**INVENTORY MANAGEMENT SYSTEM**



**Submitted By:**

Group Members:

Muhammad Ali 233510 BSCS B

Muhammad Hassan 233516 BSCS B

Muhammad Haroon 233504 BSCS B

**Submitted To:** Mam Aatka Ali

**Department of Computer Science**Air University Multan Campus

Contents

[1. Evaluation: 3](#_Toc186773602)

[2. Introduction: 4](#_Toc186773603)

[3. Objectives: 5](#_Toc186773604)

[4. Features: 5](#_Toc186773605)

[5. Business Logic: 6](#_Toc186773606)

[6. Flow Chart: 6](#_Toc186773607)

[7. Class Diagram: 7](#_Toc186773608)

[8. Implementation: 8](#_Toc186773609)

[9. Conclusion and Results: 12](#_Toc186773610)

10.References:…..………………………………………..12  
11.Appendices:..………………………………………….13

# Evaluation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Student Name** | **Roll No** | **Design (20%)** | **Functionality (30%)** | **Code Efficiency (20%)** | **Documentation & Presentation (30%)** | **Marks Obtained** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Introduction:**

**2.1 Overview of the Project:**

This project is a Windows-based application developed using .NET Framework, focusing on managing a store's operations efficiently. It includes functionalities for user and admin authentication, category and order management, supplier handling, and product management. The system ensures a streamlined workflow for both administrators and regular users, catering to the operational needs of a store.

**2.2 Problem Statement and Background:**

Traditional store management systems often rely on manual processes or outdated software, leading to inefficiencies and errors in managing orders, suppliers, and user interactions. This project aims to provide a robust solution by integrating modern software techniques to digitize and automate store management tasks.

**2.3 Significance of the Problem:**

Effective store management is crucial for ensuring customer satisfaction, operational efficiency, and profitability. This system addresses the inefficiencies of manual processes, reducing the scope for errors and enhancing productivity.

1. **Objectives:**

**3.1 Primary Goals:**

* Develop a secure login system for admins and users.
* Implement functionalities for managing categories, products, orders, and suppliers.
* Ensure a user-friendly interface for streamlined navigation.

**3.2 Expected Outcomes:**

* A fully functional store management system.
* Enhanced efficiency in managing store operations.
* A scalable application that can be enhanced further.

# Features:

**4.1 Key Functionalities:**

* Admin and user login.
* Product category management.
* Order placement and tracking.
* Supplier management.
* Dynamic user interface with real-time updates.

**4.2 Innovative Aspects:**

* Modular design for easy future enhancements.
* Integration of data validation to ensure reliable operations.
* Secure handling of sensitive user information.

# Business Logic:

**5.1 Description of Core Logic:**

The application uses object-oriented principles to manage entities like users, products, and orders. Data is processed through centralized logic layers to ensure consistency and integrity.

**5.2 Problem-Solving Approach:**

* Use of structured data validation for user inputs.
* Modular implementation of functionalities to simplify debugging and scalability.
* Implementation of efficient algorithms for data management.

# Flow Chart:

**6.1 Application Flow Representation:**

[Start] **-->** [Login Page] **-->** [Admin/User Role Check]

Admin: [Admin Dashboard] --> [Manage Categories] **-->** [Manage Orders] **-->** [Manage Suppliers]

User: [User Dashboard] **-->** [Place Order] **-->** [View Order History] **-->** [Logout]

**6.2 Annotations and Explanation:**

The flow chart represents the high-level interaction of users (admins and regular users) with the system, detailing the sequence of operations available for each role.

# Class Diagram:

**7.1 Overview of Class Structure:**

Key classes include:

* User
* Admin
* Order
* Product
* Supplier

**7.2 Class Relationships:**

* **User** and **Admin** inherit from a base class Person.
* **Order** and **Product** have a one-to-many relationship.
* **Supplier** manages multiple **Products**.

**7.3 Associations and Inheritance:**

**Class: Person**

|-- User

|-- Admin

**Class: Order**

|-- Products

|-- User

**Class: Supplier**

|-- Products

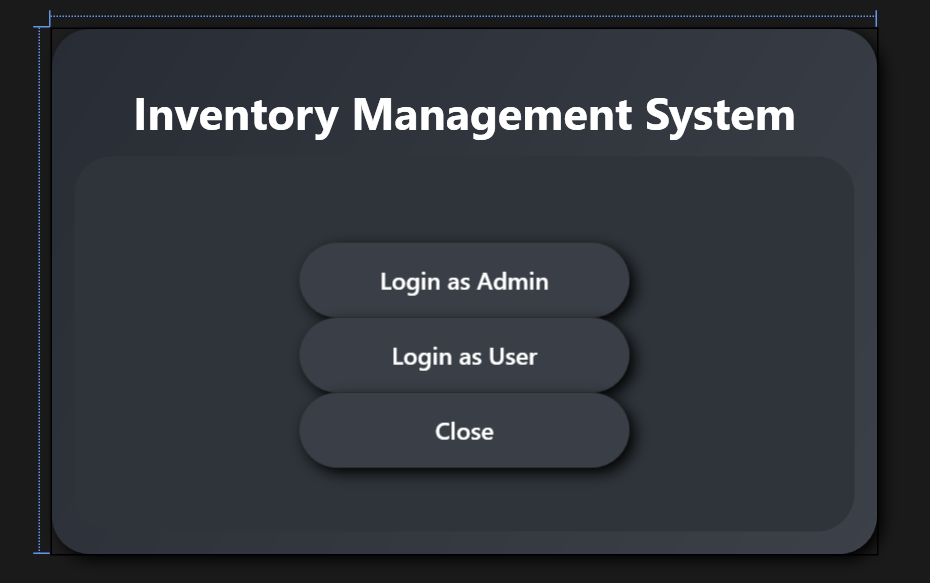
# Implementation:

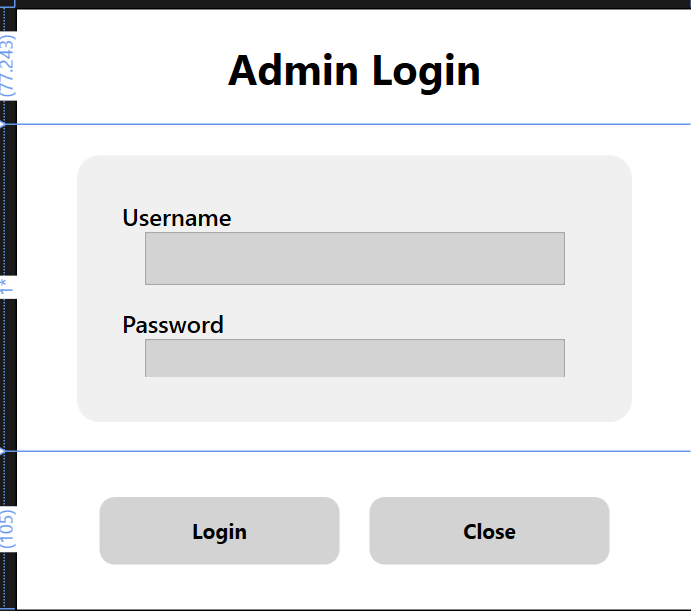
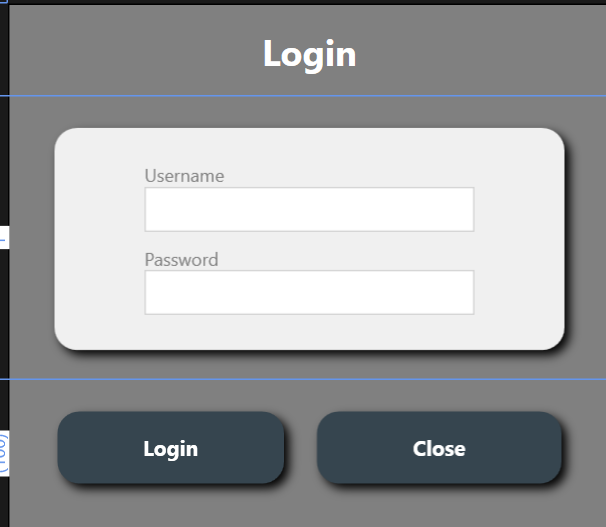
**8.1 Development Process:**

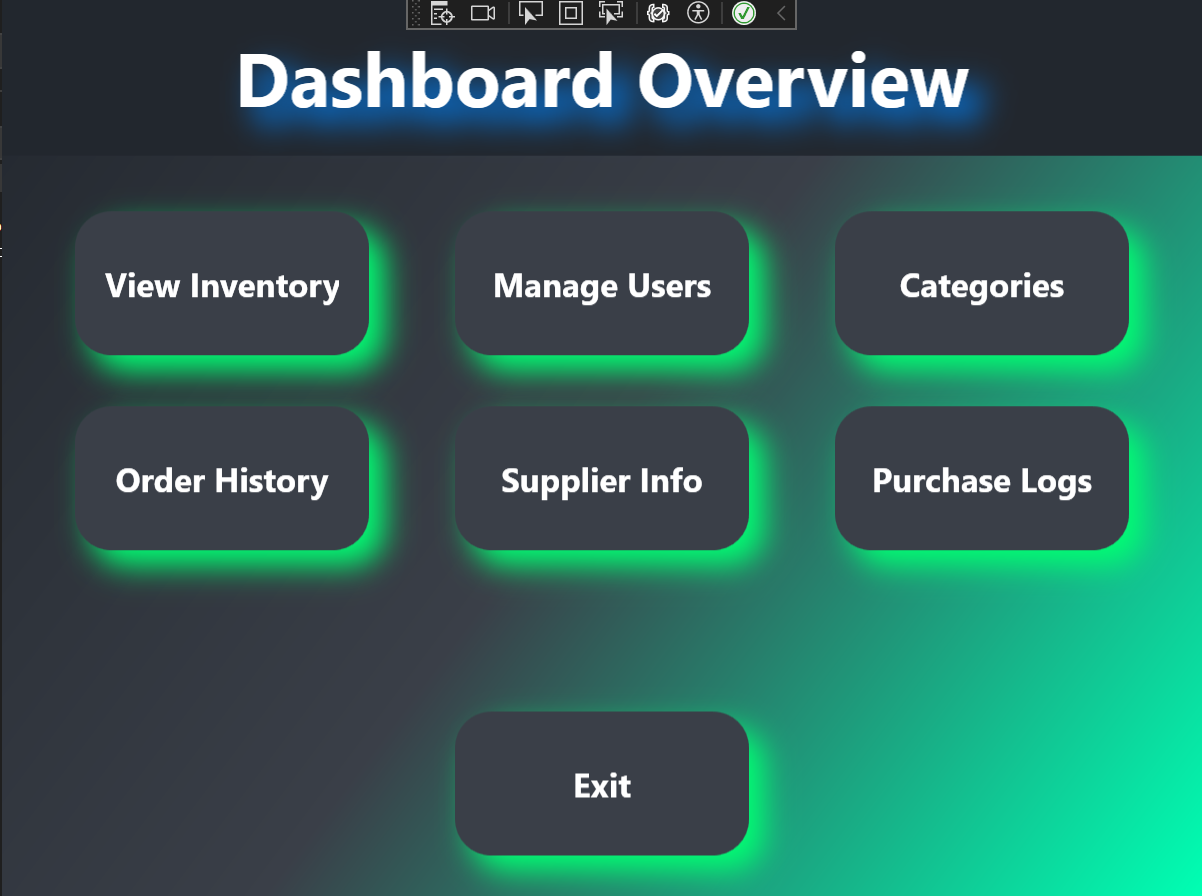
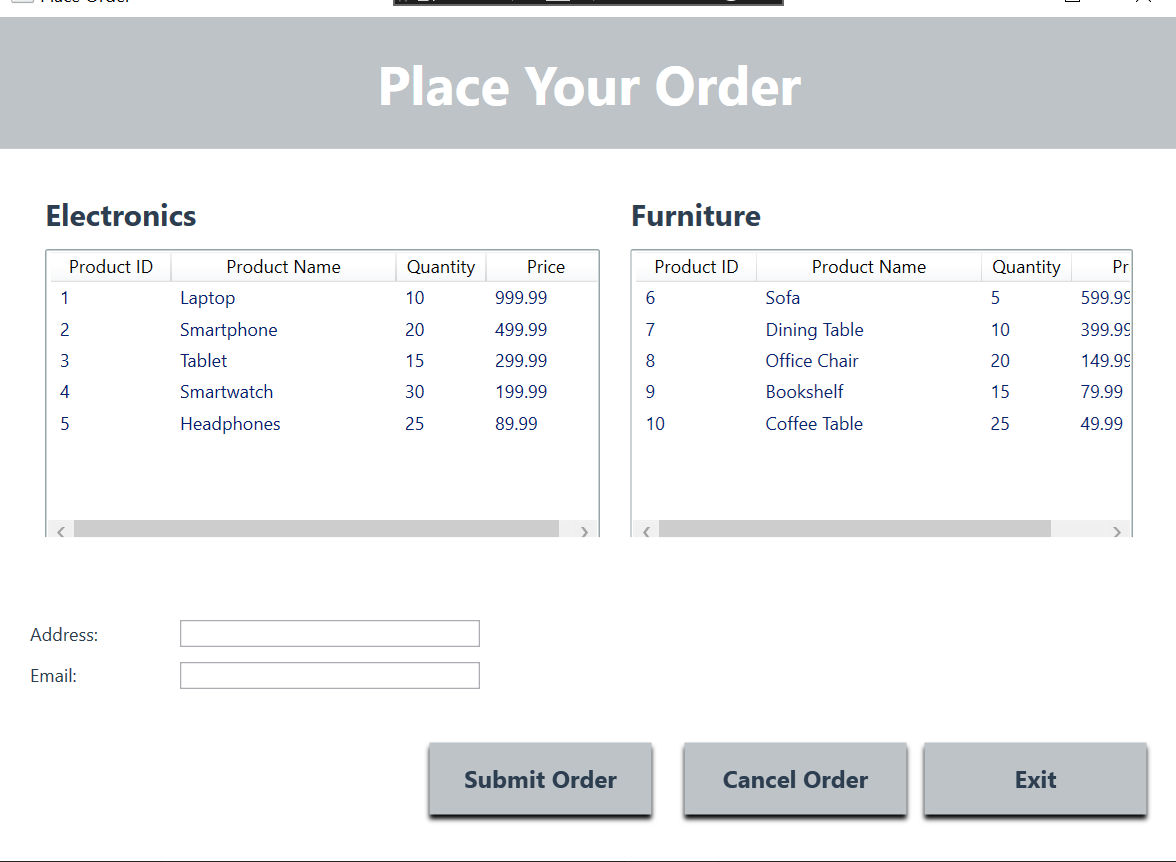
The application was developed iteratively, starting with the core functionalities, followed by interface design and testing.

**8.2 Screenshots of all User Interfaces:**

Screenshots include:

* **Login Page:** 



* **Admin Dashboard:**  **User Dashboard:** 
* **Order Management Page:** 

**8.3 Tools and Technologies Used:**

* **IDE**: Visual Studio
* **Language**: C#
* **Framework**: .NET
* **Database**: SQL Server

**8.4 Challenges and Solutions:**

* **Challenge**: Ensuring secure user authentication.
* **Solution**: Implemented robust password hashing and validation techniques.
* **Challenge**: Managing dynamic UI updates.
* **Solution**: Used XAML bindings for real-time updates.

# Conclusion and Results:

**9.1 Project Outcomes Summary:**

The project successfully meets its objectives by providing a fully functional and user-friendly store management system.

**9.2 Comparison of Objectives and Results:**

All primary goals were achieved, including secure login, dynamic UI, and robust data management.

**9.3 Future Enhancements:**

* Add mobile application support.
* Integrate analytics for better decision-making.
* Implement multi-language support for broader usability.  
    
  **10. References:**

**10.1 Cited Resources:**

* Microsoft .NET Documentation
* C# Programming Guide

**10.2 Further Reading:**

* Advanced UI Design in XAML
* SQL Server Optimization Techniques

**11. Appendices:**

**11.1 Additional Supporting Documents:**

Github Repository Link:

https://github.com/MuhammadAli-A/VisualProgrammingLab

**11.2 Code Snippets:**

// Example: User Login Validation

public bool ValidateUser(string username, string password) {

return users.Any(u => u.Username == username && u.Password == HashPassword(password));

}

// Example: Adding a Product

public void AddProduct(Product product) {

products.Add(product);

}

// Example: Placing an Order

public void PlaceOrder(User user, List<Product> products) {

var order = new Order {

User = user,

Products = products,

OrderDate = DateTime.Now

};

orders.Add(order);