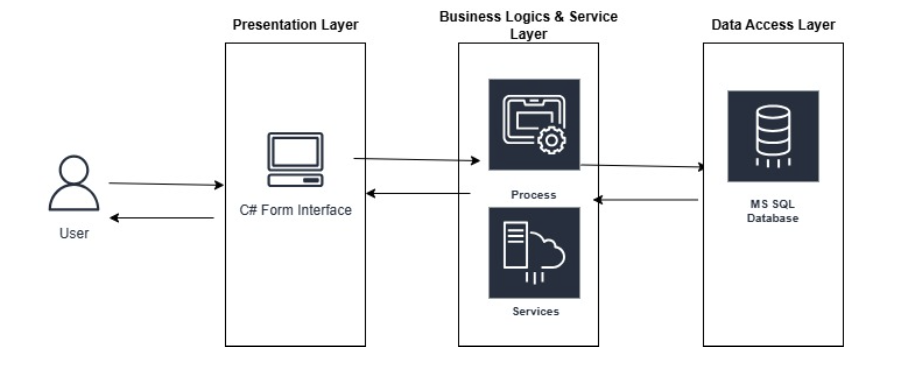
* Directory: BookHaven/UI/
* Purpose: Contains Windows Forms (.cs files) for interacting with users.
* Key Characteristics:
  + Displays data using DataGridView and other UI components.
  + Captures user input and sends it to the BLL for processing.
  + Calls BLL services and updates UI elements accordingly.
  + Uses Events and Event Handlers for user interactions.

**Breakdown of the Architecture**

| Layer | Responsibilities |
| --- | --- |
| Models (Entities) | Defines data structures |
| Data Access Layer (DAL) | Handles database queries and transactions |
| Business Logic Layer (BLL) | Implements business rules and validations |
| User Interface Layer (UI) | Manages user interactions (Windows Forms) |
| Reports Layer | Generates structured reports |
| Utilities Layer | Provides common utilities (logging, encryption) |
| Seeders Layer | Initializes database with default data |
| Enums Layer | Defines constants and enumerations |

**UML Diagrams**

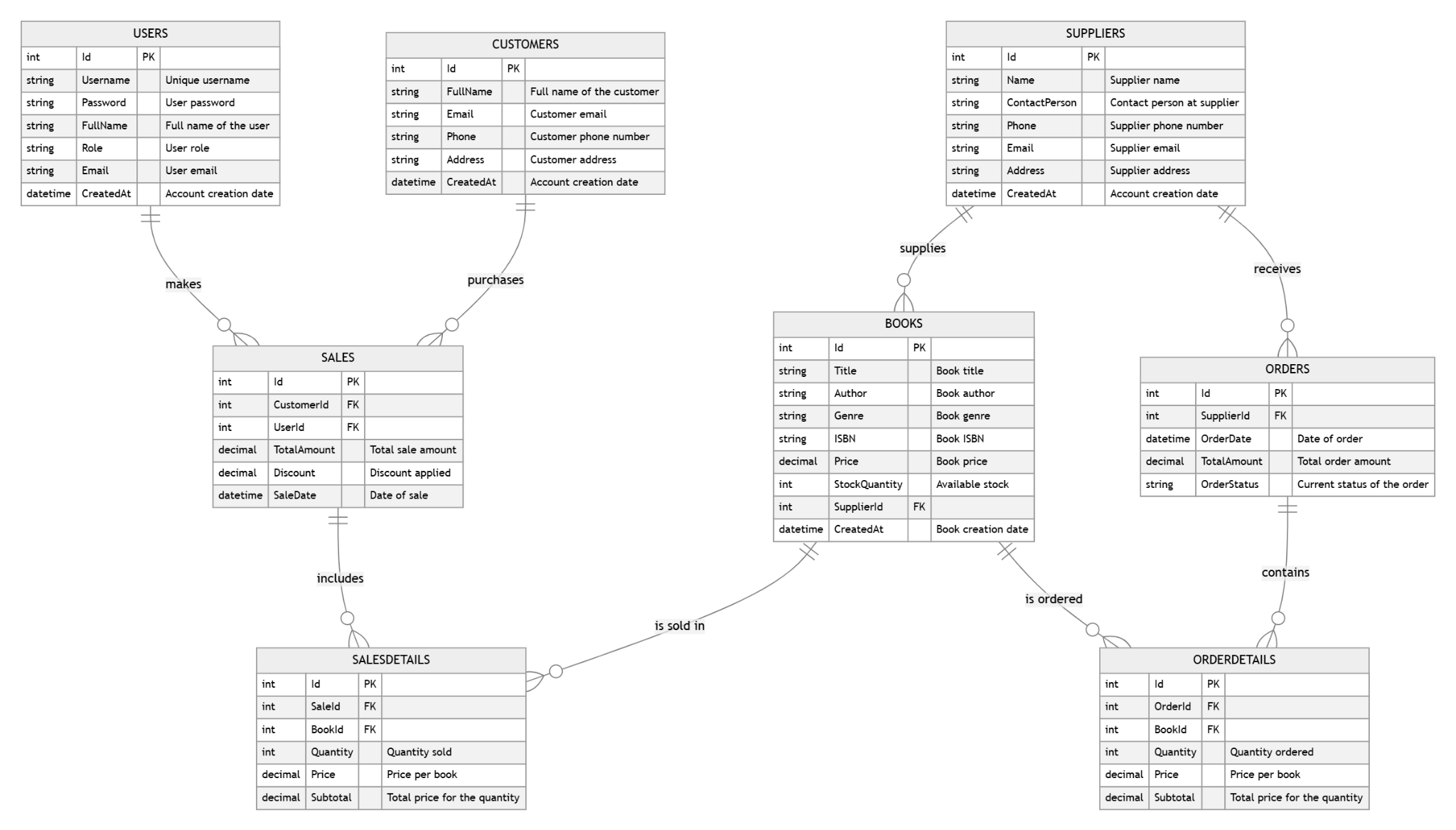
**System Architecture**

****

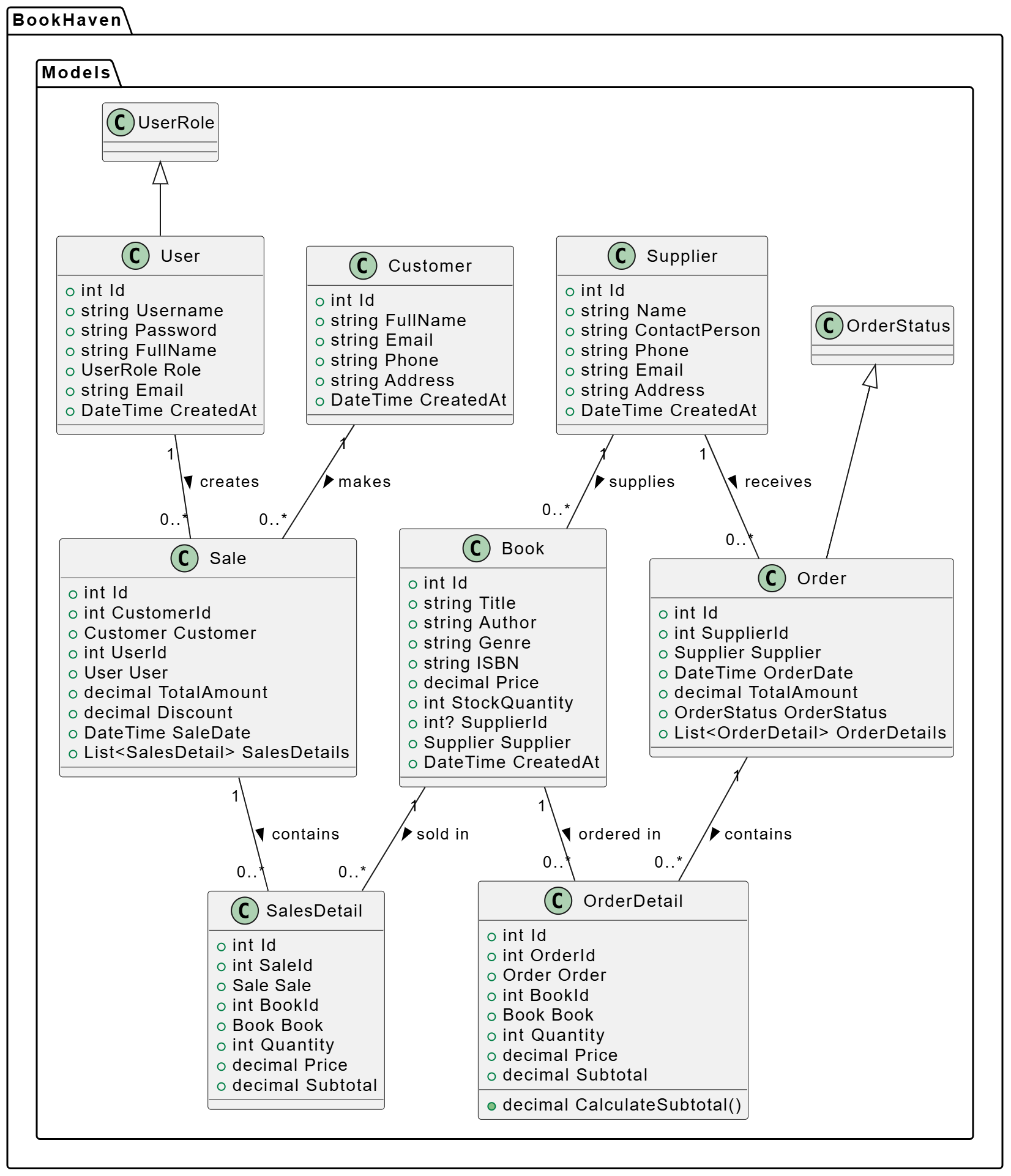
**Use Case Diagram**

****

**ER Diagram**

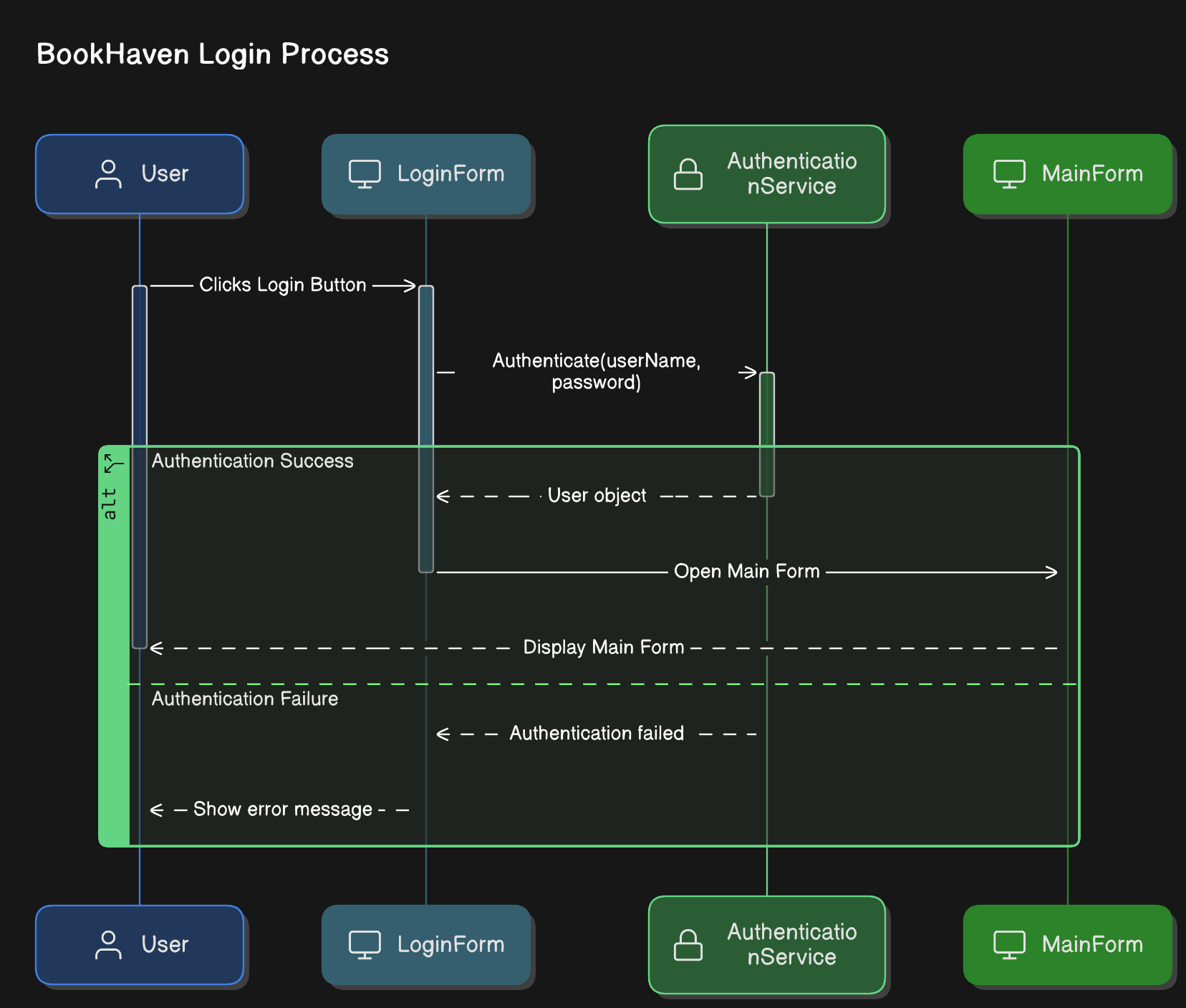
****

**Class Diagram**

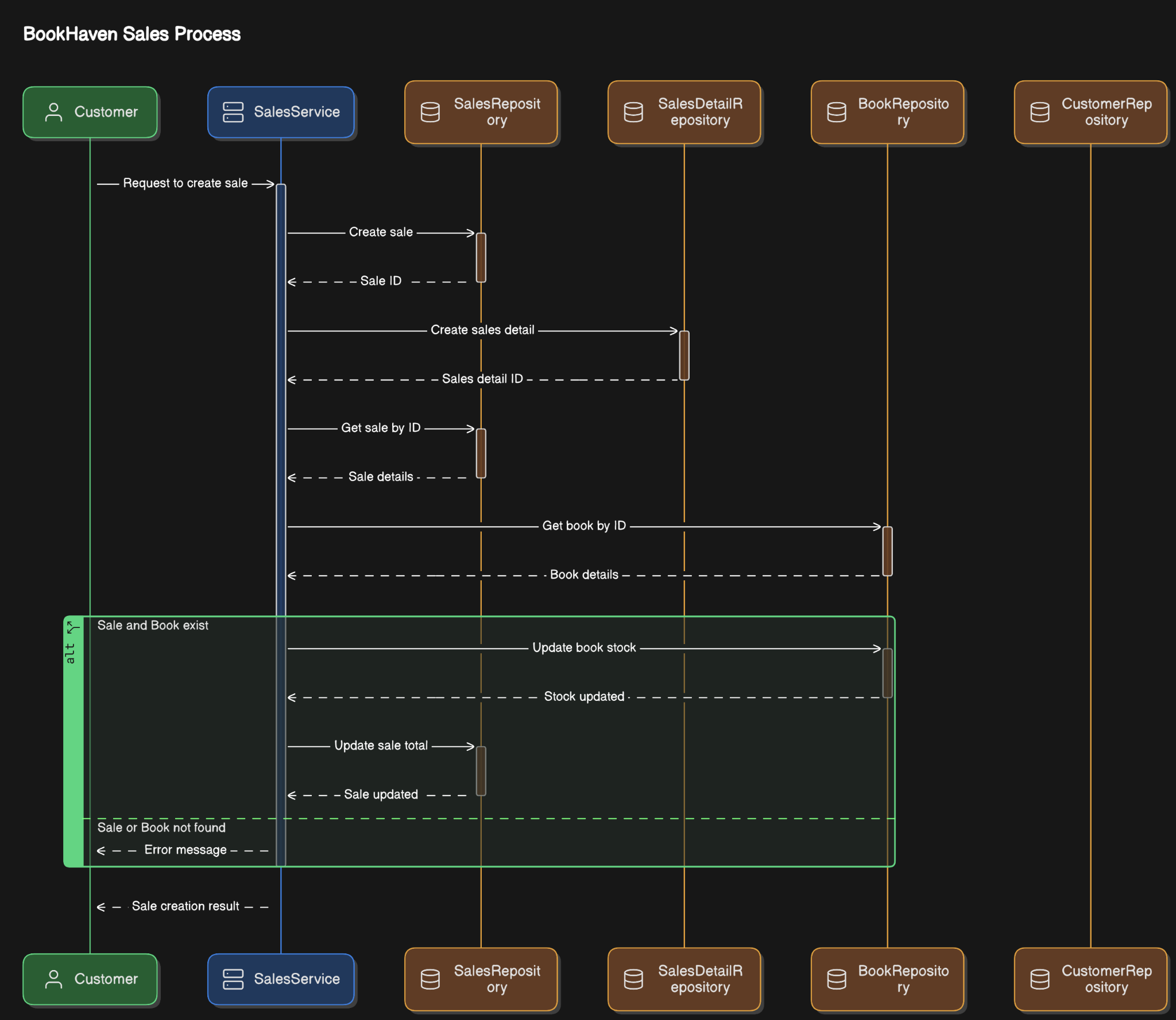
****

**Sequence Diagram**

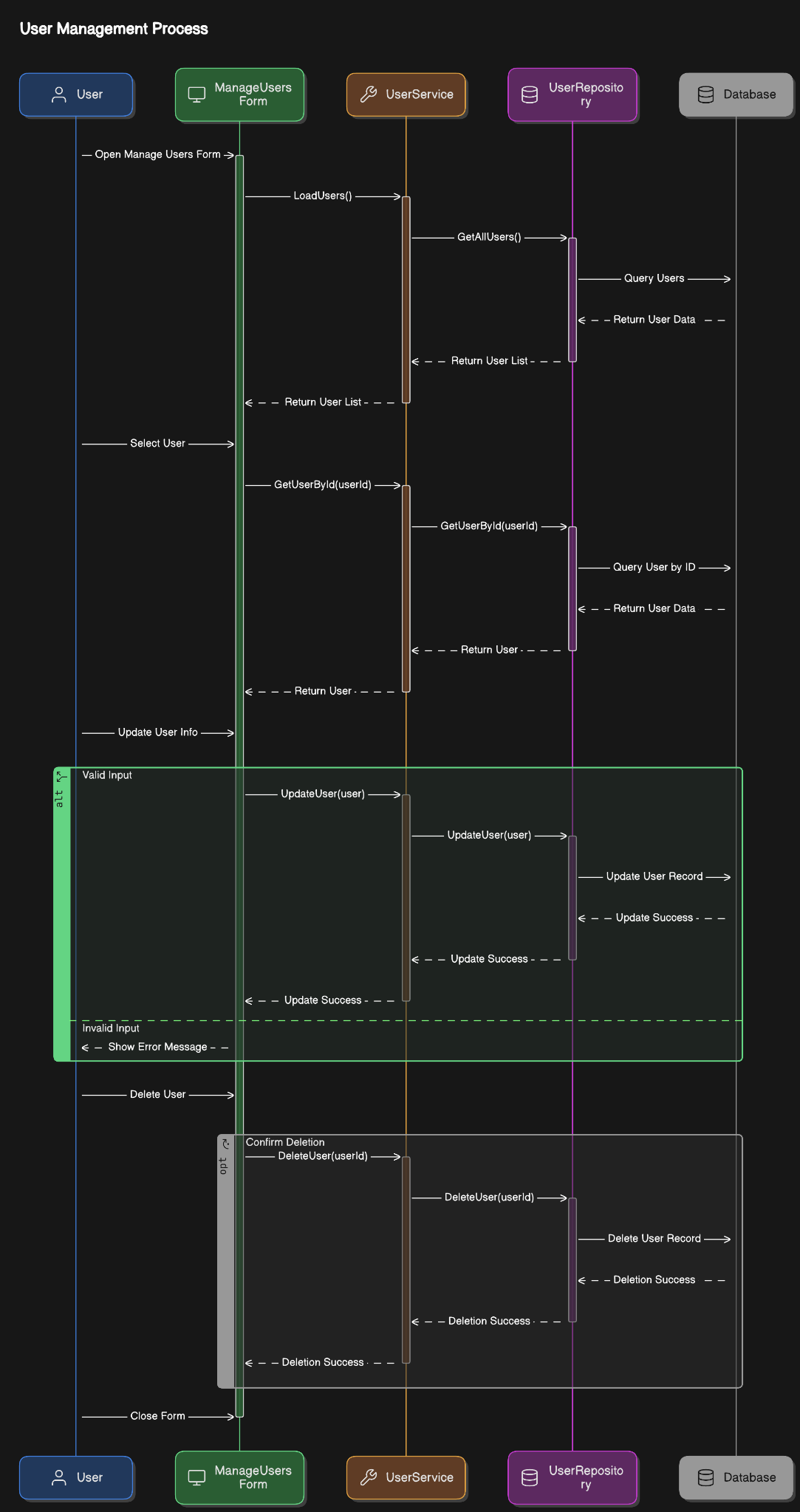
**Sequence Diagram for Login**

****

**Sequence Diagram for Sales Management**

****

**Sequence Diagram for User Management**

****

**Own Reflection of My Experience**

Developing the BookHaven bookstore management system was both a challenging and rewarding experience. As a beginner-level C# developer, this project helped me deepen my understanding of Windows Forms development, C# programming, and the MVC architecture.

**Key Learnings**

* Understanding MVC Architecture
  + Initially, structuring the project using the Model-View-Controller (MVC) pattern was challenging, but as I progressed, I realized the importance of separating concerns for maintainability and scalability.
* Database Design and SQL Integration
  + Designing a relational database and writing SQL queries were essential skills I developed.
* Authentication and Role-Based Access Control
  + Implementing user authentication and role-based access for Admin and SalesClerk roles gave me hands-on experience with secure login systems, password hashing, and user role management.
* Error Handling and Debugging
  + Debugging errors, especially in database connections and UI interactions, was a crucial learning experience. Using try-catch blocks, logging errors, and debugging tools in Visual Studio improved my troubleshooting skills.
* Windows Forms UI Design
  + Designing a user-friendly interface using Windows Forms was a challenge. I learned how to work with DataGridView, Forms, and Charts.

**Challenges Faced**

* Handling Database Transactions: Ensuring that multiple operations (such as adding a sale and updating inventory) were correctly executed required learning about transactions.
* Implementing Role-Based Access Control: Restricting features based on user roles was a new concept, requiring proper authorization logic.

**Future Improvements**

If I had more time, I would like to:

* Migrate the application to a web-based solution using ASP.NET Core.
* Implement an API layer to make it extendable.
* Enhance security measures with OAuth or JWT authentication.
* Integrate barcode scanning for faster book searching.

**Final Thoughts**

This project has been an excellent learning experience in C# programming, database management, and software development best practices. I gained confidence in my ability to develop a functional application independently, and I am motivated to continue improving my skills for future projects.

**Additional Resources**

For access to the source code and further details about the project, please visit the GitHub repository:

Repository Link: https://github.com/kasunvimarshana/BookHaven