HD IN COMPUTING AND SOFTWARE ENGINEERING

Service Oriented Computing - CSE5013

DILSHAN GUNASEKARA

CL/HD/CSE/CMU/120/05

turnitin.docx

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Table of Contents

Acknowledgement…………………………………………………………………………...3

Executive Summary 3

Introduction.. 5

Task 01 6

In-Depth Comparison of Monolithic vs. Service-Oriented Architecture (SOA) for Cozy Comfort 6

Monolithic Architecture 6

Service-Oriented Architecture (SOA):.. 7

Comparative Analysis: SOA vs. Monolithic for Cozy Comfort 8

Task 02 9

System Demonstration 9

Task 03 16

Task 04 33

Deployment Strategies for Cozy Comfort SOC Solution 33

Traditional Server Deployment... 33

Docker Containerization 33

Kubernetes (K8s) Orchestration 33

Serverless Computing 34

Optimal Deployment Strategy for Cozy Comfort.. 34

Conclusion 35

References... 36

Acknowledgement

I would want to take this opportunity to express my gratitude to my instructor, Ms. Upeka , for her tremendous guidance and aid in getting my assignment completed. I wish to acknowledge our Service Oriented Computing tutorial lecturer for her invaluable assistance and direction throughout this project. It was also enlightening and gratifying for me, as they are very good at explaining the ideas of SOC. This course has given me practical learning that has very much updated my understanding of distributed systems, APIs, and deployment techniques. We appreciate your patience, guidance and support in all of us getting from theory to praxis.

Executive Summary

This project aims to modernize the inefficient, manual order management system of "Cozy Comfort," a blanket manufacturer, by designing a **Service-Oriented Architecture (SOA)** solution. The proposed system will replace error-prone email/spreadsheet workflows with **modular services** (Manufacturer, Distributor, Seller) connected via APIs, enabling real-time inventory checks, automated order processing, and seamless scalability.

Key Challenges Addressed:

1. **Fragmented Communication:** Disjointed phone/email workflows cause delays and errors.
2. **Poor Scalability:** Manual processes hinder business growth.
3. **Lack of Real-Time Data:** No centralized visibility into stock levels or production capacity.

Proposed Solution:

- **SOA Model:** Decoupled services for Manufacturer, Distributor, and Seller, communicating via RESTful APIs.
- **Technology Stack:**
  - **Backend:** .NET-based API services (C#, ASP.NET Core).
  - **Frontend:** Optional cross-platform client (e.g., Angular/React or .NET MAUI).

- **Expected Outcomes:**
    - **Efficiency:** Automated order routing reduces processing time.
    - **Scalability:** Independent services allow targeted scaling.
    - **Reliability:** Failures in one service (e.g., Distributor) don’t crash the entire system.

Architecture Justification:
A **monolithic architecture** was rejected due to its inflexibility and scalability limitations. **SOA** was selected for its:

- **Modularity:** Easier maintenance and updates.
- **Interoperability:** APIs enable integration with future partners (e.g., e-commerce platforms).
- **Resilience:** Isolated failures prevent system-wide downtime.

By adopting SOA, "Cozy Comfort" can achieve a **streamlined, error-resistant supply chain** while positioning itself for future digital expansion. This documentation outlines the architectural rationale, service design, and implementation plan to realize these benefits.

Introduction

“Cozy Comfort” is heavily dependent on emails and spreadsheets for order processing, which is difficult and problematic for their supply chain operations. These processes are antiquated and as a result they are resulting in lag times, inaccuracies, and scalability problems as the business expands.

This project proposes a Service Oriented Computing (SOC) solution to automate workflows between manufacturers, distributors, and sellers. Our goal is to provide real time inventory reporting and Fulfilment through modular on demand services connected through API.

The documentation will:

- Consider monolithic and service-oriented architecture
- Create .NET APIs for each stakeholder
- Create client applications to interact with the system.

The SOC also has the advantage that new partners can easily join, SOC services can be scaled and developed independently and provide improved fault tolerance. This has brought the supply chain into modern day practices to eliminate inefficiencies and prepare for future growth.

In the subsequent sections, we present our architectural analysis and implementation plan to change the operations of “Cozy Comfort” through technology.

Task 01

In-Depth Comparison of Monolithic vs. Service-Oriented Architecture (SOA) for Cozy Comfort

Monolithic Architecture

In monolithic architecture, the whole blanket supply chain system and application composed of the Manufacturer, Distributor and Seller would all be developed as one indivisible application. All functions including ordering, inventory and production control would exist in the same code base with a shared database and run-time environment.

Workflow in Monolithic Architecture:

- When a customer orders something, the Seller module first looks for local stock.
- If the stock is unavailable, the request is passed on internally to the Distributor module, which has its own inventory.
- If the Distributor does not have stock, the Manufacturer module is queried for progress on production.
- These responses traverse the same application layers they passed through before the Seller and ultimately customer.

Strengths of Monolithic Design:

- Less Complicated Dev & Deploy: Because everything is packaged together, the setup is much simpler and only requires a single deployment pipeline.
- Combination testing & debugging: Testing with all interfaces is straightforward, since these are all occurring within a single system.
- Performance Efficiency: Communication within the process occurs without network latency and this speeds up internal operations.

Drawbacks for Cozy Comfort:

- Poor scalability: Scaling the entire application, as in scaling in case of a spike in Seller orders, becomes a huge waste of resources as all parts must scale together.

- High Maintenance Burden: A small bug in the logic of the Distributor could bring the whole system down requiring all parties to wait for it to come back up.
- Inflexible Upgrades: When new functionality is developed (a real-time logistics tracker, for example) it can only be deployed by redeploying the entire monolith, with associated risk and disruption.
- Vendor Lock-in: Tight coupling makes it hard to replace/upgrade parts of the system (e.g. plug in the new Manufacturer's API).

Service-Oriented Architecture (SOA):

On the other hand, SOA splits the system into independent and interoperable services, each representing a distinct business function. For "Cozy Comfort," that would mean designing:

- Seller Service: Handles customer orders, communicates with end users.
- Distributor Service: Inventory, order and shipping logistics.
- Manufacturer Service: Monitors production capacity, availability of materials, and finished goods inventory levels.

Workflow in SOA:

- The Seller Service provides a placement API for orders. If stock is not found locally, it calls the Distributor Service API to confirm it.
- In the event that there is no inventory at the Distributor, it calls the Manufacturer Service API to ask about production time.
- The response from each service is asynchronous, and the Seller collects this data to inform the customer.

Advantages for Cozy Comfort:

- Elastic Scalability: Each service can scale independently. The Seller Service, for example, can be scaled during holiday sales without impacting the Manufacturer.
- Improved Maintainability: The logistics algorithm in the Distributor can be changed independently of the Seller's code, avoiding possible regressions.

- Business Agility: New partners (e.g., a online marketplace like Amazon) can be incorporated through standard APIs without having to overhaul the system.
- Fault Isolation: In the event the Manufacturer Service crashes because of a production database issue, both Seller and Distributor Services continue to work.

Challenges of SOA:

- Initial complexity: There is overhead in designing APIs, service contracts and handling interservice communications.
- Network Latency: Calls across remote services are delayed compared to in-process monolithic calls.
- Operational Overhead: Requires strong API gateways, service discovery and monitoring tools.

Comparative Analysis: SOA vs. Monolithic for Cozy Comfort

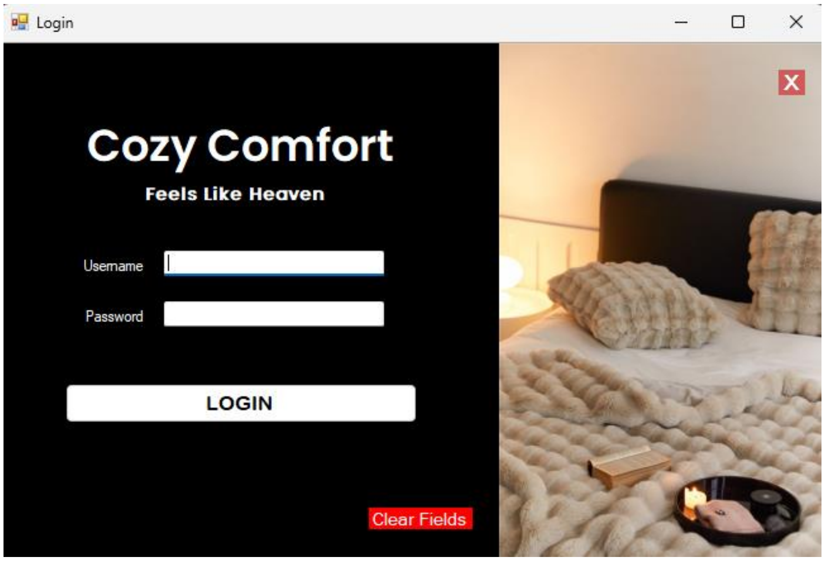
SOA is the better option to respond to the critical business needs of Cozy Comfort's distributed value chain. In contrast, monolithic systems can afford the advantage of a single deployment of code but become an unreasonable choice for a multi-stakeholder business such as Cozy Comfort where its manufacturers, distributors, and sellers are operating independently from each other. This scalability is possible because each of these units is independent; for example, when business is booming the seller service can scale up without impacting the manufacturer. As the architecture is loosely coupled, it is more easily maintainable in that the distributor logistics can be maintained without having to modify the entire system. SOA also has important fault isolation; a failure of a production system at the manufacturer will not cascade to prevent order taking at the sellers. Perhaps most importantly, SOA API driven architecture future proofs the business as it enables the addition of new sales channels or partners through standard interfaces as opposed to being a rigid monolithic system that would require entire system overhauls for additions like that. Therefore, these clear benefits of scalability, maintainability, and flexibility point to SOA as the obvious choice for enabling the growth of Cozy Comfort and resolving the process inefficiencies in place.

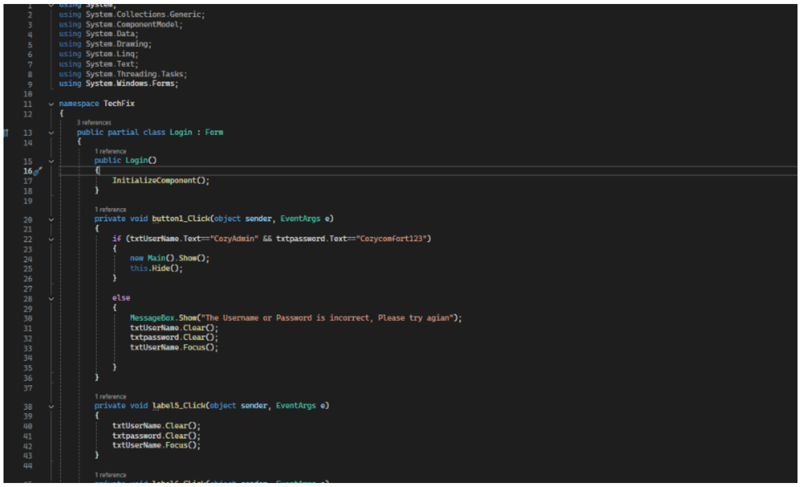
Task 02

System Demonstration

Here we will develop and deploy a Service-Oriented Computing (SOC) approach to help Cozy Comforts supply chain. It will automate order processing, allow tracking of inventory in real time, and serve as a centralized communication tool between manufacturers, distributors and sellers. Designed on SOC principles, the solution is scalable, maintainable, and reusable allowing the business to grow and will eliminate manual inefficiencies.

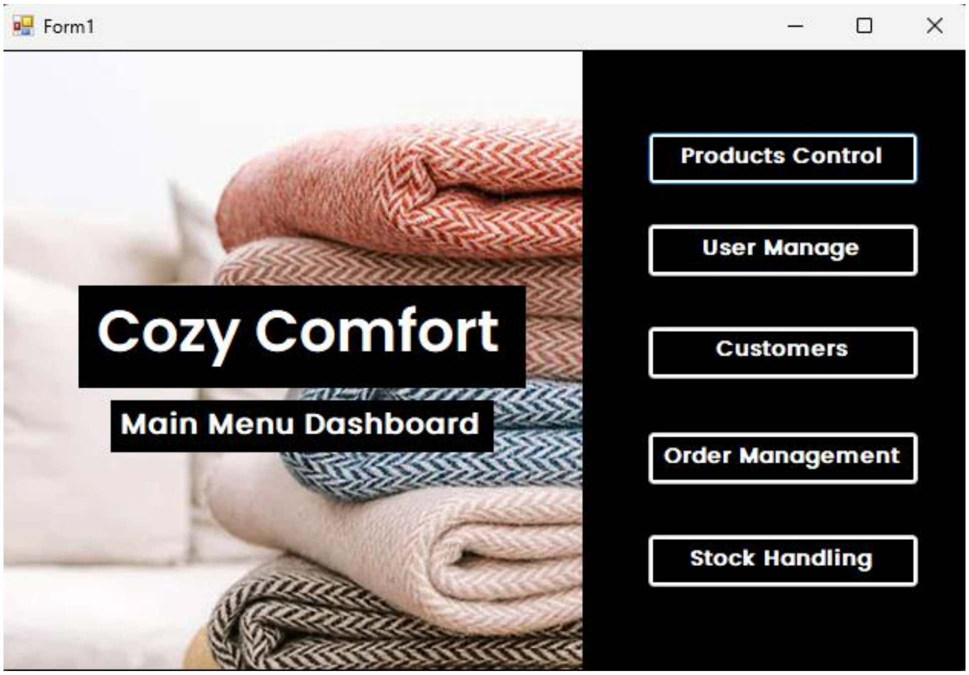
Login

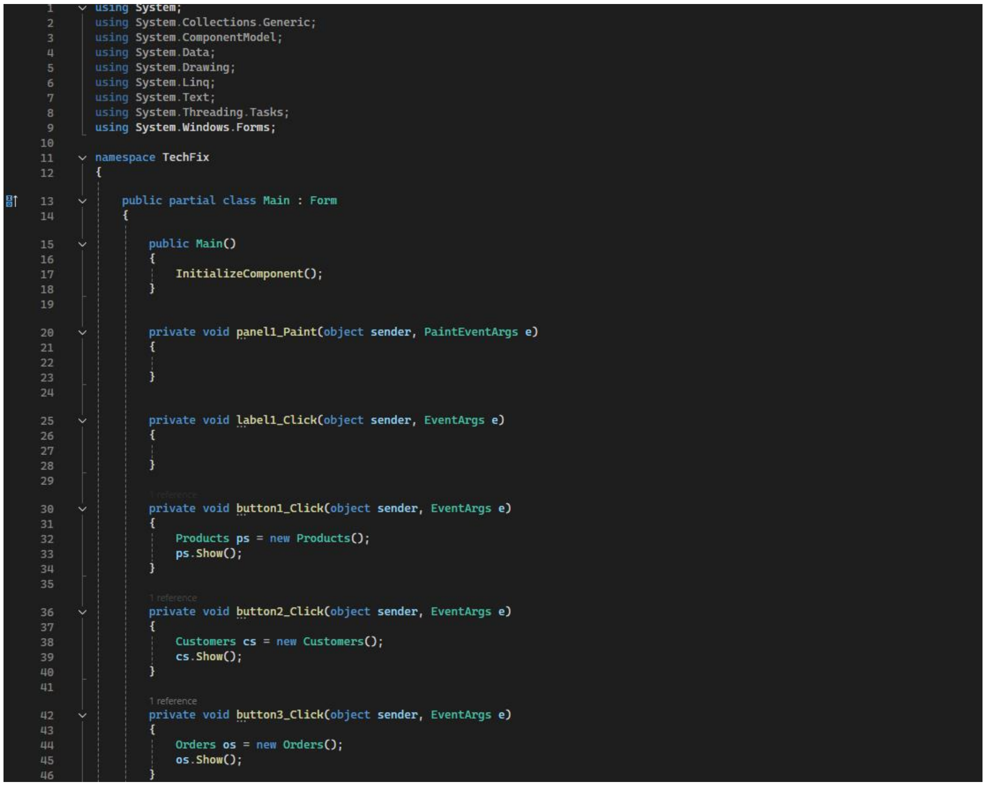




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TechFix
{
    3 references
    public partial class Login : Form
    {
        1 reference
        public Login()
        {
            InitializeComponent();
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            if (txtUserName.Text=="CozyAdmin" && txtpassword.Text=="Cozycomfort123")
            {
                new Main().Show();
                this.Hide();
            }

            else
            {
                MessageBox.Show("The Username or Password is incorrect, Please try agian");
                txtUserName.Clear();
                txtpassword.Clear();
                txtUserName.Focus();

            }
        }

        1 reference
        private void label5_Click(object sender, EventArgs e)
        {
            txtUserName.Clear();
            txtpassword.Clear();
            txtUserName.Focus();
        }

        1 reference
        private void label6_Click(object sender, EventArgs e)
```

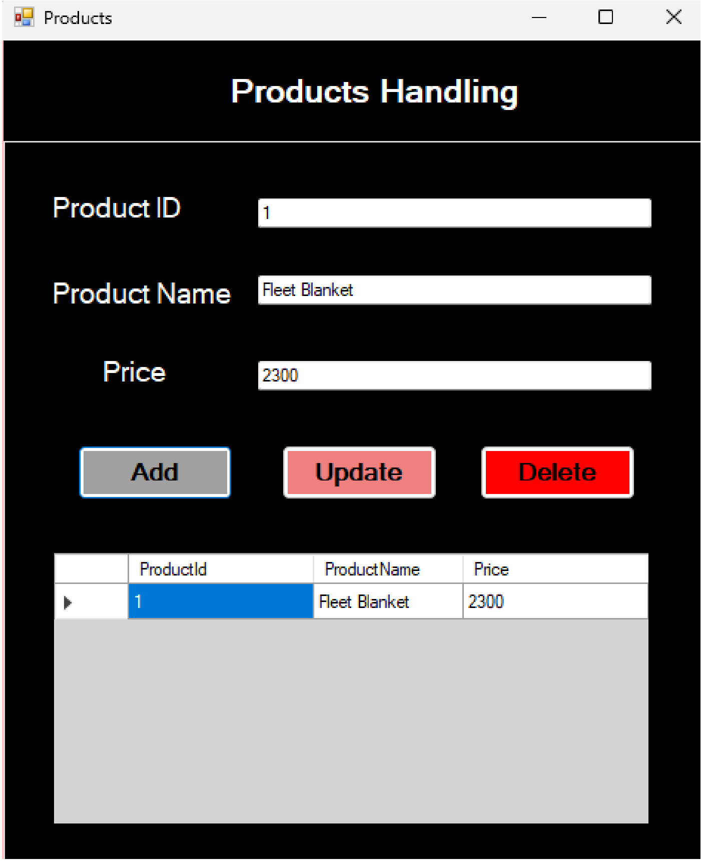
Main Dashboard





```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TechFix
{

    public partial class Main : Form
    {

        public Main()
        {
            InitializeComponent();
        }


        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }


        private void label1_Click(object sender, EventArgs e)
        {

        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            Products ps = new Products();
            ps.Show();
        }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            Customers cs = new Customers();
            cs.Show();
        }

        1 reference
        private void button3_Click(object sender, EventArgs e)
        {
            Orders os = new Orders();
            os.Show();
        }
```

Products Handling



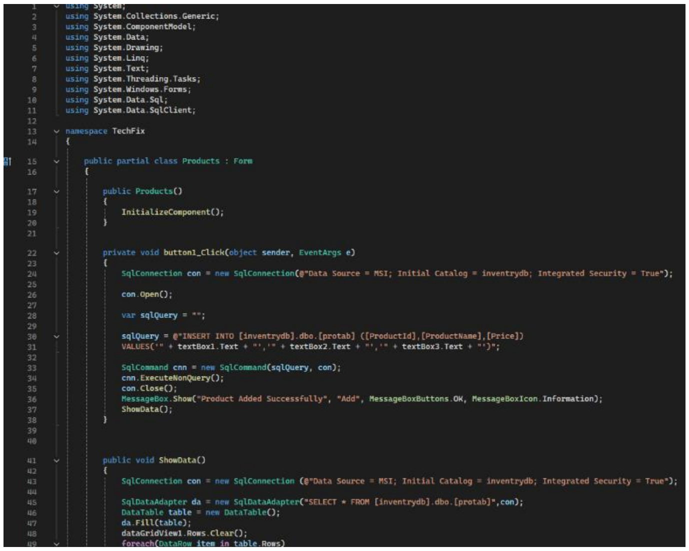
Products Handling

Product ID: 1

Product Name: Fleet Blanket

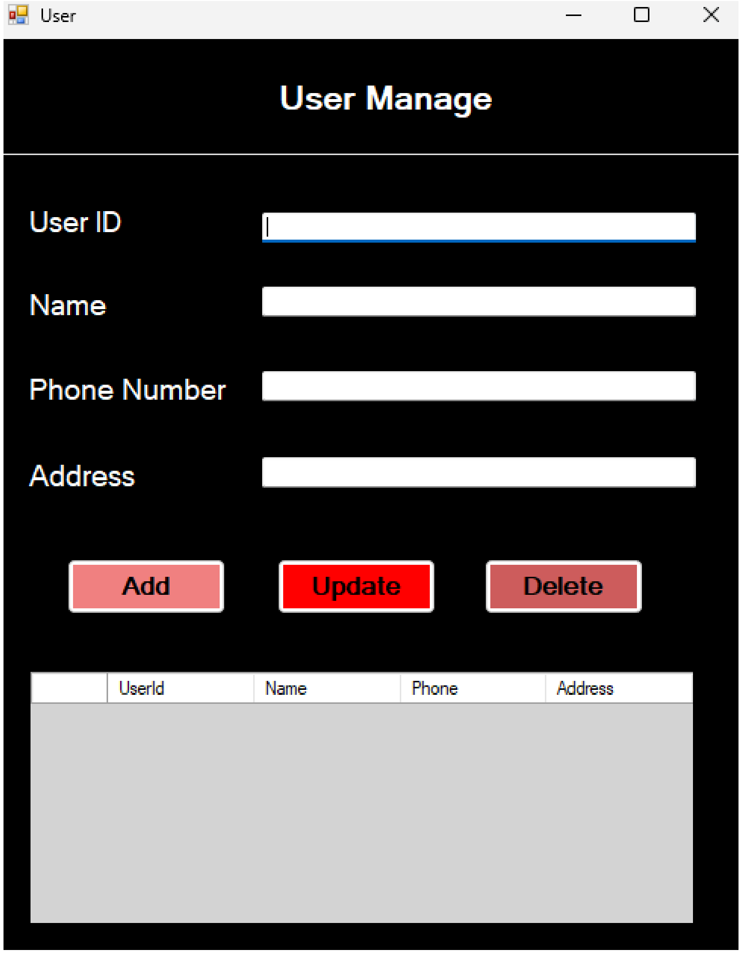
Price: 2300

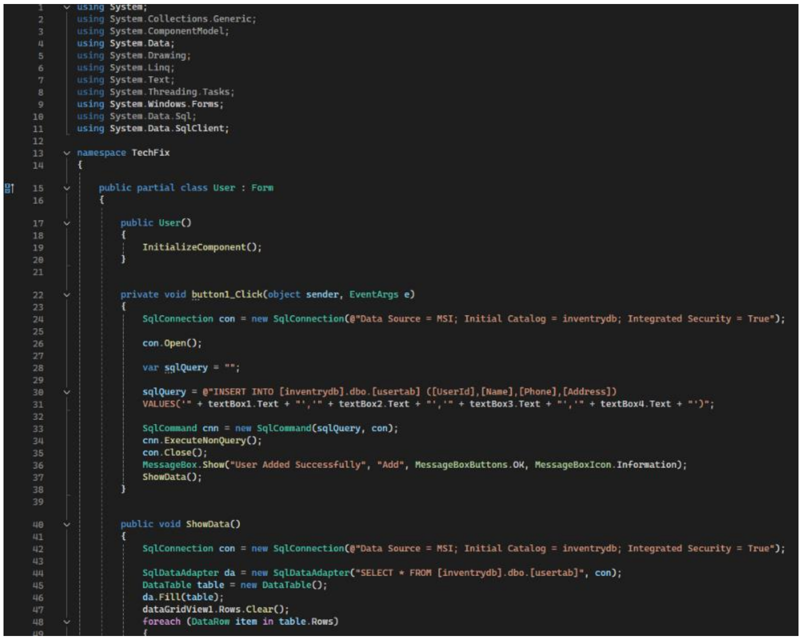
ProductId	ProductName	Price
1	Fleet Blanket	2300



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TechFix
{

    public partial class Products : Form
    {

        public Products()
        {
            InitializeComponent();
        }


        private void button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            con.Open();

            var sqlQuery = "";

            sqlQuery = @"INSERT INTO [inventrydb].dbo.[protab] ([ProductId],[ProductName],[Price])
            VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "')";

            SqlCommand cnn = new SqlCommand(sqlQuery, con);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Product Added Successfully", "Add", MessageBoxButtons.OK, MessageBoxIcon.Information);
            ShowData();
        }



        public void ShowData()
        {
            SqlConnection con = new SqlConnection (@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM [inventrydb].dbo.[protab]",con);
            DataTable table = new DataTable();
            da.Fill(table);
            dataGridView1.Rows.Clear();
            foreach(DataRow item in table.Rows)
```

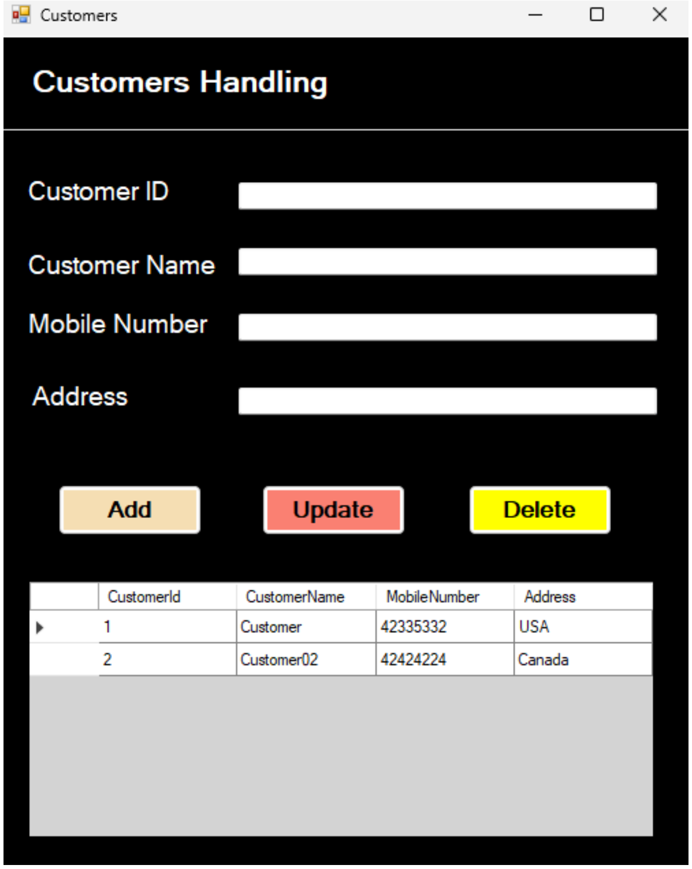
User Manage Interface



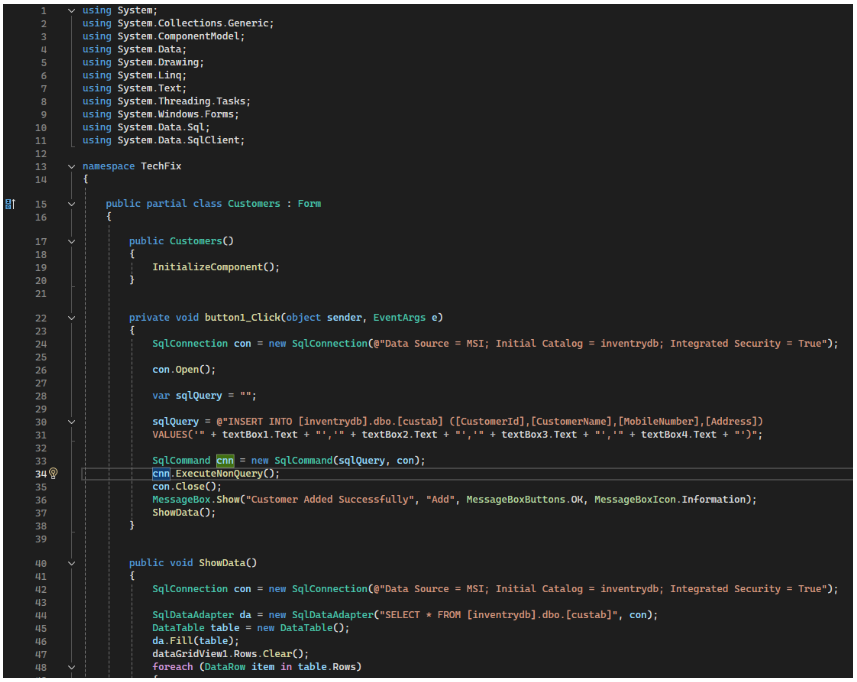


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TechFix
{

    public partial class User : Form
    {

        public User()
        {
            InitializeComponent();
        }


        private void button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            con.Open();

            var sqlQuery = "";

            sqlQuery = @"INSERT INTO [inventrydb].dbo.[usertab] ([UserId],[Name],[Phone],[Address])
            VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "')";

            SqlCommand cnn = new SqlCommand(sqlQuery, con);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("User Added Successfully", "Add", MessageBoxButtons.OK, MessageBoxIcon.Information);
            ShowData();
        }


        public void ShowData()
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM [inventrydb].dbo.[usertab]", con);
            DataTable table = new DataTable();
            da.Fill(table);
            dataGridView1.Rows.Clear();
            foreach (DataRow item in table.Rows)
            {
```

Customers Handling Interface

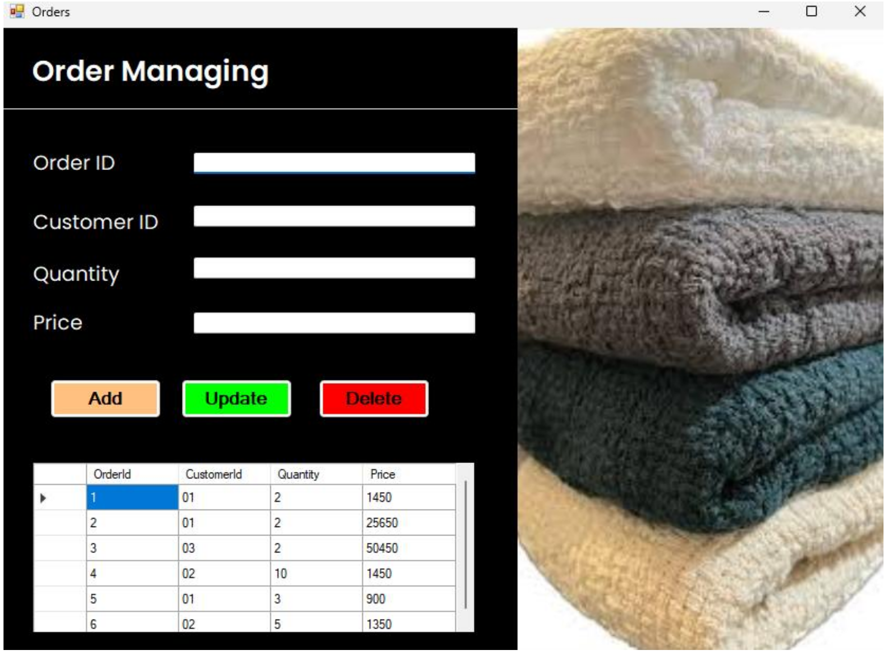


	CustomerId	CustomerName	MobileNumber	Address
▶	1	Customer	42335332	USA
	2	Customer02	42424224	Canada

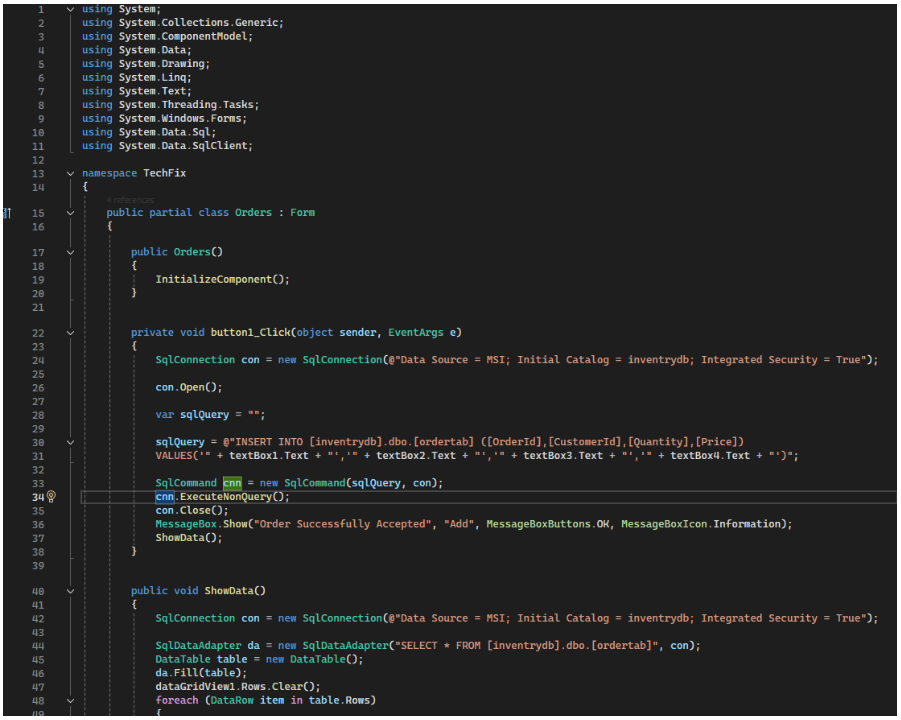


```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TechFix
{

    public partial class Customers : Form
    {

        public Customers()
        {
            InitializeComponent();
        }


        private void button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            con.Open();

            var sqlQuery = "";

            sqlQuery = @"INSERT INTO [inventrydb].dbo.[custab] ([CustomerId],[CustomerName],[MobileNumber],[Address])
            VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "')";

            SqlCommand cnn = new SqlCommand(sqlQuery, con);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer Added Successfully", "Add", MessageBoxButtons.OK, MessageBoxIcon.Information);
            ShowData();
        }


        public void ShowData()
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM [inventrydb].dbo.[custab]", con);
            DataTable table = new DataTable();
            da.Fill(table);
            dataGridView1.Rows.Clear();
            foreach (DataRow item in table.Rows)
```

Order Management Interface

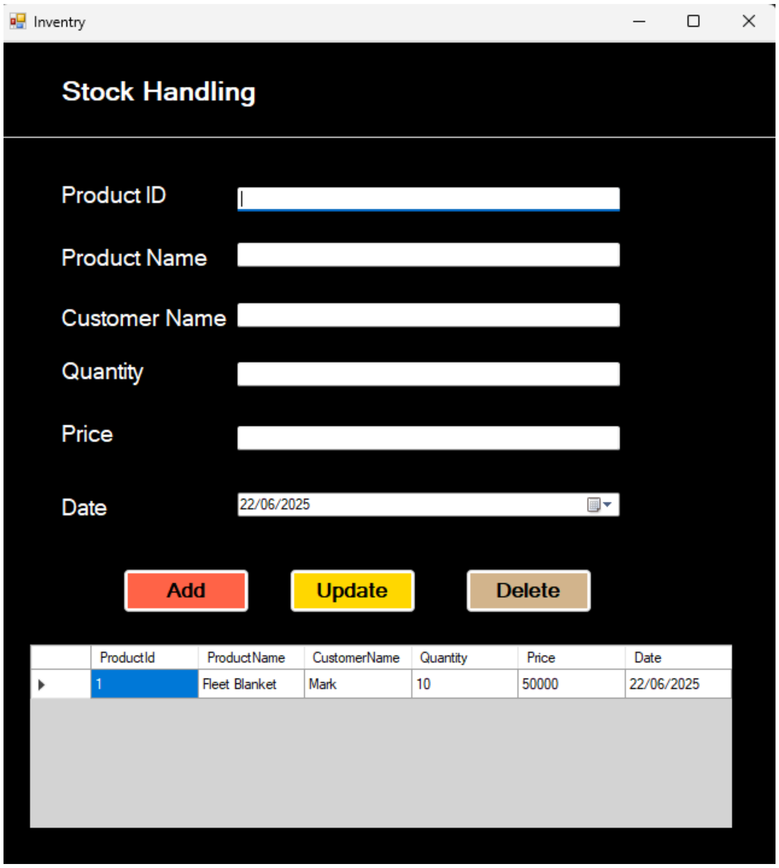


OrderId	CustomerId	Quantity	Price
1	01	2	1450
2	01	2	25650
3	03	2	50450
4	02	10	1450
5	01	3	900
6	02	5	1350



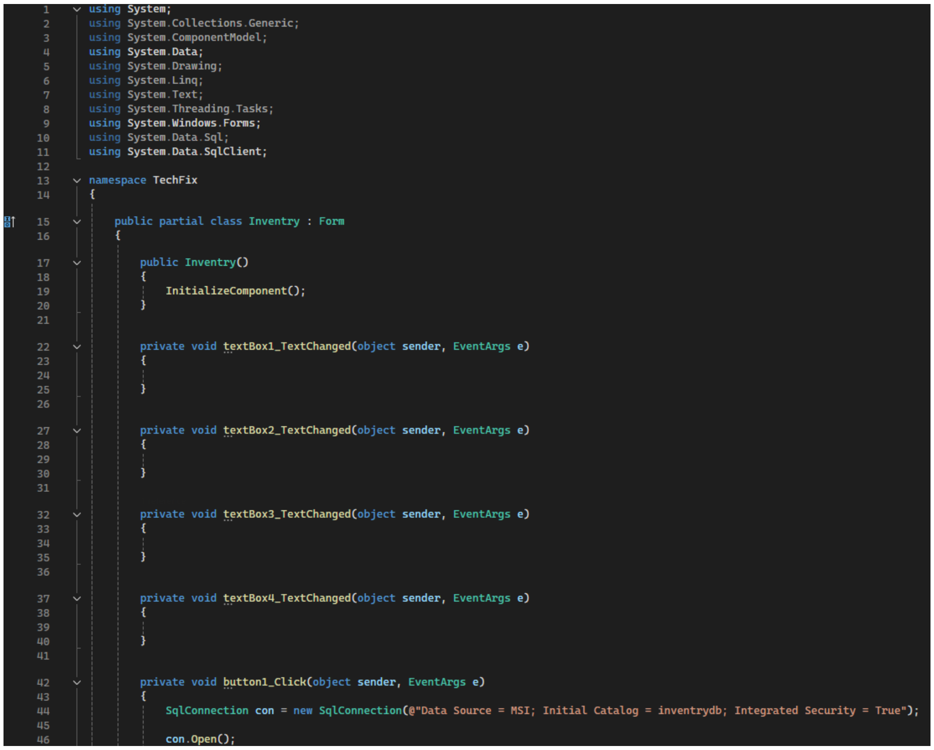
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TechFix
{
    4 references
    public partial class Orders : Form
    {

        public Orders()
        {
            InitializeComponent();
        }


        private void button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            con.Open();

            var sqlQuery = "";

            sqlQuery = @"INSERT INTO [inventrydb].dbo.[ordertab] ([OrderId],[CustomerId],[Quantity],[Price])
            VALUES('" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "')";

            SqlCommand cnn = new SqlCommand(sqlQuery, con);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Order Successfully Accepted", "Add", MessageBoxButtons.OK, MessageBoxIcon.Information);
            ShowData();
        }


        public void ShowData()
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM [inventrydb].dbo.[ordertab]", con);
            DataTable table = new DataTable();
            da.Fill(table);
            dataGridView1.Rows.Clear();
            foreach (DataRow item in table.Rows)
            {
```

Stock Handling Interface



Date 22/06/2025

	ProductId	ProductName	CustomerName	Quantity	Price	Date
▶	1	Fleet Blanket	Mark	10	50000	22/06/2025



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;

namespace TechFix
{
    public partial class Inventry : Form
    {
        public Inventry()
        {
            InitializeComponent();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source = MSI; Initial Catalog = inventrydb; Integrated Security = True");

            con.Open();
```

Task 03

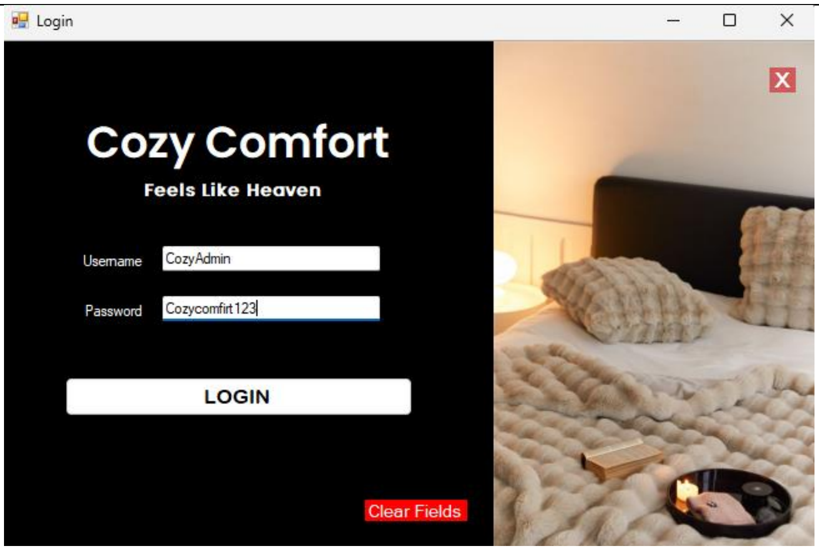
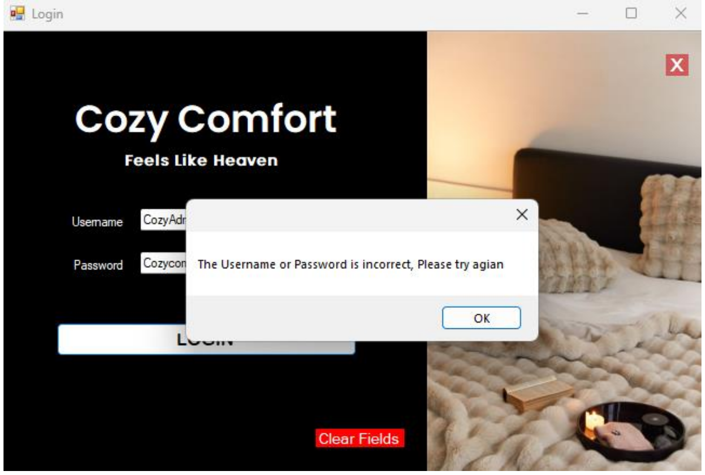
Our SOC solution will be tested and debugged thoroughly to ensure it fulfills all requirements of the supply chain and it works without issues. Here we describe our full quality assurance methodology to ensure a powerful and bug-free system.

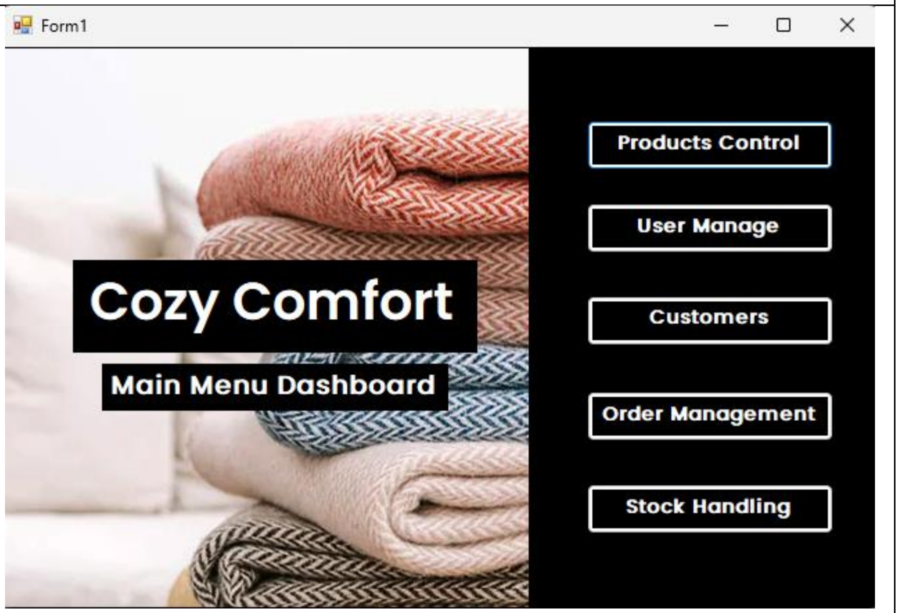
Test Plan

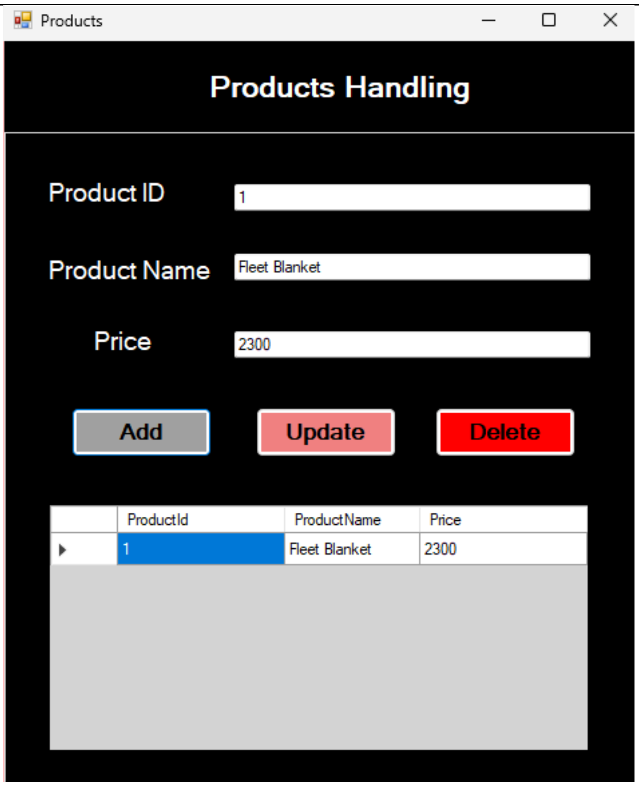
Test case ID	Test case name	Scenario	Expected result
T01	Login	Login with correct credentials	Successfully login to the system
T02	Main Dashboard loading	Add the correct credentials on the login panel to load the Main Dashboard	Successfully load the main dashboard
T03	Add product	Fill the product details and click on the add button	Successfully product added to the system
T04	Update product	Fill the product details to be updated	Successfully product updated
T05	Delete the product	Fill the product Id and to delete the product	Successfully delete the product from the system
T06	Add Customer	Fill out the customer details and click on the add button	Successfully add the customer
T07	Update customer	Fill out the customer details to be updated	Successfully update the customer

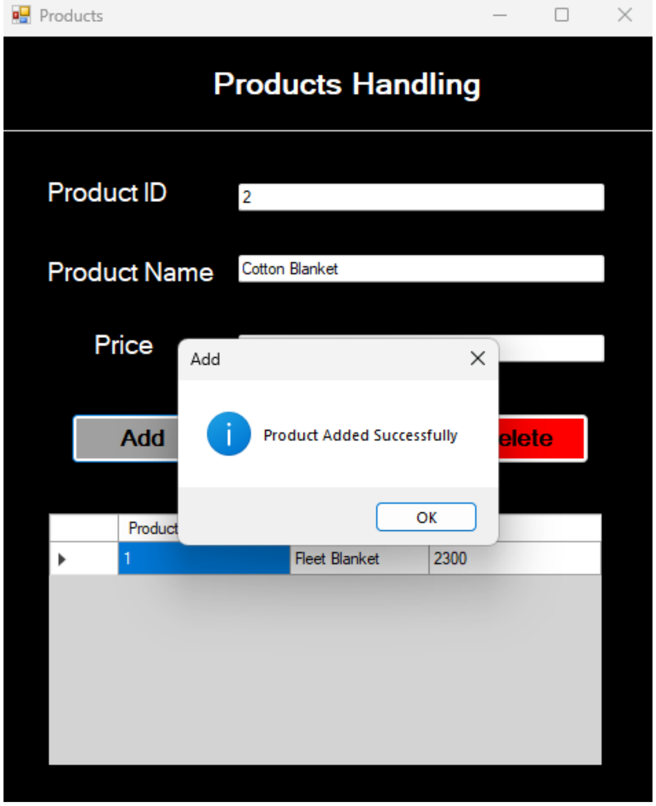
T08	Delete customer	Enter the customer ID to delete the customer	Successfully customer deleted
T09	Add order	Fill the order details to add an order	Successfully order added to the system
T10	Update order	Fill the updated details of the order and click on the update button	Successfully order added
T11	Delete an order	Enter the order ID to delete the order	Successfully order deleted form the system
T12	Add a user	Fill in the user details and click on the add button	User successfully added to the system
T13	Update user	Fill the updated details of the user and click on the update button	Successfully user updated.
T14	Delete user	Enter the user ID to delete the user	Successfully user deleted from the system

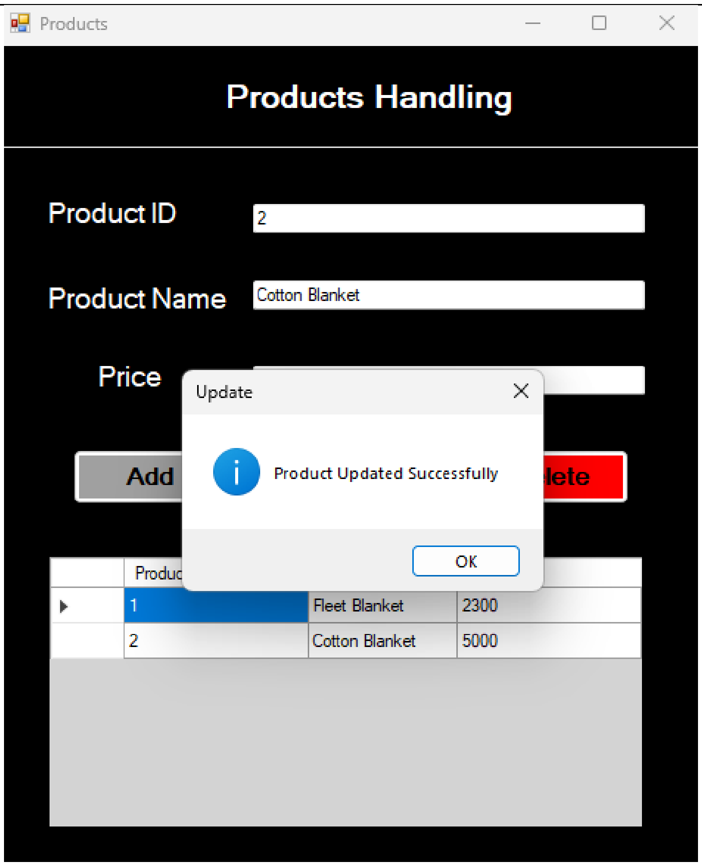
Test Cases

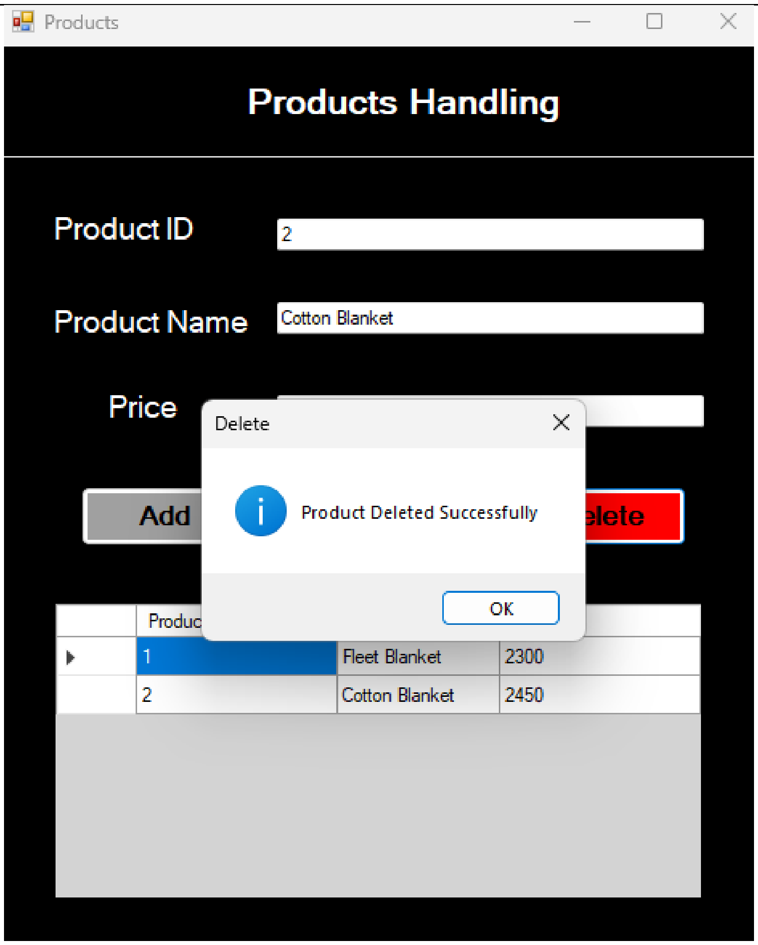
Test Case ID	T01
Test Case Name	Login
Result	Successfully login to the system
Screenshot	 Login Cozy Comfort Feels Like Heaven LOGIN Clear Fields Username CozyAdmin Password Cozycomfirt123  Login Cozy Comfort Feels Like Heaven OK Clear Fields Username CozyAdr Password Cozycon The Username or Password is incorrect, Please try agian

Test Case ID	T02
Test Case Name	Main Dashboard loading
Result	Successfully load the main dashboard
Screenshot	 Form1 Cozy Comfort Main Menu Dashboard Products Control User Manage Customers Order Management Stock Handling

Test Case ID	T03
Test Case Name	Add product
Result	Successfully product added to the system
Screenshot	 Products Handling Product ID 1 Product Name Fleet Blanket Price 2300 ProductId ProductName Price 1 Fleet Blanket 2300

	 Products Products Handling Product ID: 2 Product Name: Cotton Blanket Price Add Product Added Successfully Product 1 | Fleet Blanket | 2300

Test Case ID	T04
Test Case Name	Update product
Result	Successfully product updated
Screenshot	 Products Handling Product ID: 2 Product Name: Cotton Blanket Price Update Product Updated Successfully 1 Fleet Blanket 2300 2 Cotton Blanket 5000

Test Case ID	T05
Test Case Name	Delete product
Result	Successfully product deleted
Screenshot	

Products

Products Handling

Product ID: 2

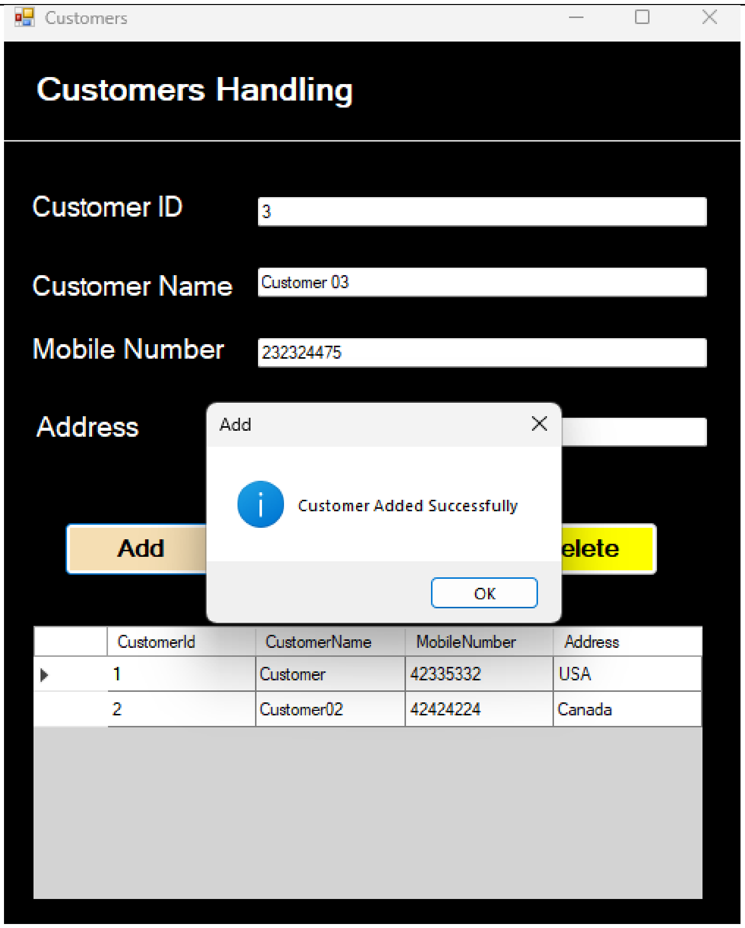
Product Name: Cotton Blanket

Price:

Delete

Product Deleted Successfully

	Produc		
▸	1	Fleet Blanket	2300
	2	Cotton Blanket	2450

Test Case ID	T06
Test Case Name	Add Customer
Result	Successfully add the customer
Screenshot	

Customers Handling

Customer ID: 3

Customer Name: Customer 03

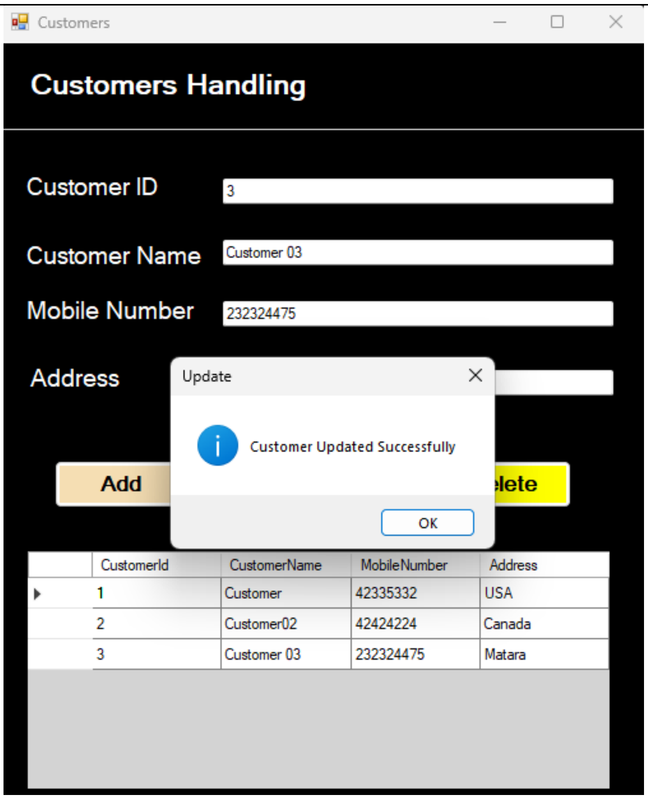
Mobile Number: 232324475

Address:

Add

Customer Added Successfully

	CustomerId	CustomerName	MobileNumber	Address
▸	1	Customer	42335332	USA
	2	Customer02	42424224	Canada

Test Case ID	T07
Test Case Name	Update customer
Result	Successfully update the customer
Screenshot	

Customers Handling

Customer ID: 3

Customer Name: Customer 03

Mobile Number: 232324475

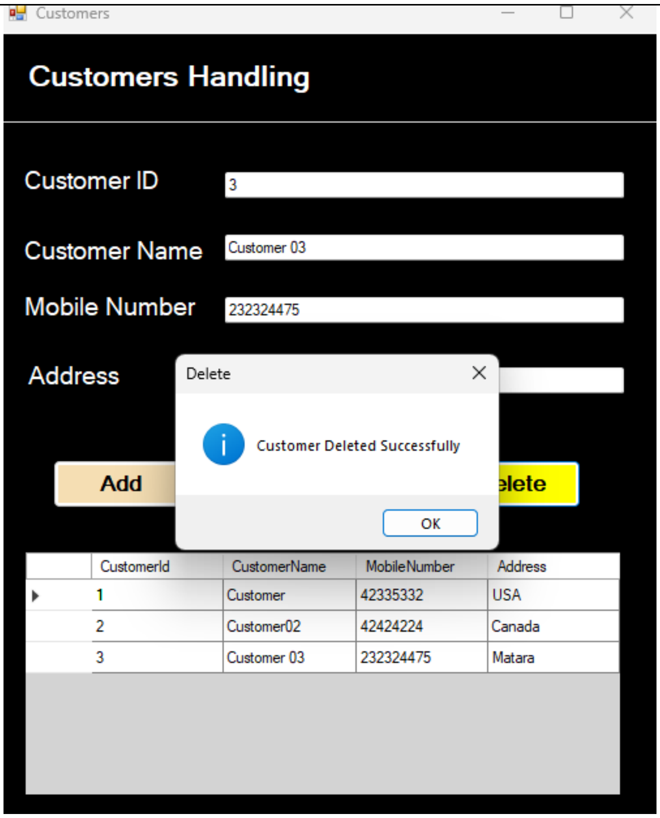
Address

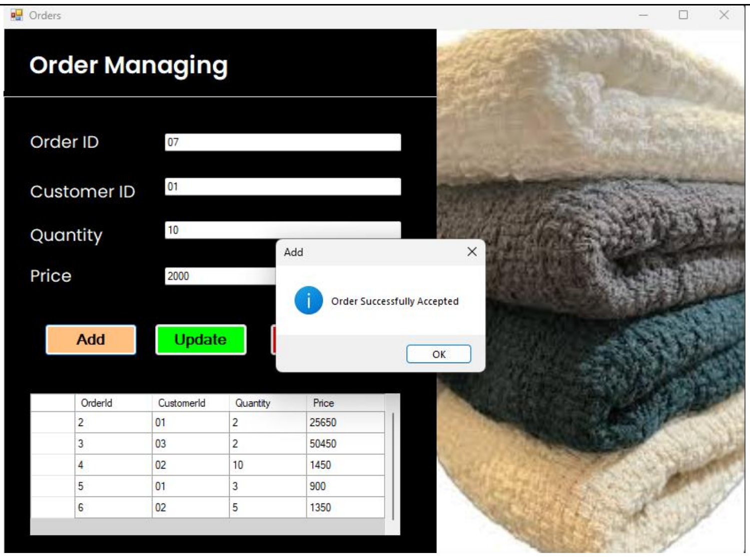
Update

Customer Updated Successfully

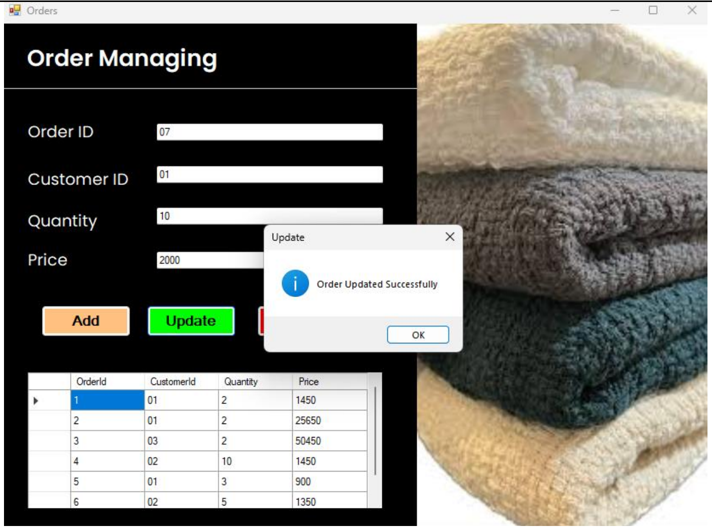
OK

CustomerId	CustomerName	MobileNumber	Address
1	Customer	42335332	USA
2	Customer02	42424224	Canada
3	Customer 03	232324475	Matara

Test Case ID	T08
Test Case Name	Delete customer
Result	Successfully deleted the customer
Screenshot	 Customers Handling Customer ID 3 Customer Name Customer 03 Mobile Number 232324475 Address Delete Customer Deleted Successfully CustomerId CustomerName MobileNumber Address 1 Customer 42335332 USA 2 Customer02 42424224 Canada 3 Customer 03 232324475 Matara

<table>
<tr><td><b>Test Case ID</b></td><td>T09</td></tr>
<tr><td><b>Test Case Name</b></td><td>Add order</td></tr>
<tr><td><b>Result</b></td><td>Successfully order added to the system</td></tr>
<tr><td><b>Screenshot</b></td><td>

Order Managing
Order ID 07
Customer ID 01
Quantity 10
Price 2000
Add
Order Successfully Accepted
<table>
<tr><th>OrderId</th><th>CustomerId</th><th>Quantity</th><th>Price</th></tr>
<tr><td>2</td><td>01</td><td>2</td><td>25650</td></tr>
<tr><td>3</td><td>03</td><td>2</td><td>50450</td></tr>
<tr><td>4</td><td>02</td><td>10</td><td>1450</td></tr>
<tr><td>5</td><td>01</td><td>3</td><td>900</td></tr>
<tr><td>6</td><td>02</td><td>5</td><td>1350</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td><b>Test Case ID</b></td><td>T10</td></tr>
<tr><td><b>Test Case Name</b></td><td>Update order</td></tr>
<tr><td><b>Result</b></td><td>Successfully order updated</td></tr>
<tr><td><b>Screenshot</b></td><td>


Orders

Order Managing

Order ID: 07

Customer ID: 01

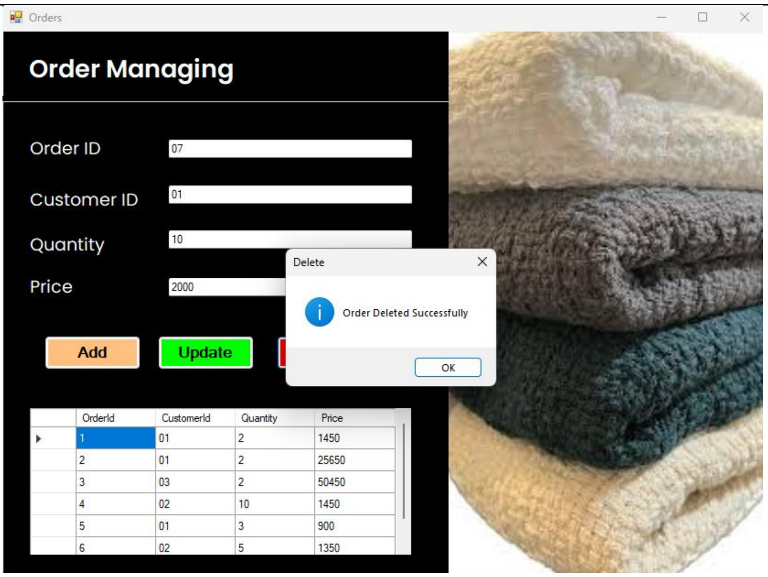
Quantity: 10

Price: 2000

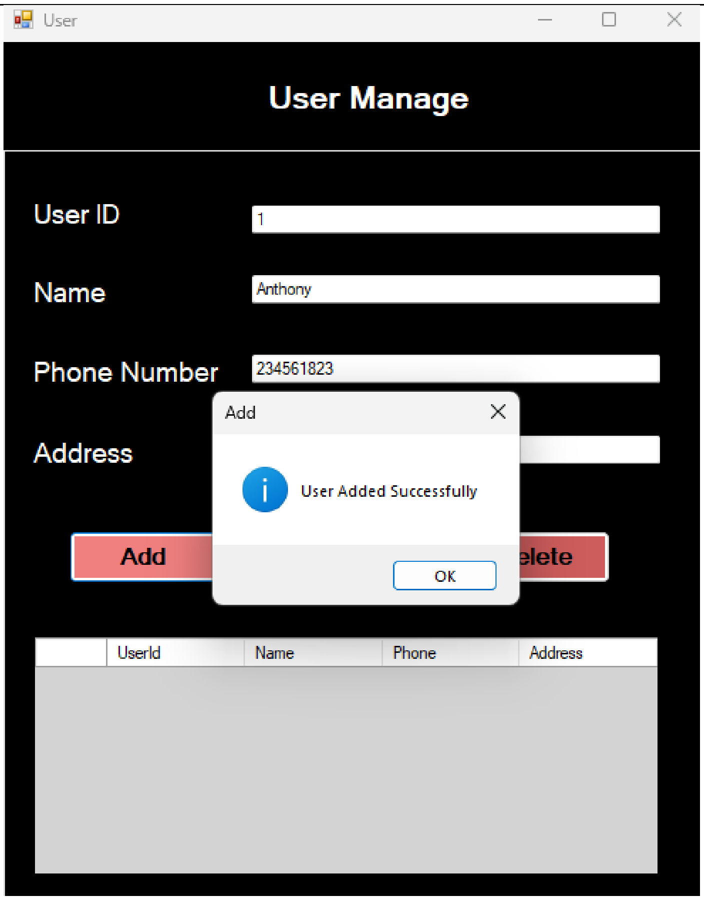
Update

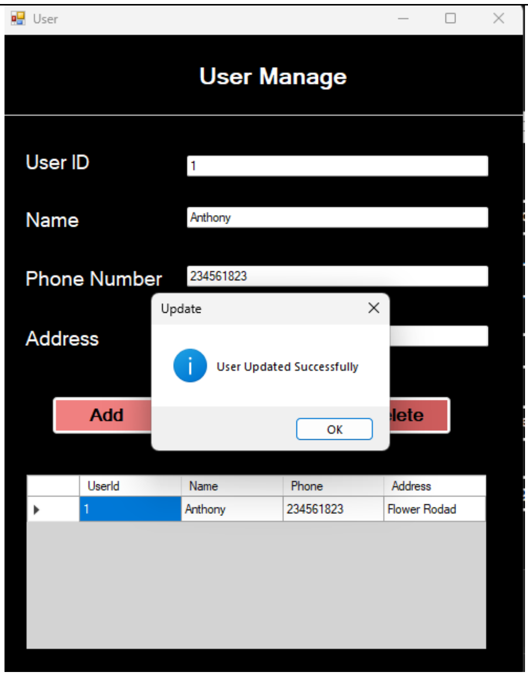
Order Updated Successfully

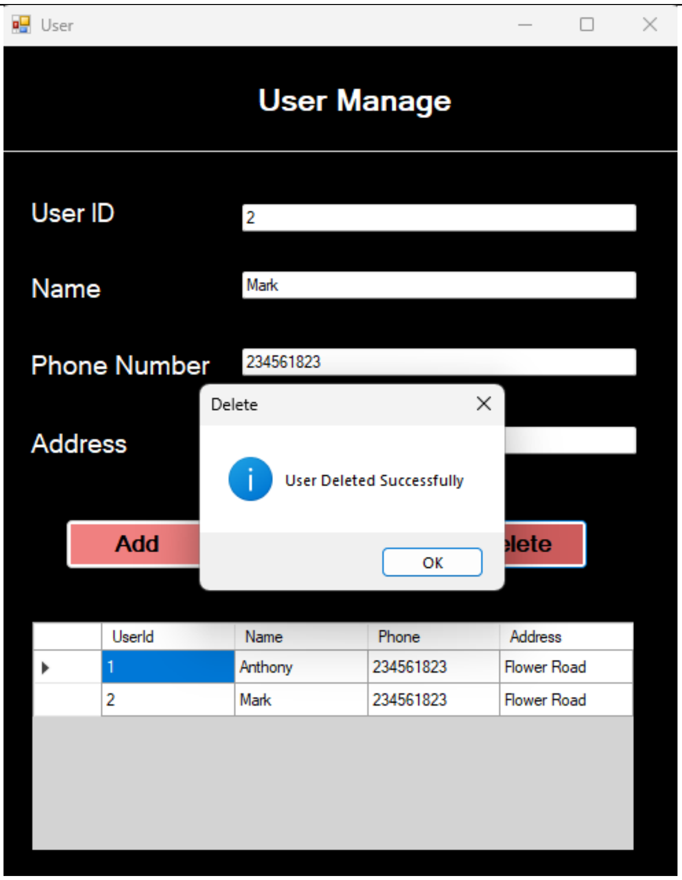
<table>
<tr><th>OrderId</th><th>CustomerId</th><th>Quantity</th><th>Price</th></tr>
<tr><td>1</td><td>01</td><td>2</td><td>1450</td></tr>
<tr><td>2</td><td>01</td><td>2</td><td>25650</td></tr>
<tr><td>3</td><td>03</td><td>2</td><td>50450</td></tr>
<tr><td>4</td><td>02</td><td>10</td><td>1450</td></tr>
<tr><td>5</td><td>01</td><td>3</td><td>900</td></tr>
<tr><td>6</td><td>02</td><td>5</td><td>1350</td></tr>
</table>
</td></tr>
</table>

Test Case ID	T11
Test Case Name	Delete an order
Result	Successfully order deleted form the system
Screenshot	 Order Managing Order ID 07 Customer ID 01 Quantity 10 Price 2000 Delete Order Deleted Successfully

	OrderId	CustomerId	Quantity	Price
▸	1	01	2	1450
	2	01	2	25650
	3	03	2	50450
	4	02	10	1450
	5	01	3	900
	6	02	5	1350

Test Case ID	T12
Test Case Name	Add a user
Result	User successfully added to the system
Screenshot	 User User Manage User ID: 1 Name: Anthony Phone Number: 234561823 Address: Add User Added Successfully OK

Test Case ID	T13
Test Case Name	Update user
Result	Successfully user updated
Screenshot	 User Manage User ID 1 Name Anthony Phone Number 234561823 Address Update User Updated Successfully UserId | Name | Phone | Address 1 | Anthony | 234561823 | Flower Rodad

<table>
<tr><td><b>Test Case ID</b></td><td>T14</td></tr>
<tr><td><b>Test Case Name</b></td><td>Delete User</td></tr>
<tr><td><b>Result</b></td><td>Successfully user deleted</td></tr>
<tr><td><b>Screenshot</b></td><td>


User Manage

User ID: 2
Name: Mark
Phone Number: 234561823
Address:

Delete
User Deleted Successfully

<table>
<tr><th>UserId</th><th>Name</th><th>Phone</th><th>Address</th></tr>
<tr><td>1</td><td>Anthony</td><td>234561823</td><td>Flower Road</td></tr>
<tr><td>2</td><td>Mark</td><td>234561823</td><td>Flower Road</td></tr>
</table>
</td></tr>
</table>

Task 04

Deployment Strategies for Cozy Comfort SOC Solution

Traditional Server Deployment

A traditional model of installing the applications on physical or virtual servers dedicated to this purpose can still be used for small deployments . This approach allows for total control of the hardware configurations and the software environment and is appropriate for companies with stable workloads . This is but not very scalable, because when more capacity is needed, one has to manually provision more servers. Maintenance can also become difficult as each update / patch will require system downtime which can interrupt business processes. For Cozy Comfort this method of deployment would be effective in the beginning stages of business but would soon become impractical as it expanded.

Docker Containerization

Containerization via Docker is a more flexible development and deployment model, in which each service and its dependencies are packaged into a lightweight, isolated container . This guarantees consistent results from place to place and resolves the "it works on my machine" issues that usually happen. They are fast to start and efficient in resource use, which makes them a good fit for development and testing. But Docker is not an orchestration tool by nature and other tools must be used in order to handle multiple containers at production level.

Kubernetes (K8s) Orchestration

Kubernetes (K8s) offers an enterprise-grade solution for businesses that need high availability and easy scalability. It is a system for automating deployment, scaling and managing containerized applications. Kubernetes has auto-scaling capabilities so the system can easily deal with peaks in traffic by dynamic resource allocation. It has a concept of self-healing in that it will automatically restart containers that have failed, thus very little downtime. Kubernetes also supports rolling updates, so that new versions can be safely deployed without taking the service down.

Serverless Computing

Serverless solutions like AWS Lambda or Azure Functions do not require any server management at all. They run code in response to events, automatically scaling with demand, and charging only for what they actually use. Ideal for event- based tasks such as notifying orders or updating inventory. On the other hand, serverless functions have to deal with cold start times and cannot run long processes or run in a stateful manner.

Optimal Deployment Strategy for Cozy Comfort

The optimal solution to achieve scalability, reliability and cost- effectiveness is to implement a hybrid deployment model that combines the use of Kubernetes for the core services and serverless for auxiliary functions. Kubernetes can take care of main stuff like order processing, inventory, etc., notifications and analytics can be handled by serverless functions. This allows critical workflows to operate at peak performance, while also allowing movement flexibility for non-critical tasks.

Final Recommendation:

- Manufacturer, Distributor, and Seller APIs for Kubernetes
- Serverless for notifications, logging, and real-time alerts

This ensures a scalable and more efficient business operation which is exactly what Cozy Comfort hopes to accomplish.

Conclusion

The SOC model is a revolutionary solution for Cozy Comfort and its supply chain inadequacies. Removing archaic manual processes that can lead to human error with automated API-driven processes, the proposed system offers real-time inventory awareness, efficient order processing, and effortless scalability that are vital to business success.

As the result of our investigation, Kubernetes (K8s) was identified as the best deployment approach since it allows for scaling on demand, fault tolerance, and zero-downtime updates. Complemented by serverless components for event-driven tasks, this hybrid model guarantees that costs are efficient without compromising performance.

The SOC model addresses each of the current issues as well as provides a future-proofed digital infrastructure for "Cozy Comfort" that will allow for: Faster, less error prone fulfillment of orders Simplifying the onboarding of new distributors/sellers Minimize costs of operation via automation. Flexibility for changing market demands

Through this solution Cozy Comfort will be able to build customers' satisfaction, partner's collaboration, and a competitive advantage in the home textiles industry. The case shows how businesses can use supply chain challenges through a thoughtful use of technology.

Recommendation, Final Version: Move forward with a Kubernetes-based SOC build-out, phased implementation, monitoring of the SOC and then maximizing ROI.

References

1. *Erl, T. (2008) SOA: Principles of Service Design. Upper Saddle River: Prentice Hall.*

2. *Newman, S. (2015) Building Microservices. Sebastopol: O'Reilly Media.*

3. *Burns, B. et al. (2016) Kubernetes: Up and Running. Sebastopol: O'Reilly Media.*

4. *Richardson, C. (2018) Microservices Patterns. Shelter Island: Manning Publications.*

5. *Microsoft (2022) ASP.NET Core documentation [Online]. Available at: https://docs.microsoft.com/en-us/aspnet/core/ (Accessed: 15 June 2023).*

6. *Docker Inc. (2023) Docker documentation [Online]. Available at: https://docs.docker.com/ (Accessed: 15 June 2023).*

7. *Kubernetes.io (2023) Kubernetes documentation [Online]. Available at: https://kubernetes.io/docs/ (Accessed: 15 June 2023).*

8. *AWS (2023) AWS Lambda documentation [Online]. Available at: https://aws.amazon.com/lambda/ (Accessed: 15 June 2023).*

9. *Azure (2023) Azure Functions documentation [Online]. Available at: https://docs.microsoft.com/en-us/azure/azure-functions/ (Accessed: 15 June 2023).*

10. *Fowler, M. (2014) Microservices [Online]. Available at: https://martinfowler.com/articles/microservices.html (Accessed: 15 June 2023).*

11. *Nadareishvili, I. et al. (2016) Microservice Architecture. Sebastopol: O'Reilly Media.*

12. *Hohpe, G. and Woolf, B. (2003) Enterprise Integration Patterns. Boston: Addison-Wesley.*

13. *Richardson, L. and Ruby, S. (2007) RESTful Web Services. Sebastopol: O'Reilly Media.*

14. *Fielding, R.T. (2000) Architectural Styles and the Design of Network-based Software Architectures. PhD thesis. University of California, Irvine.*

15. *Dragoni, N. et al. (2017) 'Microservices: Yesterday, today, and tomorrow', Present and Ulterior Software Engineering, pp. 195-216.*