PUDUCHERRY TECHNOLOGICAL UNIVERSITY

PUDUCHERRY

STATIONERY SHOP

STOCK MANAGEMENT SYSTEM

ABSTRACT

The **STATIONERY SHOP STOCK MANAGEMENT SYSTEM**, developed **using Microsoft Visual Studio C#** and powered by a **Microsoft SQL Server Database**, is designed to efficiently manage and track a variety of stationery items such as paper, pencils, pens, and envelopes. This system not only simplifies inventory management by enabling real-time monitoring of stock levels and locations but also enhances sales management with features for recording transactions, generating comprehensive sales reports, and analyzing sales trends. Administrators can seamlessly manage user roles, including adding salesman users, and maintain customer information to facilitate personalized service and targeted marketing efforts. With intuitive dashboards and detailed reports, the system provides valuable insights into inventory status, sales performance, and overall business profitability. Designed for ease of use and scalability, the Stationery Shop Stock Management System optimizes operational efficiency, improves decision-making processes, and supports the growth objectives of stationery shop owners in a competitive market landscape.

1.INTRODUCTION

The **STATIONERY SHOP STOCK MANAGEMENT SYSTEM**, developed using **MICROSOFT VISUAL STUDIO C#** and **MICROSOFT SQL SERVER DATABASE**, is a comprehensive and user-friendly solution for efficiently managing and tracking stationery items such as paper, pencils, pens, and envelopes. This system allows shop administrators to add new items, update stock levels, and categorize inventory for streamlined management, ensuring accurate stock counts and preventing overstocking or stockouts. It also maintains detailed sales records, capturing item details, quantities, prices, and salesperson information, which helps in tracking performance, identifying trends, and managing revenue. The system's role-based access control provides shop admins with full access to all functionalities, including user management, dashboard analysis, and report viewing, while salesmen can add items, view stocks, generate sales reports, and manage customer details, ensuring data security and task efficiency. The dashboard and reporting features offer visual representations of key metrics and performance indicators, aiding in quick identification of areas needing attention and facilitating informed decision-making. Detailed, customizable reports on sales trends, inventory turnover, and customer behavior provide valuable insights for strategic planning. Additionally, the system's customer management features enable the addition and updating of customer details, fostering strong relationships and personalized service, and tracking purchases for targeted marketing and loyalty programs. Overall, the Stationery Shop Stock Management System enhances operational efficiency, improves sales management, and supports data-driven decision-making, making it an essential tool for any stationery shop aiming for growth and profitability.

2.MODULE

This project consists of seven modules namely:

1. **MANAGE ITEMS MODULE**
2. **MANAGE USERS MODULE**
3. **MANAGE BILLING MODULE**
4. **MANAGE CATEGORIES MODULE**
5. **SHOP DASHBOARD MODULE**
6. **MANAGE CUSTOMER MODULE**
7. **MANAGE STOCK MODULE**

2.1 MANAGE ITEMS MODULE: -

The Manage Items module is designed to store and manage stationery item details. It tracks information such as item names, categories, quantities, prices, profits, details, and addition dates. This module ensures organized inventory management for stationery products.

2.2 MANAGE USERS MODULE: -

The Manage Users module is used to store salesman's biodata. This database will handle their login information.

2.3 MANAGE BILLING MODULE: -

The Manage Billing module generates invoices for items sold, calculating the total amount payable. It ensures accurate pricing and quantity details for each item. Once generated, bills are securely stored in the system for future reference, reporting, and auditing purposes.

2.4 MANAGE CATEGORIES MODULE: -

The Manage Categories module will create different types of categories used to separate item types.

2.5 SHOP DASHBOARD MODULE: -

The Shop Dashboard module is used to show item sales statistics based on day-to-day sales. This module manages how many items are sold, how much income is earned by each salesperson, and it shows the bonus amount rate based on salesperson performance.

2.6 MANAGE CUSTOMER MODULE:

The Manage Customer module will handle the details of purchasing customers and the details of products they purchase.

2.7 MANAGE STOCK MODULE:

The Manage Stock module will manage which items are available, which items are unavailable, and which items will soon be unavailable.

3. ER DIAGRAM

4. DATA DICTIONARY (TABLES WITH CONSTRAINTS)

4.1 Table name: ItemsTbl

Primary key: ItId

NAME	DATA TYPE	NOT NULL	DEFAULT
ItId	int	False	
ItName	varchar(50)	False	
ItCat	int	False	((0))
ItQty	int	False	
ItBPrice	int	False	
ItSPrice	int	False	
ItProfit	int	False	
ItDetails	varchar(50)	False	
ItAddDate	date	False	

4.2 Table name: UsersTbl

Primary key:UId

NAME	DATA TYPE	NOT NULL	DEFAULT
UId	int	False	
UName	varchar(50)	False	
UGen	varchar(50)	False	
UDOB	date	False	
UEmail	varchar(50)	False	
UPassword	varchar(50)	False	
UPhone	bigint	False	

4.3 Table name: BillDetailsTbl

Primary key: BillDetailId

NAME	DATA TYPE	NOT NULL	DEFAULT
BillDetailId	int	False	
SalesId	int	False	
ItemNo	nvarchar(100)	False	
ProductName	nvarchar(100)	False	
Quantity	int	False	
Price	int	False	
Total	int	False	

4.4 Table name: CatTbl

Primary key: CatId

	NAME	DATA TYPE	NOT NULL	DEFAULT
	CatId	int	False	
	CatName	varchar(50)	False	

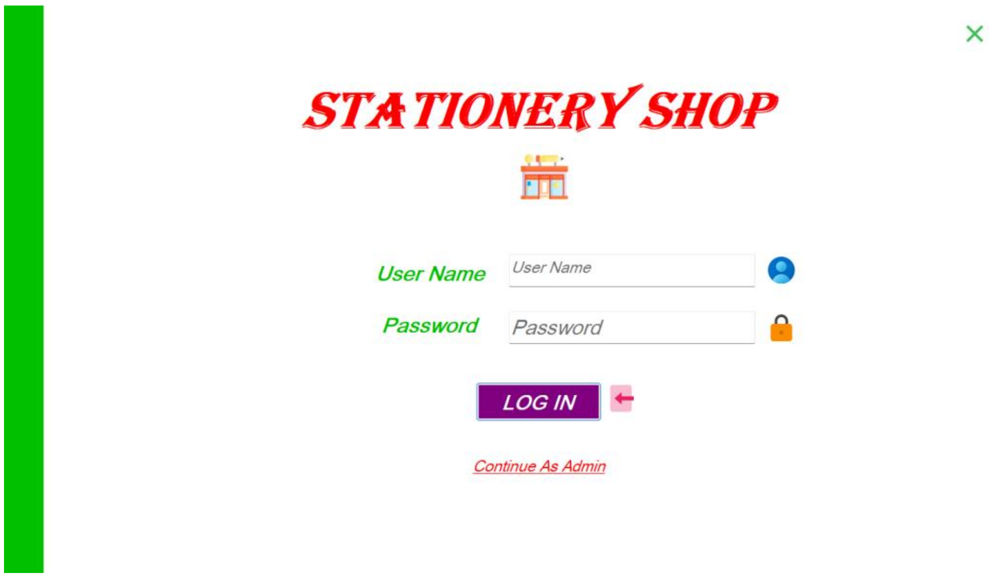
4.5 Table name: SalesTbl

Primary key:SNum

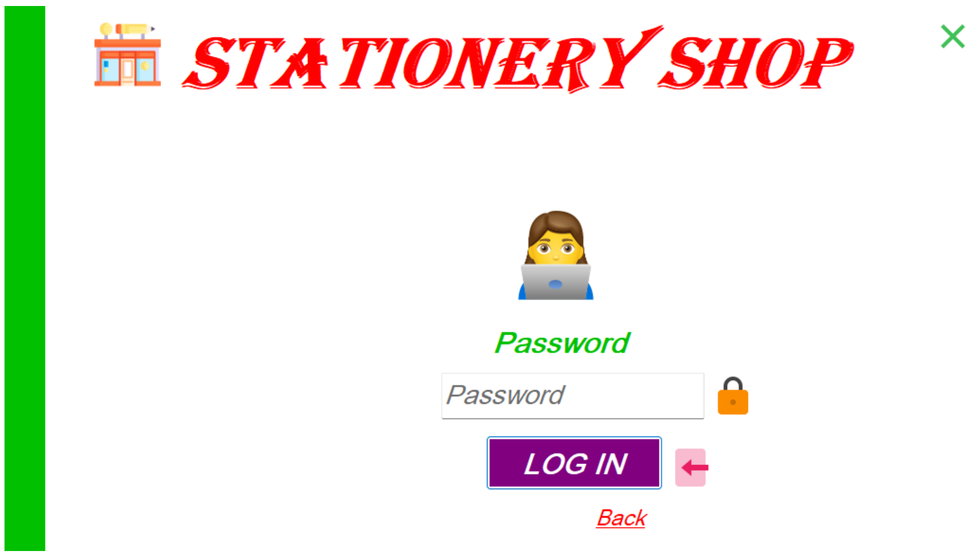
NAME	DATA TYPE	NOT NULL	DEFAULT
SNum	int	False	
SDate	date	False	
SCustomer	varchar(50)	False	
SPhone	bigint	False	
SUser	Int	False	
SAmount	Int	False	

5.MODULE DESIGN

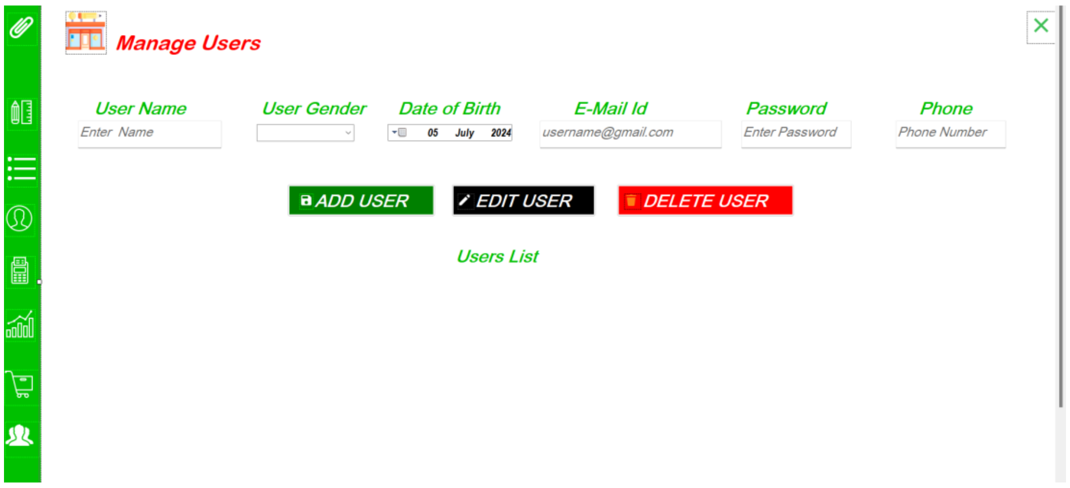
❖ **LOGIN DESIGN:**



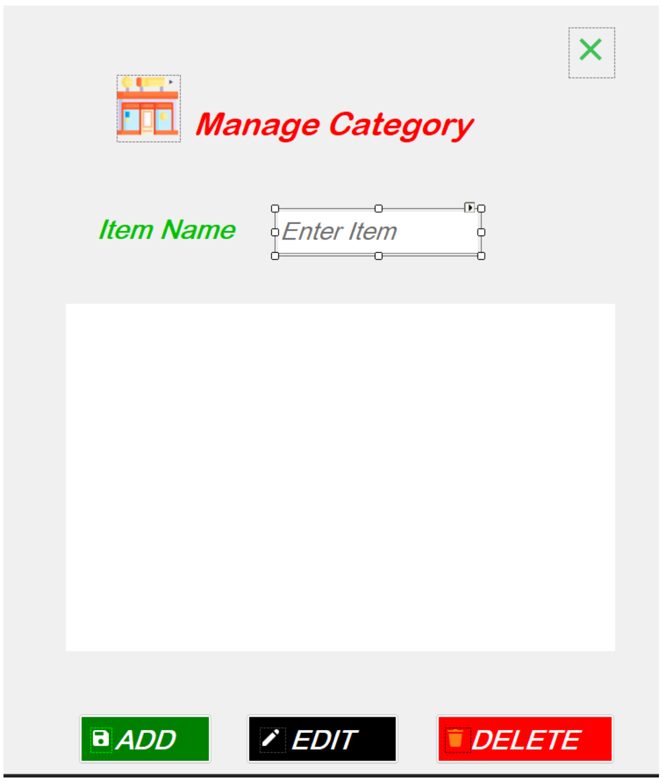
❖ **ADMIN DESIGN:**



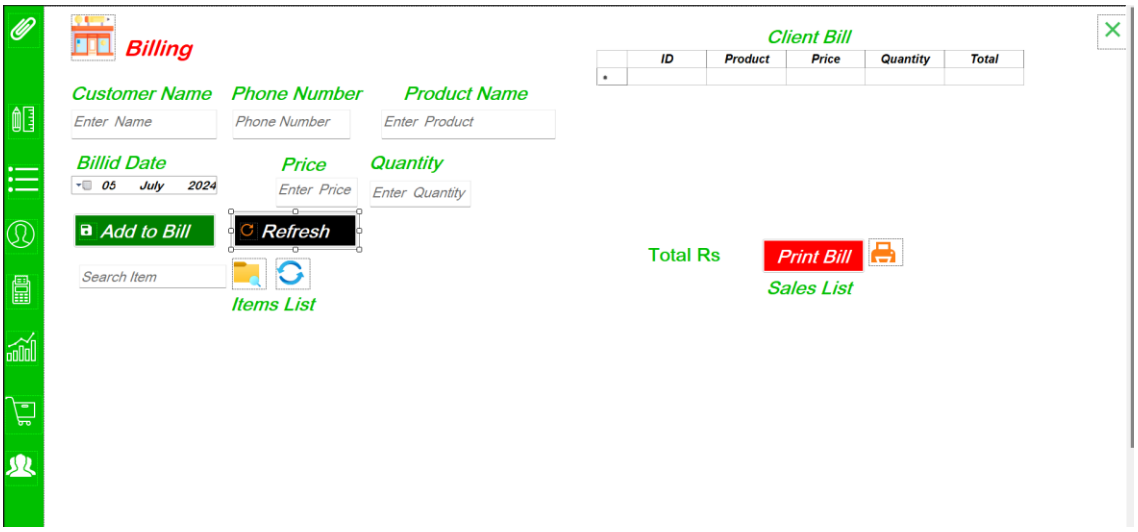
❖ MANAGE USER DESIGN:



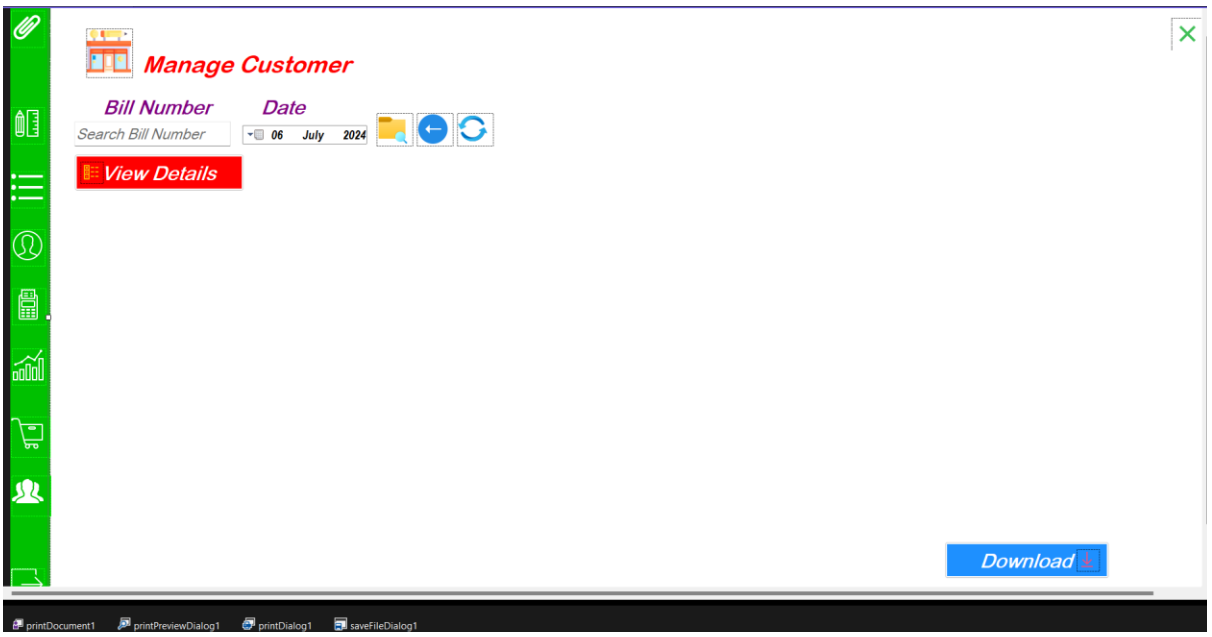
❖ MANAGE CATEGORY DESIGN:



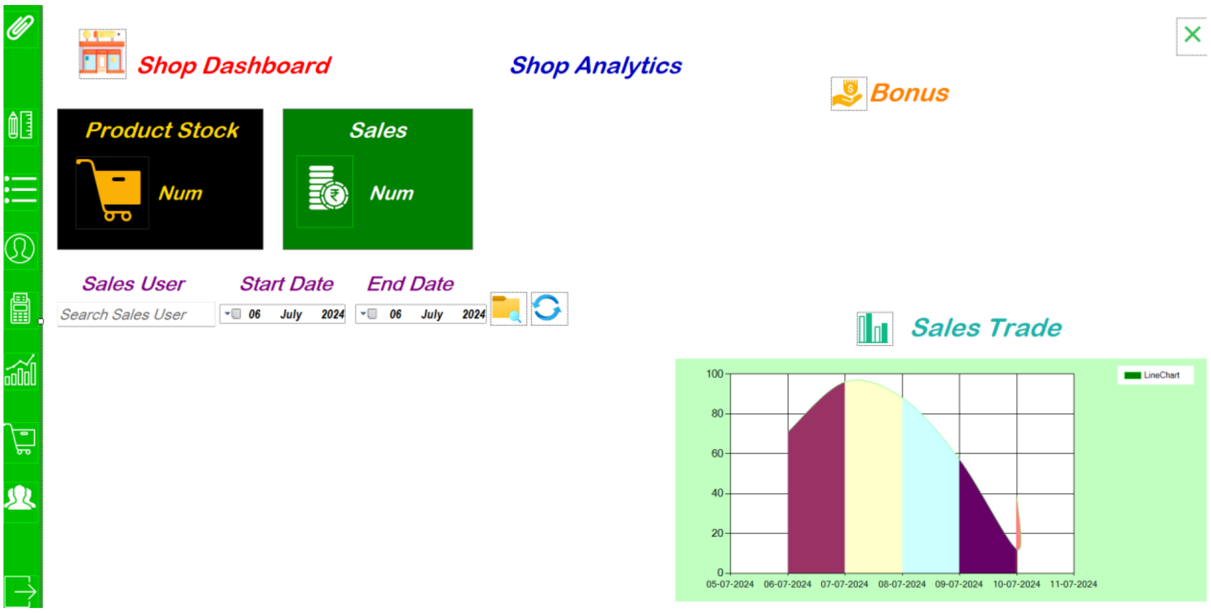
❖ MANAGE BILLING DESIGN:



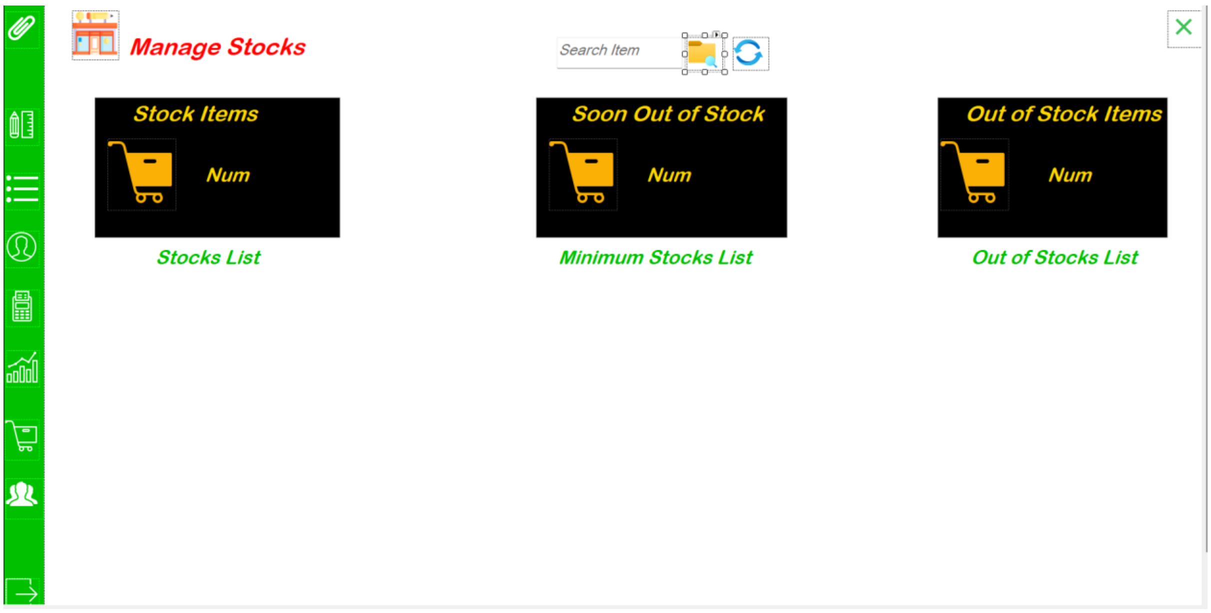
❖ MANAGE CUSTOMER DESIGN:



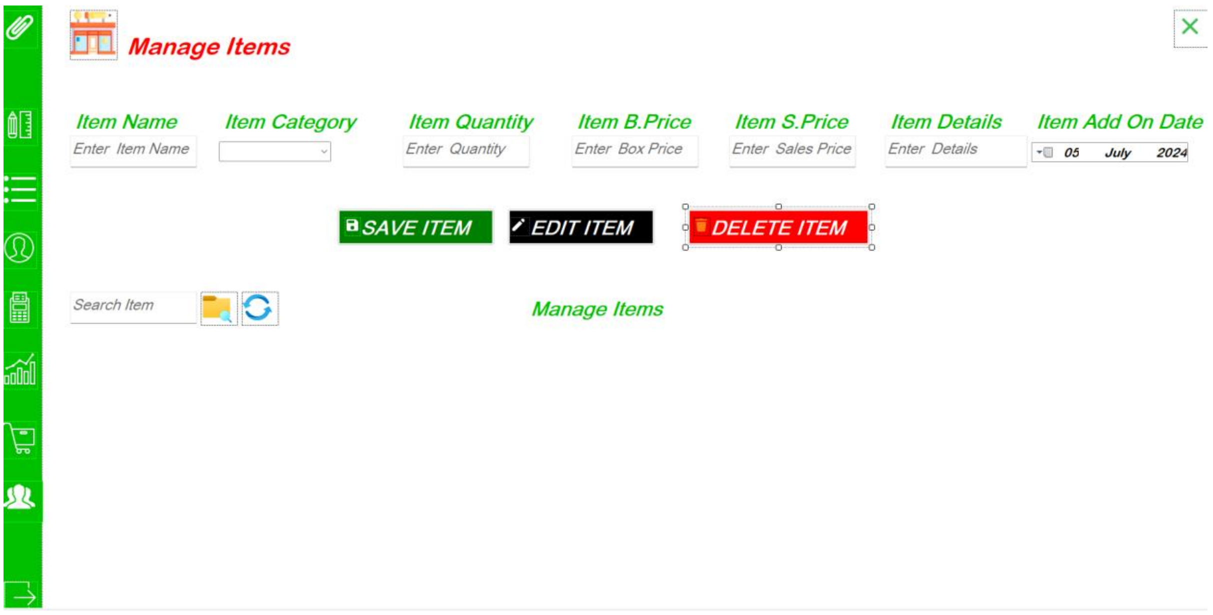
❖ MANAGE USER DESIGN:



❖ MANAGE STOCK DESIGN:



❖ MANAGE ITEM DESIGN:



6. CODING

ADMIN MODULE:

```
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 STATIONERY_SHOP
{    public partial class AdminLogin : Form
{        public AdminLogin()
{            InitializeComponent();}
private void LoginBtn_Click(object sender, EventArgs e)
{ if (PasswordTb.Text == "")
{MessageBox.Show("Enter the Password!!!");
PasswordTb.Text = "";}
else if (PasswordTb.Text == "admin")
{
Users Obj = new Users();
Obj.Show();
this.Hide();
}else
{MessageBox.Show("Wrong Password!!!");
PasswordTb.Text = "";}}
private void label4_Click(object sender, EventArgs e)
{Login Obj = new Login();
Obj.Show();
this.Hide();}}}
```

LOGIN MODULE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace STATIONERY_SHOP
{public partial class Login : Form
{public Login()
{InitializeComponent();}
```

```
public static string UName = "";
private void LoginBtn_Click(object sender, EventArgs e)
{if (UNameTb.Text == "" || PasswordTb.Text == "")
{MessageBox.Show("Please Enter UserName and Password!!!");
}else
{Con.Open();
SqlDataAdapter sda = new SqlDataAdapter("select count(*) from UserTbl where
UName='" + UNameTb.Text + "' and UPassword='" + PasswordTb.Text + "'", Con);
DataTable dt = new DataTable();
sda.Fill(dt);
if (dt.Rows[0][0].ToString() == "1")
{UName = UNameTb.Text;
Billing Obj = new Billing();
Obj.Show();
this.Hide();
Con.Close();
}else
{MessageBox.Show("Wrong UserName Or Password!!!");
}Con.Close();}}
```

USER MODULE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace STATIONERY_SHOP
{public partial class Users : Form
{public Users()
{InitializeComponent();
Methods obj = new Methods();
obj.DisplayData("UserTbl", UserGDV);
GenCb.Items.Add("Select Gender");
GenCb.SelectedIndex = 0;
EmailTb.TextChanged += EmailTb_TextChanged;
}private void SaveBtn_Click(object sender, EventArgs e)
{string name = UNameTb.Text;
string email = EmailTb.Text;
string password = PasswordTb.Text;
string phoneNumber = PhoneTb.Text;
string gender = GenCb.SelectedItem.ToString();
if (UNameTb.Text == "" || GenCb.Text == "Select Gender " || EmailTb.Text == "" ||
PasswordTb.Text == "" || PhoneTb.Text == "")
```

```
{MessageBox.Show("Missing Information!!!");
}else
{ValidatePassword(password);
if (IsNameValid(name) && IsEmailValid(email) && IsPasswordValid(password) &&
IsPhoneNumberValid(phoneNumber))
{try
{Con.Open();
SqlCommandcmd=newSqlCommand("insertintoUserTbl(UName,UGen,UDOB,UEm
ail,UPassword,Uphone) values (@UN,@UG,@UD,@UEM,@UPa,@UP)", Con);
cmd.Parameters.AddWithValue("@UN", name);
cmd.Parameters.AddWithValue("@UG", gender);
cmd.Parameters.AddWithValue("@UD", UDOB.Value.Date);
cmd.Parameters.AddWithValue("@UEM", email);
cmd.Parameters.AddWithValue("@UPa", password);
cmd.Parameters.AddWithValue("@UP", phoneNumber);
cmd.ExecuteNonQuery();
MessageBox.Show("User Added!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("UserTbl", UserGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message); } }
int Key = 0;
private void UserGDV_CellContentClick(object sender, DataGridViewCellEventArgs
e)
{UNameTb.Text = UserGDV.SelectedRows[0].Cells[1].Value.ToString();
GenCb.Text = UserGDV.SelectedRows[0].Cells[2].Value.ToString();
UDOB.Text = UserGDV.SelectedRows[0].Cells[3].Value.ToString();
EmailTb.Text = UserGDV.SelectedRows[0].Cells[4].Value.ToString();
PasswordTb.Text = UserGDV.SelectedRows[0].Cells[5].Value.ToString();
PhoneTb.Text = UserGDV.SelectedRows[0].Cells[6].Value.ToString();
if (UNameTb.Text == "")
{Key = 0;
}else
{Key = Convert.ToInt32(UserGDV.SelectedRows[0].Cells[0].Value.ToString());}}
private void EditBtn_Click(object sender, EventArgs e)
{string name = UNameTb.Text;
string password = PasswordTb.Text;
string phoneNumber = PhoneTb.Text;
string email = EmailTb.Text;
if (UNameTb.Text == "" || GenCb.Text == "" || EmailTb.Text == "" || PasswordTb.Text
== "" || PhoneTb.Text == "")
{MessageBox.Show("Missing Information!!!");
} else
{ValidatePassword(password);
if(IsNameValid(name)&&IsPasswordValid(password)&&IsPhoneNumberValid(phone
Number) && IsEmailValid(email))
{try
{Con.Open();
SqlCommandcmd=newSqlCommand("updateUserTblsetUName=@UN,UGen=@UG,
UDOB=@UD,UEmail=@UEM,UPassword=@UPa,Uphone=@UP                where
UId=@UK", Con);
cmd.Parameters.AddWithValue("@UN", name);
cmd.Parameters.AddWithValue("@UG",value:GenCb.SelectedItem.ToString());
cmd.Parameters.AddWithValue("@UD", UDOB.Value.Date);
cmd.Parameters.AddWithValue("@UEM", email);
cmd.Parameters.AddWithValue("@UPa", password);
cmd.Parameters.AddWithValue("@UP", phoneNumber);
cmd.Parameters.AddWithValue("@UK", Key);
cmd.ExecuteNonQuery();
MessageBox.Show("User Updated!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("UserTbl", UserGDV);}
catch (Exception Ex)
{MessageBox.Show(Ex.Message);}}}
private void DeleteBtn_Click(object sender, EventArgs e)
{if (Key == 0)
{MessageBox.Show("Missing Information!!!");
}else
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("delete from UserTbl where UId=@UK",
Con);
cmd.Parameters.AddWithValue("@UK", Key);
cmd.ExecuteNonQuery();
MessageBox.Show("User Deleted!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("UserTbl", UserGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message);} } }
private bool IsNameValid(string name)
{if (string.IsNullOrWhiteSpace(name))
{MessageBox.Show("Name cannot be empty.");
return false;
}string pattern = @"^[a-zA-Z ]+$";
if (!Regex.IsMatch(name, pattern))
{MessageBox.Show("Name should contain only alphabetic characters and spaces.");
return false;
}if (name.Length < 4 || name.Length > 25)
{MessageBox.Show("Name must be between 4 and 10 characters long.");
return false;}
return true;}
private void EmailTb_TextChanged(object sender, EventArgs e)
{string originalText = EmailTb.Text;
string modifiedText = ModifyText(originalText);
EmailTb.Text = modifiedText;}
private string ModifyText(string input)
```

```
{return input.ToLower();}
private bool IsEmailValid(string email)
{if (string.IsNullOrWhiteSpace(email))
{MessageBox.Show("Email address cannot be empty.");
return false;}
string pattern = @"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$";
if (!Regex.IsMatch(email, pattern))
{MessageBox.Show("Invalid email address!! format 'Username@gmail.com'.");
return false;}
return true;}
private void ValidatePassword(string password)
{if (string.IsNullOrWhiteSpace(password))
{MessageBox.Show("Password cannot be empty.");
}else if (password.Length < 8)
{MessageBox.Show("Password must be at least 8 characters long.");
}else if (!Regex.IsMatch(password, @"[A-Z]"))
{MessageBox.Show("Password must contain at least one uppercase letter.");
}else if (!Regex.IsMatch(password, @"[a-z]"))
{MessageBox.Show("Password must contain at least one lowercase letter.");
}else if (!Regex.IsMatch(password, @"[0-9]"))
{MessageBox.Show("Password must contain at least one number.");
} else if (!Regex.IsMatch(password, @"[\W_]"))
{MessageBox.Show("Password must contain at least one special character.");}}
private bool IsPasswordValid(string password)
{if(string.IsNullOrWhiteSpace(password)||password.Length<8||!Regex.IsMatch(pass
word, @"[A-Z]") ||!Regex.IsMatch(password, @"[a-z]") ||!Regex.IsMatch(password,
@"[0-9]") ||!Regex.IsMatch(password, @"[\W_]"))
{return false;
}return true;}
private void PhoneTb_KeyPress(object sender, KeyPressEventArgs e)
{if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
{e.Handled = true;
MessageBox.Show("Invalid input! Please enter digits only."); }}
private bool IsPhoneNumberValid(string phoneNumber)
{if (string.IsNullOrWhiteSpace(phoneNumber))
{MessageBox.Show("Phone number cannot be empty.");
return false;}
if (phoneNumber.Length != 10)
{MessageBox.Show("Phone number must be exactly 10 digits long.");
return false;}
if (!Regex.IsMatch(phoneNumber, @"^\d{10}$"))
{MessageBox.Show("Phone number must contain only digits.");
return false;}
return true; }}}
```

ITEMS MODULE:

```
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.SqlClient;
using System.Text.RegularExpressions;
namespace STATIONERY_SHOP
{public partial class Items : Form
{public Items()
{InitializeComponent();
Methods obj = new Methods();
obj.DisplayData("ItemTbl", ProductGDV);
GetCategory();
stringU=Login.UName;
QuantityTb.Validating+=newSystem.ComponentModel.CancelEventHandler(Quantit
yTb_Validating);
BPriceTb.Validating+=newSystem.ComponentModel.CancelEventHandler(BPriceTb
_Validating);
SPriceTb.Validating+=newSystem.ComponentModel.CancelEventHandler(SPriceTb_
Validating);
ProdDate.Validating += new CancelEventHandler(ProdDate_Validating);}
private void GetCategory()
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("select CatId, CatName from CategoryTbl",
Con);
SqlDataReader Rdr = cmd.ExecuteReader();
DataTable dt = new DataTable();
dt.Load(Rdr);
DataRow newRow = dt.NewRow();
newRow["CatId"] = 0;
newRow["CatName"] = "Select Category";
dt.Rows.InsertAt(newRow, 0);
CatCb.ValueMember = "CatId";
CatCb.DisplayMember = "CatName";
CatCb.DataSource = dt;}}
catch (Exception ex)
{MessageBox.Show("Error: " + ex.Message);}}
private void SaveBtn_Click(object sender, EventArgs e)
{string prodname = ProdNameTb.Text;if (ProdNameTb.Text == "" || CatCb.Text ==
"Select Category" || ProdDetailsTb.Text == "" || SPriceTb.Text == "" || BPriceTb.Text
== "" || ProdDetailsTb.Text == "")
{MessageBox.Show("Missing Information!!!");
}else
{if (IsProdNameValid(prodname))
{try
{int Profit = Convert.ToInt32(SPriceTb.Text) - Convert.ToInt32(BPriceTb.Text);
Con.Open();
SqlCommandcmd=newSqlCommand("insertintoItemTbl(ItName,ItCat,ItQty,ItBPrice,
ItSPrice,ItProfit,ItDetails,ItAddDate)                                   values
(@IN,@IC,@IQ,@IBP,@ISP,@IP,@ID,@IADate)", Con);
cmd.Parameters.AddWithValue("@IN", prodname);
cmd.Parameters.AddWithValue("@IC", CatCb.SelectedValue.ToString());
cmd.Parameters.AddWithValue("@IQ", QuantityTb.Text);
cmd.Parameters.AddWithValue("@IBP", BPriceTb.Text);
cmd.Parameters.AddWithValue("@ISP", SPriceTb.Text);
cmd.Parameters.AddWithValue("@IP", Profit);
cmd.Parameters.AddWithValue("@ID", ProdDetailsTb.Text);
cmd.Parameters.AddWithValue("@IADate", ProdDate.Value.Date);
cmd.ExecuteNonQuery();
MessageBox.Show("Item Added!!!");
Con.Close();
ProdNameTb.Text = "";
CatCb.Text = "Select Category";
QuantityTb.Text = "";
BPriceTb.Text = "";
SPriceTb.Text = "";
ProdDetailsTb.Text = " ";
ProdDate.Value = DateTime.Today;
Methods obj = new Methods();
obj.DisplayData("ItemTbl", ProductGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message);  }}}}
int Key = 0;
private          void          ProductGDV_CellContentClick(object          sender,
DataGridViewCellEventArgs e)
{ProdNameTb.Text = ProductGDV.SelectedRows[0].Cells[1].Value.ToString();
CatCb.Text = ProductGDV.SelectedRows[0].Cells[2].Value.ToString();
QuantityTb.Text = ProductGDV.SelectedRows[0].Cells[3].Value.ToString();
BPriceTb.Text = ProductGDV.SelectedRows[0].Cells[4].Value.ToString();
SPriceTb.Text = ProductGDV.SelectedRows[0].Cells[5].Value.ToString();
ProdDetailsTb.Text = ProductGDV.SelectedRows[0].Cells[7].Value.ToString();
ProdDate.Text = ProductGDV.SelectedRows[0].Cells[8].Value.ToString();
if (ProdNameTb.Text == "")
{Key = 0;
}else
{Key = Convert.ToInt32(ProductGDV.SelectedRows[0].Cells[0].Value.ToString());}}
private void DeleteBtn_Click(object sender, EventArgs e)
{if (Key == 0)
{MessageBox.Show("Missing Information!!!");
}else
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("delete from ItemTbl where ItId=@PK", Con);
cmd.Parameters.AddWithValue("@PK", Key);
cmd.ExecuteNonQuery();
MessageBox.Show("Product Deleted!!!");
```

```
Con.Close();
ProdNameTb.Text = "";
CatCb.Text = "Select Category";
QuantityTb.Text = "";
BPriceTb.Text = "";
SPriceTb.Text = "";
ProdDetailsTb.Text = " ";
ProdDate.Value = DateTime.Today;
Methods obj = new Methods();
obj.DisplayData("ItemTbl", ProductGDV);}
catch (Exception Ex)
{MessageBox.Show(Ex.Message);}}}
private void EditBtn_Click_1(object sender, EventArgs e)
{string prodname = ProdNameTb.Text;
if (prodname == "" || CatCb.SelectedValue == null || CatCb.SelectedValue.ToString()
== "0" || ProdDetailsTb.Text == "" || SPriceTb.Text == "" || BPriceTb.Text == "" ||
QuantityTb.Text == "")
{MessageBox.Show("Missing or Invalid Information!!!");
return;}
if (!IsProdNameValid(prodname))
{return;
}if (Key == 0)
{MessageBox.Show("Please select a product to edit.");
return;}
try{
int Profit = Convert.ToInt32(SPriceTb.Text) - Convert.ToInt32(BPriceTb.Text);
Con.Open();
SqlCommandcmd=newSqlCommand("updateItemTblsetItName=@IN,ItCat=@IC,ItQ
ty=@IQ,ItBPrice=@IBP,ItSPrice=@ISP,ItProfit=@IP,ItDetails=@ID,ItAddDate=@I
ADate where ItId=@PKey", Con);
cmd.Parameters.AddWithValue("@IN", prodname);
cmd.Parameters.AddWithValue("@IC", CatCb.SelectedValue.ToString());
cmd.Parameters.AddWithValue("@IQ", QuantityTb.Text);
cmd.Parameters.AddWithValue("@IBP", BPriceTb.Text);
cmd.Parameters.AddWithValue("@ISP", SPriceTb.Text);
cmd.Parameters.AddWithValue("@IP", Profit);
cmd.Parameters.AddWithValue("@ID", ProdDetailsTb.Text);
cmd.Parameters.AddWithValue("@IADate", ProdDate.Value.Date);
cmd.Parameters.AddWithValue("@PKey", Key);
int rowsAffected = cmd.ExecuteNonQuery();
if (rowsAffected > 0)
{MessageBox.Show("Item Updated!!!");
ProdNameTb.Text = "";
CatCb.Text = "Select Category";
QuantityTb.Text = "";
BPriceTb.Text = "";
SPriceTb.Text = "";
ProdDetailsTb.Text = " ";
ProdDate.Value = DateTime.Today;
Methods obj = new Methods();
```

```
obj.DisplayData("ItemTbl", ProductGDV);
}else
{MessageBox.Show("Update failed. No rows affected.");
}Con.Close();}
catch (Exception Ex)
{MessageBox.Show("An error occurred: " + Ex.Message);}}
private bool isAdminLoggedIn = false;
MessageBox.Show("Only administrators are allowed to access this page.", "Access
Denied", MessageBoxButtons.OK, MessageBoxIcon.Information);}}
private void SearchItem()
{Con.Open();
String Query = "select * from ItemTbl where ItName='" + SearchTb.Text + "'";
SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
SqlCommandBuilder Builder = new SqlCommandBuilder(sda);
var ds = new DataSet();
sda.Fill(ds);
ProductGDV.DataSource = ds.Tables[0];
Con.Close();}
private void ShowItem()
{Con.Open();
String Query = "select * from ItemTbl";
SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
SqlCommandBuilder Builder = new SqlCommandBuilder(sda);
var ds = new DataSet();
sda.Fill(ds);
ProductGDV.DataSource = ds.Tables[0];
Con.Close();}
private void RefreshImg_Click(object sender, EventArgs e)
{ShowItem();
SearchTb.Text = "";}
private bool IsProdNameValid(string name)
{if (string.IsNullOrWhiteSpace(name))
{MessageBox.Show("Product Name cannot be empty.");
return false;}
string pattern = @"^[a-zA-Z1-9 ]+$";
if (!Regex.IsMatch(name, pattern))
{MessageBox.Show("Product Name should contain only alphabetic characters and
spaces.");
return false;}
if (name.Length < 4 || name.Length > 30)
{MessageBox.Show("Product Name must be between 4 and 30 characters long.");
return false;}return true;}
privatevoidQuantityTb_Validati(object                                    sender,
System.ComponentModel.CancelEventArgs e)
{string quantityText = QuantityTb.Text.Trim();
int quantity;
if (string.IsNullOrEmpty(quantityText))
{MessageBox.Show("Quantity cannot be empty.");
e.Cancel = true;}
else if (!int.TryParse(quantityText, out quantity))
```

```
{MessageBox.Show("Invalid Quantity! Please enter a valid number.");
e.Cancel = true;}
else if (quantity < 0)
{MessageBox.Show("Quantity cannot be negative.");
e.Cancel = true;}
else if (quantity > 1000)
{MessageBox.Show("Quantity cannot exceed 1,000.");
e.Cancel = true;}}
privatevoidBPriceTb_Validating(objectsender,.ComponentModel.CancelEventArgs e)
{string priceText = BPriceTb.Text.Trim();
double productPrice;
if (string.IsNullOrEmpty(priceText))
{MessageBox.Show("BoxPrice cannot be empty.");
e.Cancel = true;}
else if (!double.TryParse(priceText, out productPrice))
{MessageBox.Show("Invalid BoxPrice! Please enter a valid number.");
e.Cancel = true;}
else if (productPrice < 0)
{MessageBox.Show("Price cannot be negative.");
e.Cancel = true;
}else if (productPrice > 10000)
{MessageBox.Show("Price cannot exceed 10,000.");
e.Cancel=true;}}
privatevoidSPriceTb_Validating(objectsender,System.ComponentModel.CancelEvent
Args e)
{string priceText = SPriceTb.Text.Trim();
double salesPrice;
if (string.IsNullOrEmpty(priceText))
{MessageBox.Show("Sales Price cannot be empty.");
e.Cancel = true;}
else if (!double.TryParse(priceText, out salesPrice))
{MessageBox.Show("Invalid Sales Price! Please enter a valid number.");
e.Cancel = true;
}else if (salesPrice < 0)
{MessageBox.Show("Sales Price cannot be negative.");
e.Cancel = true;
}else if (salesPrice > 10000)
{MessageBox.Show("Sales Price cannot exceed 10,000.");
e.Cancel = true;}    }
private void ProdDate_Validating(object sender, CancelEventArgs e)
{DateTime selectedDate = ProdDate.Value.Date;
DateTime today = DateTime.Today;
DateTime minDate = new DateTime(1990, 1, 1);
if (selectedDate > today)
{MessageBox.Show("Invalid Date!!! Product Add Date cannot be in the future.");
e.Cancel = true;}
else if (selectedDate < minDate)
{MessageBox.Show("Invalid Date!!! Product Add Date cannot be before January 1,
1990.");
e.Cancel = true;}}}}
```

CATEGORY MODULE:

```
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.SqlClient;
using System.Text.RegularExpressions;
namespace STATIONERY_SHOP
{public partial class Category : Form
{public Category()
{InitializeComponent();
Methods obj = new Methods();
obj.DisplayData("CategoryTbl", CategoryGDV);}
private void AddBtn_Click(object sender, EventArgs e)
{string categoryName = CatNameTb.Text;
if (CatNameTb.Text == "")
{MessageBox.Show("Missing Information!!!");
}  else
{if (IsCategoryNameValid(categoryName))
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("insert into CategoryTbl(CatName) values
(@CN)", Con);
cmd.Parameters.AddWithValue("@CN", categoryName);
cmd.ExecuteNonQuery();
MessageBox.Show("Category Added!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("CategoryTbl", CategoryGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message);   }}}
private void EditBtn_Click(object sender, EventArgs e)
{string categoryName = CatNameTb.Text;
if (CatNameTb.Text == "")
{MessageBox.Show("Missing Information!!!");
}  else
{if (IsCategoryNameValid(categoryName))
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("update CategoryTbl set CatName=@CN
where CatId=@CK", Con);
cmd.Parameters.AddWithValue("@CN", categoryName);
cmd.Parameters.AddWithValue("@CK", Key);
cmd.ExecuteNonQuery();
MessageBox.Show("Category Updated!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("CategoryTbl", CategoryGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message);}     }}}
int Key = 0;
privatevoidCategoryGDV_CellContentClick(objectsender,
DataGridViewCellEventArgs e)
{if (CategoryGDV.SelectedRows.Count > 0)
{if (CategoryGDV.SelectedRows[0].Cells.Count > 1)
{var cellValue1 = CategoryGDV.SelectedRows[0].Cells[1].Value;
CatNameTb.Text = cellValue1 != null ? cellValue1.ToString() : string.Empty;
if (string.IsNullOrEmpty(CatNameTb.Text))
{Key = 0;
}else
{var cellValue0 = CategoryGDV.SelectedRows[0].Cells[0].Value;
if (cellValue0 != null && int.TryParse(cellValue0.ToString(), out int result))
{Key = result;
}else
{Key = 0;}}}
Else{MessageBox.Show("Selected row does not contain enough cells.");}
}else
{MessageBox.Show("No row is selected.");
CatNameTb.Text = string.Empty;
Key = 0;}}
private void DeleteBtn_Click(object sender, EventArgs e)
{if (Key == 0)
{MessageBox.Show("Missing Information!!!");
}else
{try
{Con.Open();
SqlCommand cmd = new SqlCommand("delete from CategoryTbl where
CatId=@CK", Con);
cmd.Parameters.AddWithValue("@CK", Key);
cmd.ExecuteNonQuery();
MessageBox.Show("Category Deleted!!!");
Con.Close();
Methods obj = new Methods();
obj.DisplayData("CategoryTbl", CategoryGDV);
}catch (Exception Ex)
{MessageBox.Show(Ex.Message);}       }
private bool IsCategoryNameValid(string name)
{if (string.IsNullOrWhiteSpace(name))
{MessageBox.Show("Category Name cannot be empty.");
return false;
}string pattern = @"^[a-zA-Z ]+$";
if (!Regex.IsMatch(name, pattern))
{MessageBox.Show("Category Name should contain only alphabetic characters and
spaces.");
```

```
return false;}
if (name.Length < 4 || name.Length > 25)
{MessageBox.Show("Category Name must be between 4 and 10 characters long.");
return false;
}return true;}}
```

BILLING MODULE:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Printing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.ComponentModel.Design.ObjectSelectorEditor;
using static System.Runtime.InteropServices.JavaScript.JSType;
using PdfSharp.Pdf;
using PdfSharp.Drawing;
using PdfSharp.Fonts;
namespace STATIONERY_SHOP
{public partial class Customer : Form
{private List<BillDetail> billDetails = new List<BillDetail>();
private string printCustomer = "";
private DateTime printDate = DateTime.MinValue;
private string printUser = "";
private decimal printAmount = 0;
private long printPhone = 0;
private int printInvoice = 0;
public Customer()
{InitializeComponent();
DisplayCustomerDetails();}
private void DisplayCustomerDetails()
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SNum AS [Invoice Number], SCustomer AS [Customer Name], SDate AS
[Billing Date], UserTbl.UName AS [Sales Man], SAmount AS [Total Amount]
FROM SalesTbl
INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
DataSet ds = new DataSet();
adapter.Fill(ds);
SalesDGV.DataSource = ds.Tables[0];}}
```

```
private void SearchImg_Click_1(object sender, EventArgs e)
{string searchQuery = @"
SELECT SNum AS [Invoice Number], SCustomer AS [Customer Name], SDate AS
[Billing Date], UserTbl.UName AS [Sales Man], SAmount AS [Total Amount]
FROM SalesTbl
INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId
WHERE 1 = 1";
int sNum;
if (!string.IsNullOrWhiteSpace(SearchTb.Text) && int.TryParse(SearchTb.Text, out
sNum))
{condition += " AND SNum = @SNum";
}
if (SartDate.Value != DateTime.MinValue)
{condition += " AND CONVERT(date, SDate) = @SDate";
}
searchQuery += condition;
if (!string.IsNullOrWhiteSpace(SearchTb.Text) && int.TryParse(SearchTb.Text, out
sNum))
{adapter.SelectCommand.Parameters.AddWithValue("@SNum", sNum);
}if (SartDate.Value != DateTime.MinValue)
{adapter.SelectCommand.Parameters.AddWithValue("@SDate",
SartDate.Value.Date);}
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
}else
{SalesDGV.DataSource = null;
MessageBox.Show("No results found matching the criteria.");}
}catch (Exception ex)
{MessageBox.Show($"Error fetching data: {ex.Message}");}}}
private void PrintBill(int sNum)
{string customer = "";
DateTime date = DateTime.MinValue;
string user = "";
decimal amount = 0;
private void FillSalesGridByDate(DateTime date)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SNum, SDate, SCustomer, SPhone, UT.UName AS SUser, SAmount
FROM SalesTbl ST
INNER JOIN UserTbl UT ON ST.SUser = UT.UId
WHERE CONVERT(date, ST.SDate) = @SDate";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
adapter.SelectCommand.Parameters.AddWithValue("@SDate", date);
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
}
else{SalesDGV.DataSource = null;
MessageBox.Show("No results found for the selected date.");}
}catch (Exception ex)
{MessageBox.Show($"Error fetching SalesTbl data: {ex.Message}");}}}
private void RefreshImg_Click(object sender, EventArgs e)
{DisplayCustomerDetails();}
private void ViewBtn_Click(object sender, EventArgs e)
{try
{if (SalesDGV.SelectedRows.Count > 0)
{int selectedSalesId = Convert.ToInt32(SalesDGV.SelectedRows[0].Cells["Invoice
Number"].Value);
DisplayBillDetails(selectedSalesId);
PreparePrintDocument(selectedSalesId);
PrintPreviewDialog printPreview = new PrintPreviewDialog
{Document = printDocument1};
int totalAmount = CalculateTotalAmount(selectedSalesId);
GrdTotalLbl.Text = $"Total: Rs {totalAmount}";
printPreviewControl1.Document = printDocument1;
printPreviewControl1.Zoom = 1.0; // Set initial zoom level if needed
printPreviewControl1.InvalidatePreview();}
else{MessageBox.Show("Please select a row to view details.", "No Row Selected",
MessageBoxButtons.OK, MessageBoxIcon.Information);    }
} catch (Exception ex)
{MessageBox.Show($"Error in ViewBtn_Click: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}}
private int CalculateTotalAmount(int salesId)
{int totalAmount = 0;
using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SUM(Total) AS TotalAmount
FROM BillDetailsTbl
WHERE SalesId = @SalesId";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@SalesId", salesId);
try{con.Open();
object result = cmd.ExecuteScalar();
if (result != DBNull.Value && result != null)
{totalAmount = Convert.ToInt32(result);}
}catch (Exception ex)
{MessageBox.Show($"Error calculating total amount: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);}
}return totalAmount; }
private void DisplayBillDetails(int salesId)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT ItemNo, ProductName, Quantity, Price, Total
FROM BillDetailsTbl
```

```
WHERE SalesId = @SalesId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
adapter.SelectCommand.Parameters.AddWithValue("@SalesId", salesId);
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
}else
{SalesDGV.DataSource = null;
MessageBox.Show("No bill details found for the selected invoice.", "No Data",
MessageBoxButtons.OK, MessageBoxIcon.Information);}}
catch (Exception ex)
{MessageBox.Show($"Error fetching bill details: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);}}
private void PreparePrintDocument(int salesId)
{FetchSalesDetails(salesId);
using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"SELECT ItemNo, ProductName, Quantity, Price, TotalFROM
BillDetailsTblWHERE SalesId = @SalesId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
adapter.SelectCommand.Parameters.AddWithValue("@SalesId", salesId);
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{billDetails.Clear();
foreach (DataRow row in ds.Tables[0].Rows)
{billDetails.Add(new BillDetail
{ItemNo = row["ItemNo"].ToString(),
ProductName = row["ProductName"].ToString(),
Quantity = Convert.ToInt32(row["Quantity"]),
Price = Convert.ToDecimal(row["Price"]),
Total = Convert.ToDecimal(row["Total"])});}}
}  catch (Exception ex)
{MessageBox.Show($"Error fetching bill details for printing: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);}        }
private void FetchSalesDetails(int salesId)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"SELECT SNum,SCustomer, SDate, UT.UName AS SUser,
SAmount,SPhone FROM SalesTbl ST  INNER JOIN UserTbl UT ON ST.SUser =
UT.UId WHERE ST.SNum = @SalesId";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@SalesId", salesId);
try
{con.Open();
SqlDataReader reader = cmd.ExecuteReader();
if (reader.Read())
{printCustomer = reader["SCustomer"].ToString();
printInvoice = Convert.ToInt32(reader["SNum"]);
printDate = Convert.ToDateTime(reader["SDate"]);
printUser = reader["SUser"].ToString();
printAmount = Convert.ToDecimal(reader["SAmount"]);
printPhone = Convert.ToInt64(reader["SPhone"]);
}reader.Close();
}catch (Exception ex)
{MessageBox.Show($"Error fetching SalesTbl details: {ex.Message}");}}}
private void printDocument1_PrintPage_1(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{int startX = 50; // X-coordinate for starting point
int startY = 50; // Y-coordinate for starting point
int lineSpacing = 20; // Spacing between lines
string title = "STATIONERY SHOP";
Font titleFont = new Font("Algerian", 32, FontStyle.Regular);
SizeF titleSize = e.Graphics.MeasureString(title, titleFont);
e.Graphics.DrawString(title, titleFont, Brushes.LimeGreen, new
PointF((e.PageBounds.Width - titleSize.Width) / 2, startY));
Font CustdetailsFont = new Font("Calibri Light (Headings)", 14, FontStyle.Bold);
string[] customerDetails = {"Customer Name: " + printCustomer,"Date: " +
printDate.ToShortDateString(),"Customer Contact: " + printPhone.ToString()
};
Font detailsFont = new Font("Calibri Light (Headings)", 12, FontStyle.Regular);
for (int i = 0; i < customerDetails.Length; i++)
{e.Graphics.DrawString(customerDetails[i], detailsFont, Brushes.Black, new
PointF(startX, startY + (3 + i) * lineSpacing));
}
string invoiceNumber = "Invoice Number: " + printInvoice.ToString();
SizeF invoiceSize = e.Graphics.MeasureString(invoiceNumber, detailsFont);
e.Graphics.DrawString(invoiceNumber, detailsFont, Brushes.Black, new
PointF(e.PageBounds.Width - invoiceSize.Width - startX, startY + 3 * lineSpacing));
string salesUser = "Sales User: " + printUser;
SizeF salesUserSize = e.Graphics.MeasureString(salesUser, detailsFont);
e.Graphics.DrawString(salesUser, detailsFont, Brushes.Black, new
PointF(e.PageBounds.Width - salesUserSize.Width - startX, startY + 4 * lineSpacing));
e.Graphics.DrawLine(new Pen(Brushes.Black), startX, startY + 7 * lineSpacing,
e.PageBounds.Width - startX, startY + 7 * lineSpacing);
e.Graphics.DrawString("ID PRODUCT QUANTITY PRICE TOTAL",
new Font("Calibri Light (Headings)", 14, FontStyle.Bold), Brushes.Black, new
Point(40, startY + 8 * lineSpacing));
int pos = startY + 10 * lineSpacing;
foreach (BillDetail detail in billDetails)
{e.Graphics.DrawString($"{detail.ItemNo}", new Font("Calibri Light (Headings)", 12,
FontStyle.Italic), Brushes.Black, new Point(43, pos));
e.Graphics.DrawString($"{detail.ProductName}", new Font("Calibri Light
(Headings)", 12, FontStyle.Italic), Brushes.Black, new Point(95, pos));
e.Graphics.DrawString($"{detail.Quantity}", new Font("Calibri Light (Headings)", 12,
FontStyle.Italic), Brushes.Black, new Point(260, pos));
e.Graphics.DrawString($"{detail.Price:C}", new Font("Calibri Light (Headings)", 12,
FontStyle.Italic), Brushes.Black, new Point(390, pos));
```

```
e.Graphics.DrawString($"{detail.Total:C}", new Font("Calibri Light (Headings)", 12,
FontStyle.Italic), Brushes.Black, new Point(490, pos));
pos += lineSpacing;
}
e.Graphics.DrawString("Grand Total: Rs " + printAmount, new Font("Century Gothic",
14, FontStyle.Bold), Brushes.Black, new Point(220, pos + 20));
e.Graphics.DrawString("*~*~*~*~*Thank You! for Your Purchase!*~*~*~*~*", new
Font("Brush Script MT", 18, FontStyle.Italic), Brushes.DeepPink, new PointF(50, pos
+ 55));
}public class BillDetail
{public string ItemNo { get; set; }
public string ProductName { get; set; }
public int Quantity { get; set; }
public decimal Price { get; set; }
public decimal Total { get; set; }}
private void BillPrint_Click(object sender, EventArgs e)
{
try
{printDocument1.Print();
}catch (Exception ex)
{MessageBox.Show($"Error printing bill: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}}}
```

CUSTOMER MODULE:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.ComponentModel.Design.ObjectSelectorEditor;
using static System.Runtime.InteropServices.JavaScript.JSType;
namespace STATIONERY_SHOP
{public partial class Customer : Form
{public Customer()
{InitializeComponent();
DisplayCustomerDetails();}
private void DisplayCustomerDetails()
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SNum AS [Invoice Number], SCustomer AS [Customer Name], SDate AS
[Billing Date], UserTbl.UName AS [Sales Man], SAmount AS [Total Amount]
```

```
FROM SalesTbl
INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
DataSet ds = new DataSet();
adapter.Fill(ds);
SalesDGV.DataSource = ds.Tables[0];}}
private void SearchImg_Click_1(object sender, EventArgs e)
{string searchQuery = @"SELECT ItemNo, ProductName, Quantity, Price, Total
FROM BillDetailsTblWHERE 1 = 1";
if (!string.IsNullOrWhiteSpace(SearchTb.Text))
{condition += " AND SalesId = @SNum";
}
if (SartDate.Value != DateTime.MinValue)
{condition += " AND SalesId IN (SELECT SNum FROM SalesTbl WHERE
CONVERT(date, SDate) = @SDate)";
}searchQuery += condition;
using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{SqlDataAdapter adapter = new SqlDataAdapter(searchQuery, con);
if (!string.IsNullOrWhiteSpace(SearchTb.Text))
{adapter.SelectCommand.Parameters.AddWithValue("@SNum",
int.Parse(SearchTb.Text)); }
if (SartDate.Value != DateTime.MinValue)
{adapter.SelectCommand.Parameters.AddWithValue("@SDate",
SartDate.Value.Date);}
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
if (!string.IsNullOrWhiteSpace(SearchTb.Text))
{PrintBill(int.Parse(SearchTb.Text));
}else if (SartDate.Value != DateTime.MinValue)
{FillSalesGridByDate(SartDate.Value.Date); }
}else
{SalesDGV.DataSource = null;
MessageBox.Show("No results found matching the criteria.");}
}catch (Exception ex)
{MessageBox.Show($"Error fetching data: {ex.Message}");}}}
private void PrintBill(int sNum)
{string customer = "";
DateTime date = DateTime.MinValue;
string user = "";
decimal amount = 0;
using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"SELECT SCustomer, SDate, UT.UName AS SUser, SAmount
FROM SalesTbl ST INNER JOIN UserTbl UT ON ST.SUser = UT.UId  WHERE
ST.SNum = @SalesId";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@SalesId", sNum);
try
```

```
{con.Open();
SqlDataReader reader = cmd.ExecuteReader();
if (reader.Read())
{customer = reader["SCustomer"].ToString();
date = Convert.ToDateTime(reader["SDate"]);
user = reader["SUser"].ToString();
amount = Convert.ToDecimal(reader["SAmount"]);
}reader.Close();
}catch (Exception ex)
{MessageBox.Show($"Error fetching SalesTbl details: {ex.Message}");}
}
private void FillSalesGridByDate(DateTime date)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SNum, SDate, SCustomer, SPhone, UT.UName AS SUser, SAmount
FROM SalesTbl ST
INNER JOIN UserTbl UT ON ST.SUser = UT.UId
WHERE CONVERT(date, ST.SDate) = @SDate";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
adapter.SelectCommand.Parameters.AddWithValue("@SDate", date);
DataSet ds = new DataSet();
try
{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
}else
{
SalesDGV.DataSource = null;
MessageBox.Show("No results found for the selected date.");}
}catch (Exception ex)
{MessageBox.Show($"Error fetching SalesTbl data: {ex.Message}");}}}
private void RefreshImg_Click(object sender, EventArgs e)
{SearchTb.Text = "";
DisplayCustomerDetails();}
private void ViewBtn_Click(object sender, EventArgs e)
{try
{if (SalesDGV.SelectedRows.Count > 0){
int
selectedSalesId=Convert.ToInt32(SalesDGV.SelectedRows[0].Cells["SNum"].Value);
DisplayBillDetails(selectedSalesId);
PrintBill(selectedSalesId);
}else
{MessageBox.Show("Please select a row to view details.", "No Row Selected",
MessageBoxButtons.OK, MessageBoxIcon.Information);}
}catch (Exception ex)
{MessageBox.Show($"Error in ViewBtn_Click: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);}}
private void DisplayBillDetails(int salesId)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT ItemNo, ProductName, Quantity, Price, Total FROM BillDetailsTbl
WHERE SalesId = @SalesId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
adapter.SelectCommand.Parameters.AddWithValue("@SalesId", salesId);
DataSet ds = new DataSet();
try{adapter.Fill(ds);
if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
{SalesDGV.DataSource = ds.Tables[0];
}else
{SalesDGV.DataSource = null;
MessageBox.Show("No bill details found for the selected invoice.", "No Data", MessageBoxButtons.OK, MessageBoxIcon.Information);}
}catch (Exception ex)
{MessageBox.Show($"Error fetching bill details: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);}}}
```

DASHBOARD MODULE:

```
using Microsoft.VisualBasic.ApplicationServices;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization;
using System.Windows.Forms.DataVisualization.Charting;
using static System.Net.Mime.MediaTypeNames;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;
namespace STATIONERY_SHOP
{public partial class Dashboard : Form
{public Dashboard()
{InitializeComponent();
LoadDashboardData();}
private void LoadDashboardData()
{try
{DateTime currentDate = DateTime.Today;
CountItems(null, new DateTime(currentDate.Year, currentDate.Month, 1), currentDate);
SumSales();
ShowChart();
DisplaySales();
DisplaySalary(currentDate.Month);
}catch (Exception ex)
{MessageBox.Show($"An error occurred while loading dashboard data: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }}
```

```
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"SELECT COUNT(*)FROM SalesTblINNER JOIN UserTbl ON
SalesTbl.SUser = UserTbl.UId";
SqlCommand cmd = new SqlCommand(query, con);
if (!string.IsNullOrEmpty(userName))
{query += " WHERE UserTbl.UName = @UName";
cmd.Parameters.AddWithValue("@UName", userName);}
if (startDate != null && endDate != null)
{if (!string.IsNullOrEmpty(userName))
{query += " AND";
}else
{query += " WHERE";}
query += " SDate >= @StartDate AND SDate <= @EndDate";
cmd.Parameters.AddWithValue("@StartDate", startDate.Value);
cmd.Parameters.AddWithValue("@EndDate", endDate.Value);
}else
{query += " WHERE DATEPART(MONTH, SDate) = @Month";
cmd.Parameters.AddWithValue("@Month", DateTime.Today.Month);}
cmd.CommandText = query;
con.Open();
object result = cmd.ExecuteScalar();
if (result != null)
{StockLbl.Text = result.ToString() + " Items";}}}
private void SumSales(string userName = null, DateTime? startDate = null, DateTime?
endDate = null)
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{
string query = "SELECT SUM(SAmount) AS TotalSales FROM SalesTbl WHERE 1 =
1";
if (!string.IsNullOrEmpty(userName))
{query += " AND SUser = (SELECT UId FROM UserTbl WHERE UName =
@UName)"; }
if (startDate != null)
{query += " AND SDate >= @StartDate";
}if (endDate != null)
{query += " AND SDate <= @EndDate";
}
SqlCommand cmd = new SqlCommand(query, con);
if (!string.IsNullOrEmpty(userName))
{cmd.Parameters.AddWithValue("@UName", userName);
}if (startDate != null)
{cmd.Parameters.AddWithValue("@StartDate", startDate.Value);
}if (endDate != null)
{cmd.Parameters.AddWithValue("@EndDate", endDate.Value);}
con.Open();
object result = cmd.ExecuteScalar();
if (result != null && result != DBNull.Value)
{SalesLbl.Text = "Rs " + result.ToString();
}else
{SalesLbl.Text = "Rs 0";}}}
```

```
private void ShowChart()
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = "SELECT SUM(SAmount) AS Amount, SDate FROM SalesTbl
GROUP BY SDate";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
DataTable dt = new DataTable();
adapter.Fill(dt);
Chart1.Series.Clear();
Chart1.ChartAreas.Clear();
Chart1.ChartAreas.Add(new ChartArea("ChartArea1"));
Series series = new Series("SalesSeries");
series.ChartType = SeriesChartType.SplineArea;
series.XValueType = ChartValueType.DateTime;
series.YValueType = ChartValueType.Double;
foreach (DataRow row in dt.Rows)
{DateTime date = Convert.ToDateTime(row["SDate"]);
double amount = Convert.ToDouble(row["Amount"]);
series.Points.AddXY(date, amount);}
Chart1.Series.Add(series);}}
private void DisplaySales()
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SCustomer, SDate, UserTbl.UName AS SUser, SAmount
FROM SalesTbl INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId";
SqlDataAdapter adapter = new SqlDataAdapter(query, con);
DataSet ds = new DataSet();
adapter.Fill(ds);
SalesDGV.DataSource = ds.Tables[0];}}

private void SearchUser(string userName, DateTime startDate, DateTime endDate)
{try
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SCustomer, SDate, UserTbl.UName AS SUser, SAmountFROM SalesTbl
INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId WHERE UserTbl.UName =
@UNameAND SDate >= @StartDateAND SDate <= @EndDate";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@UName", userName);
cmd.Parameters.AddWithValue("@StartDate", startDate);
cmd.Parameters.AddWithValue("@EndDate", endDate);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
adapter.Fill(ds);
SalesDGV.DataSource = ds.Tables[0];
CountItems(userName, startDate, endDate);
SumSales(userName, startDate, endDate);}
}catch (Exception ex)
{MessageBox.Show($"An error occurred while searching user: {ex.Message}",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);}}
private void UpdateChart(string userName, DateTime startDate, DateTime endDate)
```

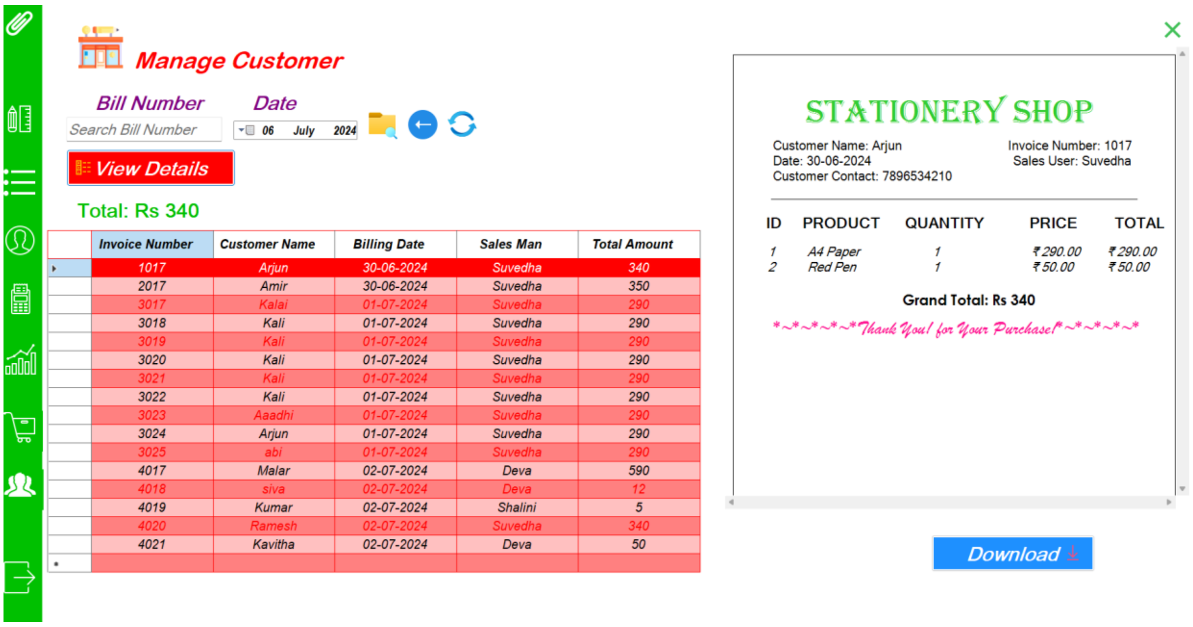
```
{try
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"
SELECT SDate, SUM(SAmount) AS AmountFROM SalesTbl INNER JOIN UserTbl
ON SalesTbl.SUser = UserTbl.UId WHERE UserTbl.UName = @UName
AND SDate >= @StartDateAND SDate <= @EndDateGROUP BY SDate";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@UName", userName);
cmd.Parameters.AddWithValue("@StartDate", startDate);
cmd.Parameters.AddWithValue("@EndDate", endDate);
con.Open();
SqlDataReader reader = cmd.ExecuteReader();
Chart1.Series.Clear();
Chart1.ChartAreas.Clear();
Chart1.ChartAreas.Add(new ChartArea("ChartArea1"));
Series series = new Series("SalesSeries");
series.ChartType = SeriesChartType.SplineArea;
series.XValueType = ChartValueType.DateTime;
series.YValueType = ChartValueType.Double;
while (reader.Read())
{DateTime date = Convert.ToDateTime(reader["SDate"]);
double amount = Convert.ToDouble(reader["Amount"]);
series.Points.AddXY(date, amount);}
Chart1.Series.Add(series);
reader.Close();}
}catch (Exception ex)
{MessageBox.Show($"An error occurred while updating chart: {ex.Message}",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);}}

private void SearchImg_Click(object sender, EventArgs e)
{try
{string userName = SearchTb.Text;
DateTime startDate = DateTime.Parse(SartDate.Text);
DateTime endDate = DateTime.Parse(EndDate.Text);
SearchUser(userName, startDate, endDate);
UpdateChart(userName, startDate, endDate);}
catch (FormatException ex)
{MessageBox.Show("Invalid date format entered. Please enter a valid date.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);}}
private void ShowSales()
{try
{Con.Open();
string Query = @"SELECT SCustomer, SDate, UserTbl.UName AS SUser, SAmount
FROM SalesTbl INNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId";
SqlDataAdapter stockSda = new SqlDataAdapter(Query, Con);
SqlCommandBuilder stockBuilder = new SqlCommandBuilder(stockSda);
var stockDs = new DataSet();
stockSda.Fill(stockDs);
SalesDGV.DataSource = stockDs.Tables[0];
}catch (Exception ex)
{MessageBox.Show("An error occurred while loading sales data: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
}finally
{Con.Close();}}
private void RefreshImg_Click(object sender, EventArgs e)
{try
{ShowChart();
DisplaySales();
SearchTb.Text = "";
DateTime currentMonth = DateTime.Today;
CountItems(null, new DateTime(currentMonth.Year, currentMonth.Month, 1), currentMonth);
SumSales();
}catch (Exception ex)
{MessageBox.Show($"An error occurred while refreshing data: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);}}
private void DisplaySalary(int monthNumber)
{try
{using (SqlConnection con = new SqlConnection(Con.ConnectionString))
{string query = @"SELECT UserTbl.UName AS SUser,DATENAME(MONTH, SalesTbl.SDate) AS MonthName,SUM(SalesTbl.SAmount) AS TotalSales,CASE
WHEN SUM(SalesTbl.SAmount) > 1000 THEN 2000ELSE 500END AS Salary
FROM SalesTblINNER JOIN UserTbl ON SalesTbl.SUser = UserTbl.UId WHERE DATEPART(MONTH, SalesTbl.SDate) = @MonthNumberGROUP BY UserTbl.UName, DATENAME(MONTH, SalesTbl.SDate)ORDER BY UserTbl.UName";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@MonthNumber", monthNumber);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
adapter.Fill(ds);
SalaryDGV.Columns.Clear();
SalaryDGV.DataSource = ds.Tables[0];
SalaryDGV.Columns["SUser"].HeaderText = "User";
SalaryDGV.Columns["MonthName"].HeaderText = "Month";
SalaryDGV.Columns["TotalSales"].HeaderText = "Total Sales";
SalaryDGV.Columns["Salary"].HeaderText = "Salary";}
}catch (Exception ex)
{MessageBox.Show($"An error occurred while displaying salary: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);}}}}
```

STOCK MODULE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace STATIONERY_SHOP
{public partial class Stock : Form
{public Stock()
{InitializeComponent();
CountItems();
DisplayStockData();
DisplayOutStockData();
DisplaySoonOutStockData();}
private void SearchItem()
{try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string stockQuery = "SELECT ItId, ItName, ItQty FROM ItemTbl WHERE
ItName=@ItName AND ItQty > 0";
SqlDataAdapter stockSda = new SqlDataAdapter(stockQuery, con);
stockSda.SelectCommand.Parameters.AddWithValue("@ItName", SearchTb.Text);
DataTable stockDt = new DataTable();
stockSda.Fill(stockDt);
StockDGV.DataSource = stockDt;
string outstockQuery = "SELECT ItId, ItName FROM ItemTbl WHERE
ItName=@ItName AND ItQty = 0";
SqlDataAdapter outstockSda = new SqlDataAdapter(outstockQuery, con);
outstockSda.SelectCommand.Parameters.AddWithValue("@ItName",
SearchTb.Text);
DataTable outstockDt = new DataTable();
outstockSda.Fill(outstockDt);
OutstockDGV.DataSource = outstockDt;
string soonOutstockQuery = "SELECT ItId, ItName FROM ItemTbl WHERE
ItName=@ItName AND ItQty < 50 AND ItQty > 0";
SqlDataAdapter soonOutstockSda = new SqlDataAdapter(soonOutstockQuery, con);
soonOutstockSda.SelectCommand.Parameters.AddWithValue("@ItName",
SearchTb.Text);
DataTable soonOutstockDt = new DataTable();
soonOutstockSda.Fill(soonOutstockDt);
MiniDGV.DataSource = soonOutstockDt;}
}catch (Exception ex)
{MessageBox.Show("An error occurred: " + ex.Message);}}
private void DisplayStockData()
{try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string query = "SELECT ItId, ItName, ItQty FROM ItemTbl WHERE ItQty > 0";
SqlDataAdapter sda = new SqlDataAdapter(query, con);
DataTable dt = new DataTable();
sda.Fill(dt);
StockDGV.DataSource = dt;}}
catch (Exception ex)
```

```
{MessageBox.Show("An error occurred: " + ex.Message);
}}
private void DisplayOutStockData()
{try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string query = "SELECT ItId, ItName FROM ItemTbl WHERE ItQty = 0";
SqlDataAdapter sda = new SqlDataAdapter(query, con);
DataTable dt = new DataTable();
sda.Fill(dt);
OutstockDGV.DataSource = dt;}
}catch (Exception ex)
{MessageBox.Show("An error occurred: " + ex.Message);}
}
private void DisplaySoonOutStockData()
{try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string query = "SELECT ItId, ItName FROM ItemTbl WHERE ItQty <50 AND ItQty
> 0";
SqlDataAdapter sda = new SqlDataAdapter(query, con);
DataTable dt = new DataTable();
sda.Fill(dt);
MiniDGV.DataSource = dt;}
}catch (Exception ex)
{MessageBox.Show("An error occurred: " + ex.Message);}}
private void SearchImg_Click(object sender, EventArgs e)
{SearchItem();}
private void ShowItem()
{
try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string stockQuery = "SELECT ItId, ItName, ItQty FROM ItemTbl WHERE ItQty > 0";
SqlDataAdapter stockSda = new SqlDataAdapter(stockQuery, con);
SqlCommandBuilder stockBuilder = new SqlCommandBuilder(stockSda);
var stockDs = new DataSet();
stockSda.Fill(stockDs);
StockDGV.DataSource = stockDs.Tables[0];
string outstockQuery = "SELECT ItId, ItName FROM ItemTbl WHERE ItQty = 0";
SqlDataAdapter outstockSda = new SqlDataAdapter(outstockQuery, con);
SqlCommandBuilder outstockBuilder = new SqlCommandBuilder(outstockSda);
var outstockDs = new DataSet();
outstockSda.Fill(outstockDs);
OutstockDGV.DataSource = outstockDs.Tables[0];
string SoonoutstockQuery = "SELECT ItId, ItName FROM ItemTbl WHERE ItQty =
0";
SqlDataAdapter SoonoutstockSda = new SqlDataAdapter(SoonoutstockQuery, con);
SqlCommandBuilder SoonoutstockBuilder = new
SqlCommandBuilder(SoonoutstockSda);
var SoonoutstockDs = new DataSet();
outstockSda.Fill(SoonoutstockDs);
OutstockDGV.DataSource = SoonoutstockDs.Tables[0];}
}catch (Exception ex)
{MessageBox.Show("An error occurred: " + ex.Message);}}
private void RefreshImg_Click(object sender, EventArgs e)
{ShowItem();
SearchTb.Text = "";}
private void CountItems()
{try
{using (SqlConnection con = new SqlConnection(connectionString))
{con.Open();
string queryStock = "SELECT COUNT(*) FROM Itemtbl WHERE ItQty > 0";
using (SqlDataAdapter sdaStock = new SqlDataAdapter(queryStock, con))
{DataTable dtStock = new DataTable();
sdaStock.Fill(dtStock);
if (dtStock.Rows.Count > 0)
{StockLbl.Text = dtStock.Rows[0][0].ToString() + " Items";}
else{StockLbl.Text = "0 Items";}}
string queryOutStock = "SELECT COUNT(*) FROM Itemtbl WHERE ItQty = 0";
using (SqlDataAdapter sdaOutStock = new SqlDataAdapter(queryOutStock, con))
{DataTable dtOutStock = new DataTable();
sdaOutStock.Fill(dtOutStock);
if (dtOutStock.Rows.Count > 0)
{OutStockLbl.Text = dtOutStock.Rows[0][0].ToString() + " Items";
} else
{OutStockLbl.Text = "0 Items";}}
string querySoonOutStock = "SELECT COUNT(*) FROM Itemtbl WHERE ItQty < 50
AND ItQty > 0";
using (SqlDataAdapter sdaSoonOutStock = new SqlDataAdapter(querySoonOutStock,
con))
{DataTable dtSoonOutStock = new DataTable();
sdaSoonOutStock.Fill(dtSoonOutStock);
if (dtSoonOutStock.Rows.Count > 0)
{MiniLbl.Text = dtSoonOutStock.Rows[0][0].ToString() + " Items";
}else
{ MiniLbl.Text = "0 Items";} } } }
catch (Exception ex)
{    MessageBox.Show("An error occurred: " + ex.Message);}} }}
```

7.OUTPUT



Manage Customer

Bill Number | Date

Search Bill Number | 06 July 2024

View Details

Total: Rs 340

	Invoice Number	Customer Name	Billing Date	Sales Man	Total Amount
▸	1017	Arjun	30-06-2024	Suvedha	340
	2017	Amir	30-06-2024	Suvedha	350
	3017	Kalai	01-07-2024	Suvedha	290
	3018	Kali	01-07-2024	Suvedha	290
	3019	Kali	01-07-2024	Suvedha	290
	3020	Kali	01-07-2024	Suvedha	290
	3021	Kali	01-07-2024	Suvedha	290
	3022	Kali	01-07-2024	Suvedha	290
	3023	Aaadhi	01-07-2024	Suvedha	290
	3024	Arjun	01-07-2024	Suvedha	290
	3025	abi	01-07-2024	Suvedha	290
	4017	Malar	02-07-2024	Deva	590
	4018	siva	02-07-2024	Deva	12
	4019	Kumar	02-07-2024	Shalini	5
	4020	Ramesh	02-07-2024	Suvedha	340
	4021	Kavitha	02-07-2024	Deva	50
*					

STATIONERY SHOP

Customer Name: Arjun
Date: 30-06-2024
Customer Contact: 7896534210

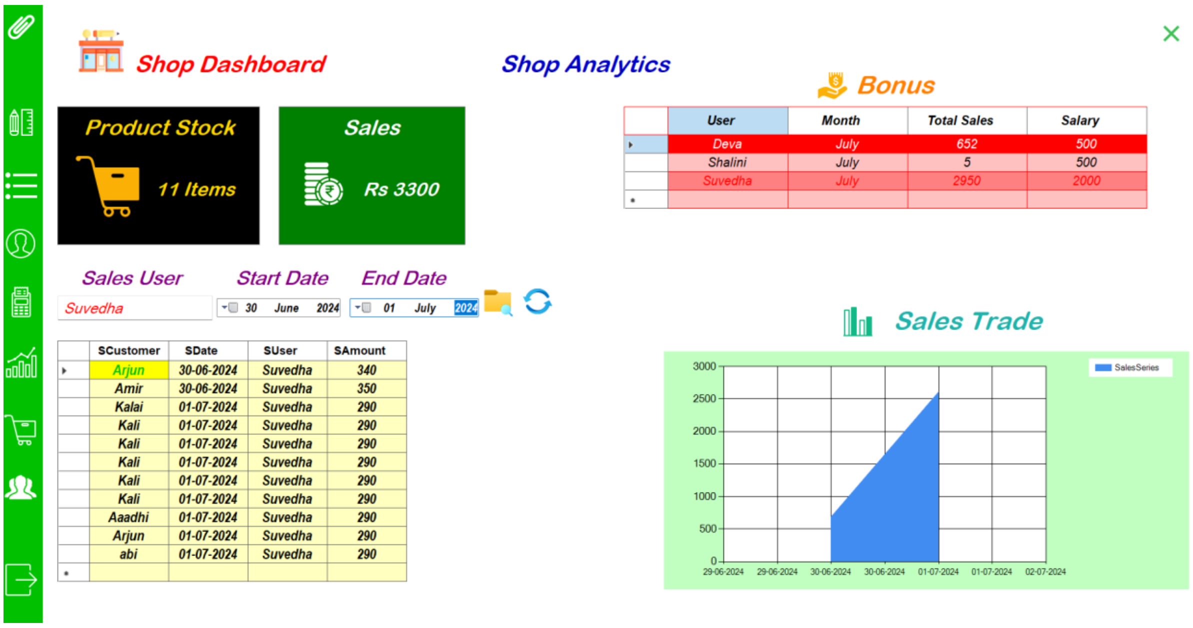
Invoice Number: 1017
Sales User: Suvedha

ID	PRODUCT	QUANTITY	PRICE	TOTAL
1	A4 Paper	1	₹290.00	₹290.00
2	Red Pen	1	₹50.00	₹50.00

Grand Total: Rs 340

~~*~*~*Thank You! for Your Purchase!*~*~*~*~*

Download



Shop Dashboard

Product Stock: 11 Items

Sales: Rs 3300

Sales User | Start Date | End Date

Suvedha | 30 June 2024 | 01 July 2024

	SCustomer	SDate	SUser	SAmount
▸	Arjun	30-06-2024	Suvedha	340
	Amir	30-06-2024	Suvedha	350
	Kalai	01-07-2024	Suvedha	290
	Kali	01-07-2024	Suvedha	290
	Kali	01-07-2024	Suvedha	290
	Kali	01-07-2024	Suvedha	290
	Kali	01-07-2024	Suvedha	290
	Kali	01-07-2024	Suvedha	290
	Aaadhi	01-07-2024	Suvedha	290
	Arjun	01-07-2024	Suvedha	290
	abi	01-07-2024	Suvedha	290
*				

Shop Analytics

Bonus

	User	Month	Total Sales	Salary
▸	Deva	July	652	500
	Shalini	July	5	500
	Suvedha	July	2950	2000
*				

Sales Trade