AMERICAN INTERNATIONAL UNIVERISTY-BANGLADESH (AIUB)

Faculty of Science & Technology

Object Oriented Programming – 2

Summer 2023 – 2024

Section: J

Project Title

Pharmacy Application Management System

Supervised By

DR. MD IFTEKHARUL MOBIN

ASSISTANT PROFESSOR, Faculty, DEPARTMENT OF COMPUTER SCIENCE

Submitted By

NAME	ID
S. S. ZOBAER AHMED	22-49415-3
KHYRUL ALAM	22-49398-3

Table Of Contents

Introduction

- **Background**

  Pharmacies have traditionally managed their operations through manual processes, from handling prescriptions to managing inventory and customer transactions. However, as the pharmaceutical industry grows more complex, these manual methods are becoming increasingly inefficient and error-prone. The increasing demand for pharmaceuticals and the complexity of modern-day pharmacy management make it harder to rely solely on manual systems.

  One of the significant challenges pharmacies face in managing operations manually is inventory control. Keeping accurate track of stock levels without automation is not only time-consuming but also prone to errors. These errors can result in either overstocking or, more worryingly, understocking of essential medications. Inaccuracies in recording quantities, batch numbers, and expiry dates can lead to serious risks, such as dispensing expired or incorrect medications, potentially endangering patient health.

  Similarly, manual order processing is inefficient and prone to delays, particularly during busy hours when multiple customers need to be served. Recording transactions by hand, processing payments, and generating receipts not only slow down the service but also increases the likelihood of errors. This, in turn, affects the customer experience and can lead to dissatisfaction or complaints, especially when mistakes occur in handling orders or billing.

  The handling of prescriptions is another critical area where manual systems reveal their limitations. Processing prescriptions without the aid of technology increases the likelihood of human error. Pharmacies risk providing incorrect dosages or medications due to misinterpretation or simple mistakes in reading handwritten prescriptions. Furthermore, the manual verification of prescriptions is labor-intensive and can lead to delays, further slowing down the pharmacy's operations.

  In addition to inventory and order processing, pharmacies must manage their financial operations, which include billing, accounting, and payroll. Manual methods often result in discrepancies in pricing or billing, leading to either a loss of revenue for the pharmacy or overcharging of customers, both of which have serious implications. Calculating taxes, applying discounts, and tracking financial data all become more difficult when handled manually, making it harder for pharmacy owners to maintain accurate financial records and evaluate the business's financial health.

Moreover, staff management, including tracking attendance and calculating salaries, is significantly more challenging without an automated system. For a pharmacy, ensuring proper records of employee working hours and generating payroll on time is crucial. Manual systems often cause delays or errors in salary payments, which can lead to dissatisfaction among staff and reduce overall productivity.

In conclusion, pharmacies that continue to rely on manual methods face a host of challenges that affect efficiency, accuracy, and customer satisfaction. As the need for effective pharmacy management continues to grow, it is increasingly clear that modern, automated systems are essential to streamline operations and minimize the risks associated with manual processes.

- ## Objective

The primary objective of the Pharmacy Management System is to streamline the daily operations of a pharmacy by automating crucial functions such as inventory management, order processing, and customer transactions. This automation aims to improve overall efficiency while reducing the likelihood of human error. Ensuring accuracy in inventory is another key focus, with the system designed to provide real-time updates on stock levels, track expiration dates, and issue alerts for low stock or expired medications.

A central goal is also to enhance customer service by reducing wait times and improving customer satisfaction through faster order processing, secure payment methods, and efficient prescription handling. Data security is a critical concern, and the system will incorporate robust security mechanisms, such as user authentication and data encryption, to protect sensitive information like customer data, transaction history, and prescription records.

In addition, the system is expected to generate detailed reports in real-time, covering areas like sales, stock levels, employee performance, and financial metrics. This will empower management with the insights needed to make informed decisions and evaluate the business's performance. The system will also include role-based access controls, ensuring that different users, from admins to staff, can access specific functionalities relevant to their roles.

Another significant objective is to optimize financial management by automating the billing and invoicing process, ensuring that all sales, taxes, and payroll information are accurately tracked. The system is also built to comply with relevant pharmaceutical regulations, ensuring proper handling of prescriptions and minimizing legal risks.

Additionally, enhancing user experience is a priority, and the system is designed with an intuitive interface that ensures ease of use for both staff and customers. Finally, the system will allow users to view their order history and track the status of their current orders, adding transparency and fostering trust between the pharmacy and its customers.

- ## Scope

The Pharmacy Management System is designed to automate and streamline the operations of a pharmacy, covering various aspects such as inventory management, sales processing, customer service, employee management, and financial tracking. This system will cater to both administrative and operational tasks, ensuring a more efficient and error-free environment. It will provide role-based access for different users like administrators, pharmacists, and staff, each with specific privileges tailored to their responsibilities.

Inventory management, a critical aspect of the system, will include features like real-time stock updates, expiration tracking, and automated reordering alerts. In terms of sales, the system will handle order processing, secure payment integration, and real-time transaction updates, ensuring accurate billing and invoicing. The system will also support multi-location management, allowing larger pharmacies or pharmacy chains to operate under a single platform.

Furthermore, the system will comply with pharmaceutical regulations, managing sensitive data such as patient records and prescription histories securely. Customers will also benefit from features such as order tracking, viewing order history, and making online payments, thus enhancing the overall customer experience. The system’s scope extends to reporting capabilities, providing detailed analytics on sales, inventory, employee performance, and financial summaries to aid in decision-making.

- ## Significance

The significance of implementing a Pharmacy Management System lies in the transformative impact it will have on the efficiency and accuracy of pharmacy operations. By automating essential processes, the system reduces the margin for human error, particularly in areas like inventory management and prescription handling, which are critical for patient safety. The system also enhances operational efficiency, ensuring that tasks such as order processing, billing, and inventory control are performed faster and with greater precision.

From a business perspective, this system offers invaluable insights through detailed reports on sales trends, stock levels, and financial performance, allowing management to make data-driven decisions that can improve profitability and operational efficiency. Additionally, the secure handling of sensitive information, such as customer records and prescription data, not only ensures compliance with healthcare regulations but also builds trust with customers, reinforcing the pharmacy’s reputation for reliability and safety.

For customers, the system provides greater convenience through faster service, the ability to track orders, and the option to view their order history. This enhances customer satisfaction and loyalty, which is vital in a competitive marketplace. Furthermore, the automation of financial processes, including billing and payroll management, minimizes discrepancies and ensures transparency in the pharmacy's financial operations.

Ultimately, the system will not only improve the day-to-day operations of the pharmacy but also provide a scalable solution that can grow with the business, making it a significant tool for modernizing pharmacy management.

System Design & Architecture

- **System Overview**

The Pharmacy Management System is designed to make running a pharmacy easier and more efficient. It automates many of the important tasks that a pharmacy must handle daily. The system is built with different user roles in mind, like administrators, pharmacists, and staff. Each of these roles has access to certain parts of the system based on what they need to do, ensuring that the pharmacy's data stays secure while giving users the tools they need to do their jobs.

At the heart of the system is a central database. This database stores information about products, stock levels, sales, customers, employees, and financial transactions. By having everything stored in one place, the system keeps the information up-to-date and accurate. It also connects with payment methods to make sure transactions are processed smoothly and securely, following the rules and regulations for handling sensitive customer and prescription data.

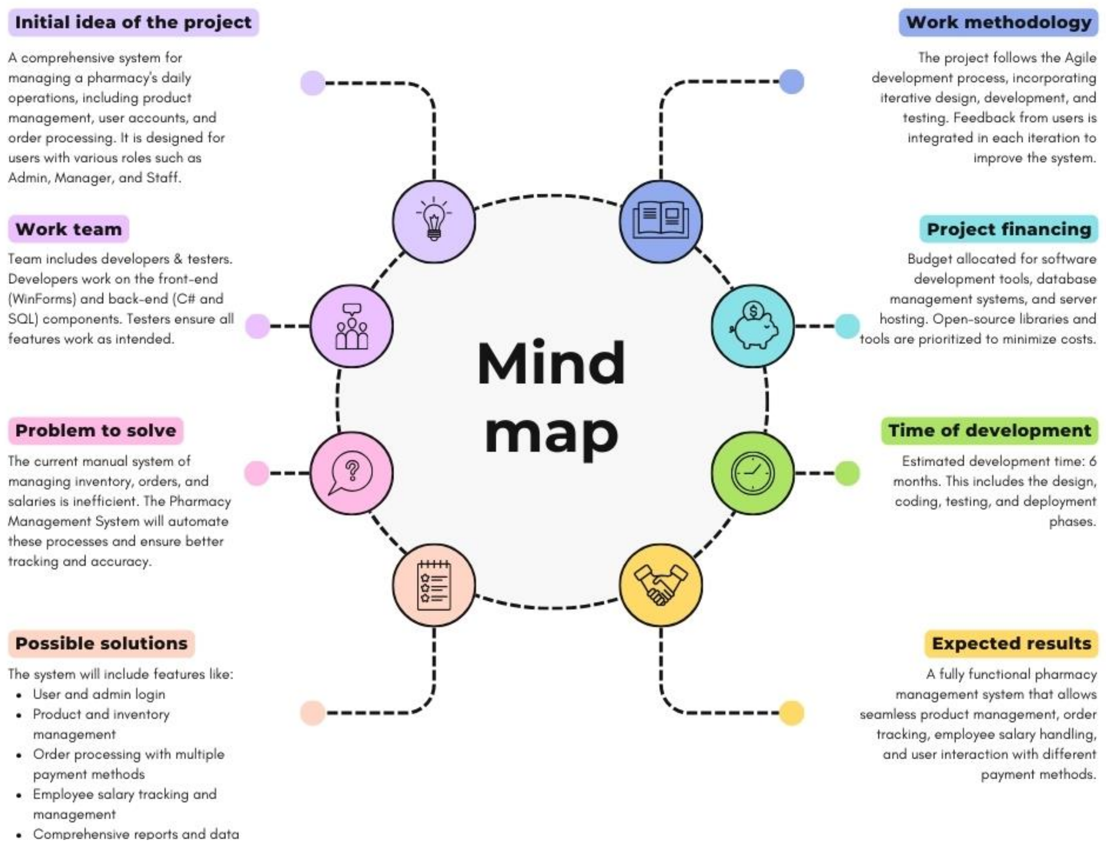
The system is designed to be easy to use, with an interface that helps users quickly move between different functions. For example, it includes an inventory management feature that tracks stock levels in real time and sends alerts when it's time to restock. It also manages orders, billing, and payments, reducing the chance of mistakes and saving time.

The system can grow with the pharmacy, meaning it can handle more users and transactions as the business expands without slowing down. It's also flexible, allowing new features to be added easily in the future. Built-in reporting tools provide useful insights into things like sales, employee performance, and stock levels, helping pharmacy owners make better decisions.

Overall, this system simplifies both the day-to-day operations and management tasks of a pharmacy, making it a valuable tool for improving efficiency and reducing errors.

- ## Mind Map

Here's a summary of our initial mind mapping and design planning for the project. This mind mapping was designed before starting our project



Initial idea of the project

A comprehensive system for managing a pharmacy's daily operations, including product management, user accounts, and order processing. It is designed for users with various roles such as Admin, Manager, and Staff.

Work team

Team includes developers & testers. Developers work on the front-end (WinForms) and back-end (C# and SQL) components. Testers ensure all features work as intended.

Problem to solve

The current manual system of managing inventory, orders, and salaries is inefficient. The Pharmacy Management System will automate these processes and ensure better tracking and accuracy.

Possible solutions

The system will include features like:

- User and admin login
- Product and inventory management
- Order processing with multiple payment methods
- Employee salary tracking and management
- Comprehensive reports and data

Work methodology

The project follows the Agile development process, incorporating iterative design, development, and testing. Feedback from users is integrated in each iteration to improve the system.

Project financing

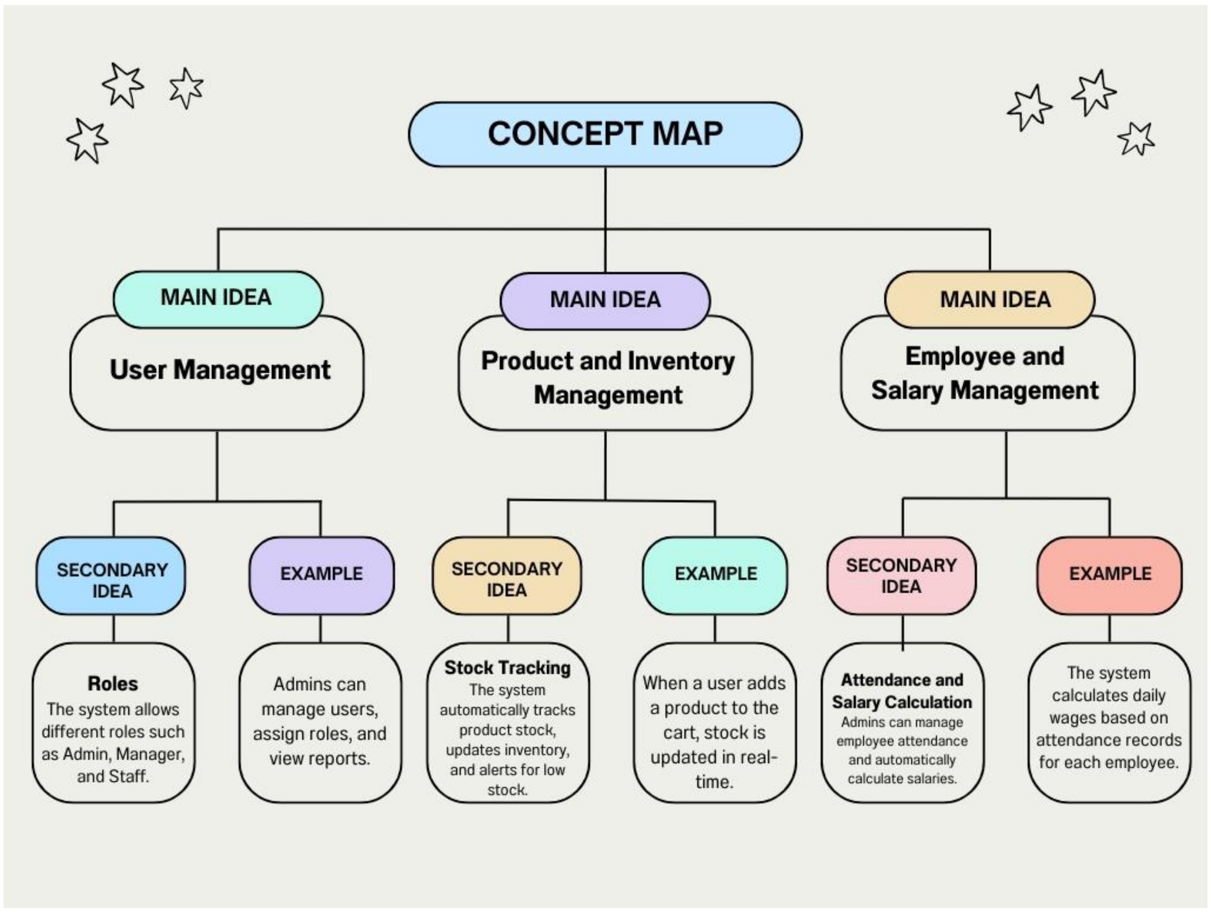
Budget allocated for software development tools, database management systems, and server hosting. Open-source libraries and tools are prioritized to minimize costs.

Time of development

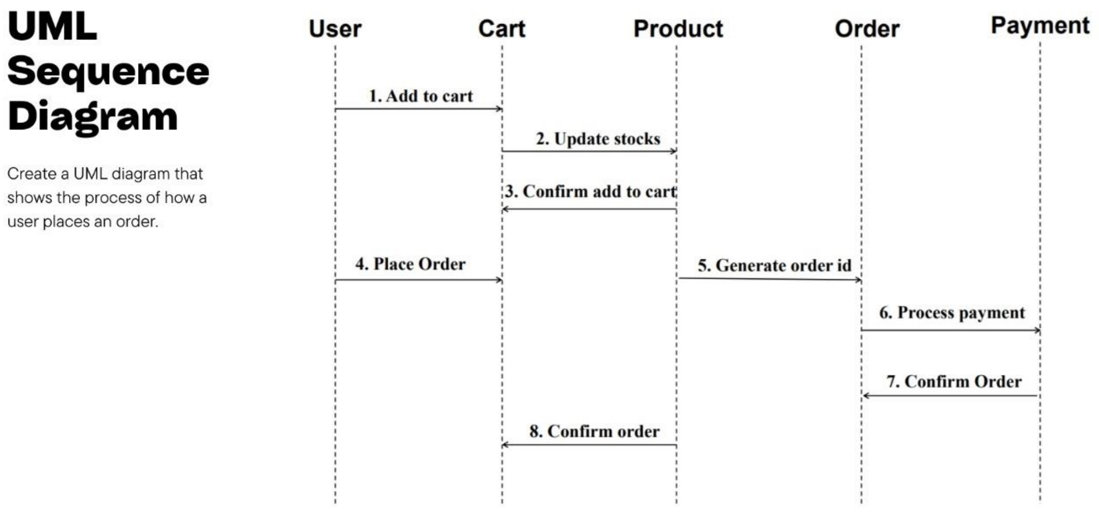
Estimated development time: 6 months. This includes the design, coding, testing, and deployment phases.

Expected results

A fully functional pharmacy management system that allows seamless product management, order tracking, employee salary handling, and user interaction with different payment methods.

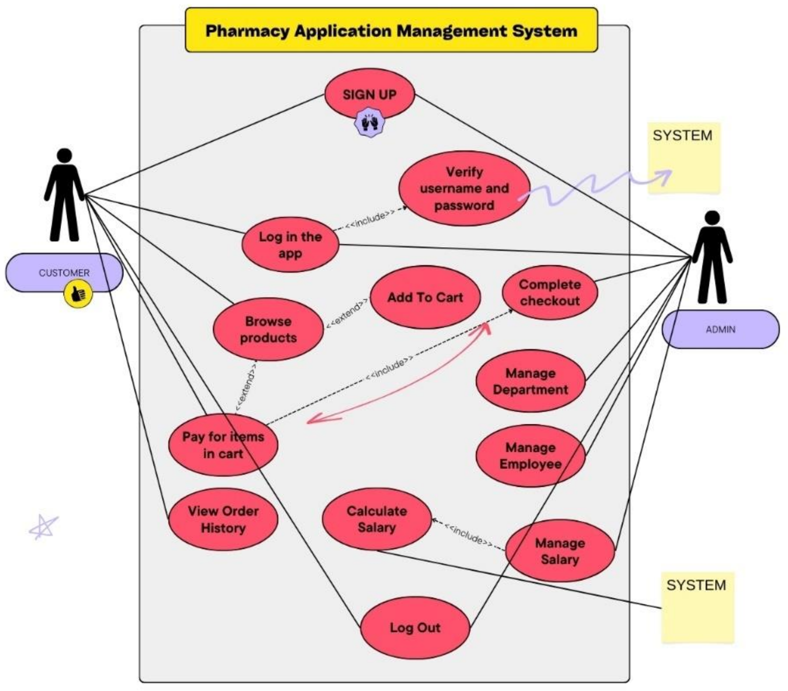

CONCEPT MAP
MAIN IDEA
User Management
MAIN IDEA
Product and Inventory Management
MAIN IDEA
Employee and Salary Management
SECONDARY IDEA
Roles
The system allows different roles such as Admin, Manager, and Staff.
EXAMPLE
Admins can manage users, assign roles, and view reports.
SECONDARY IDEA
Stock Tracking
The system automatically tracks product stock, updates inventory, and alerts for low stock.
EXAMPLE
When a user adds a product to the cart, stock is updated in real-time.
SECONDARY IDEA
Attendance and Salary Calculation
Admins can manage employee attendance and automatically calculate salaries.
EXAMPLE
The system calculates daily wages based on attendance records for each employee.

- **UML Diagrams**

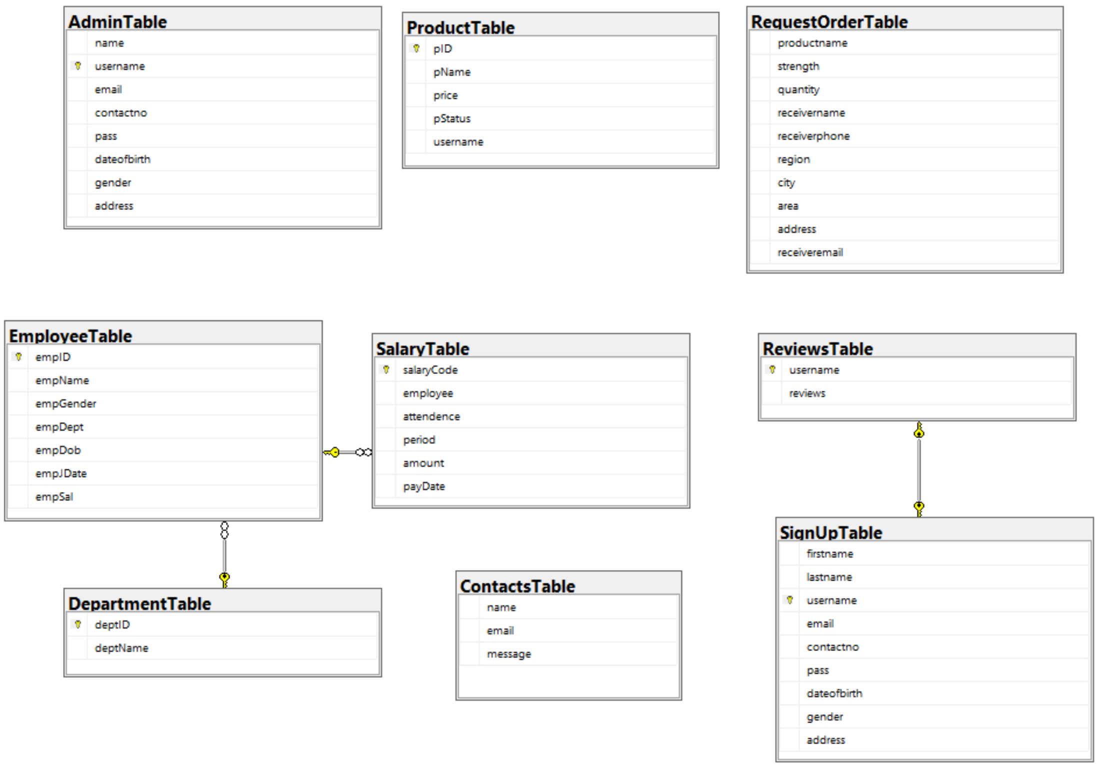


UML Use Case Diagram

Build a UML diagram that shows how users will interact with the Pharmacy app.



- **Database Schema Diagram**



- ## System Components
    - **User Management**: User registration, login, and role-based access control.
    - **Product Management**: Adding, updating, and managing product inventory.
    - **Order System**: Cart functionality and payment methods.
    - **Department, Employee and Salary Management**: Add Department, Add new Employee, Attendance tracking and salary calculation.

Database

- ## Database Creation

Query:

```
CREATE DATABASE Pharmacy Management System
```

- ## Tables

SignUpTable Creation Query:

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SignUpTable](
	[firstname] [varchar](50) NULL,
	[lastname] [varchar](50) NULL,
	[username] [varchar](50) NOT NULL,
	[email] [varchar](50) NULL,
	[contactno] [int] NULL,
	[pass] [varchar](50) NULL,
	[dateofbirth] [date] NULL,
	[gender] [varchar](50) NULL,
	[address] [varchar](50) NULL,
 CONSTRAINT [PK_SignUpTable] PRIMARY KEY CLUSTERED
(
	[username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Visual Representation:

Column Name	Data Type	Allow Nulls
firstname	varchar (50)	Not Null
lastname	varchar (50)	Not Null
username (Primary Key)	varchar (50)	Not Null
email	varchar (50)	Not Null
contactno	int	Not Null
pass	varchar (50)	Not Null
dateofbirth	date	Not Null
gender	varchar (50)	Not Null
address	varchar (50)	Not Null

AdminTable Creation Quey:

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AdminTable](
	[name] [varchar](50) NULL,
	[username] [varchar](50) NOT NULL,
	[email] [varchar](50) NULL,
	[contactno] [int] NULL,
	[pass] [varchar](50) NULL,
	[dateofbirth] [date] NULL,
	[gender] [varchar](50) NULL,
	[address] [varchar](50) NULL,
 CONSTRAINT [PK_AdminTable] PRIMARY KEY CLUSTERED
(
	[username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Visual Representation:

Column Name	Data Type	Allow Nulls
name	varchar (50)	Not Null
username (Primary Key)	varchar (50)	Not Null
email	varchar (50)	Not Null
contactno	int	Not Null
pass	varchar (50)	Not Null
dateofbirth	date	Not Null
gender	varchar (50)	Not Null

address	varchar (50)	Not Null

<u>DepartmentTable Creation Query</u>:

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[DepartmentTable](
	[deptID] [int] IDENTITY(1,1) NOT NULL,
	[deptName] [varchar](50) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[deptID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Visual Representation:

Column Name	Data Type	Allow Nulls
deptID (Primary Key)	int	Not Null
deptName	varchar(50)	Not Null

<u>EmployeeTable Creation Query:</u>

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[EmployeeTable](
	[empID] [int] IDENTITY(1,1) NOT NULL,
	[empName] [varchar](50) NOT NULL,
	[empGender] [varchar](50) NOT NULL,
	[empDept] [int] NOT NULL,
	[empDob] [date] NOT NULL,
	[empJDate] [date] NOT NULL,
	[empSal] [varchar](50) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[empID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[EmployeeTable]  WITH CHECK ADD
CONSTRAINT [FK1] FOREIGN KEY([empDept])
REFERENCES [dbo].[DepartmentTable] ([deptID])
GO
ALTER TABLE [dbo].[EmployeeTable] CHECK CONSTRAINT [FK1]
```

Visual Representation:

Column Name	**Data Type**	**Allow Nulls**
empID (Primary Key)	Int	Not Null
empName	varchar (50)	Not Null
empGender	varchar (50)	Not Null
empDept (Foreign Key)	varchar (50)	Not Null
empDob	date	Not Null
empJDate	date	Not Null
empSal	varchar (50)	Not Null

SalaryTable Creation Query:

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SalaryTable](
	[salaryCode] [int] IDENTITY(1,1) NOT NULL,
	[employee] [int] NOT NULL,
	[attendence] [int] NOT NULL,
	[period] [varchar](50) NOT NULL,
	[amount] [int] NOT NULL,
	[payDate] [date] NOT NULL,
PRIMARY KEY CLUSTERED
(
	[salaryCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[SalaryTable]  WITH CHECK ADD  CONSTRAINT
[FK2] FOREIGN KEY([employee])
REFERENCES [dbo].[EmployeeTable] ([empID])
```

```
GO
ALTER TABLE [dbo].[SalaryTable] CHECK CONSTRAINT [FK2]
```

Visual Represenation:

Column Name	**Data Type**	**Allow Nulls**
salaryCode (Primary Key)	int	Not Null
employee (Foreign Key)	int	Not Null
attendence	int	Not Null
period	Varchar (50)	Not Null
amount	int	Not Null
payDate	date	Not Null

RequestOrderTable Creation Query:

```
 USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[RequestOrderTable](
	[productname] [varchar](50) NOT NULL,
	[strength] [varchar](50) NOT NULL,
	[quantity] [int] NOT NULL,
	[receivername] [varchar](50) NOT NULL,
	[receiverphone] [varchar](50) NOT NULL,
	[region] [varchar](50) NOT NULL,
	[city] [varchar](50) NOT NULL,
	[area] [varchar](50) NOT NULL,
	[address] [varchar](50) NOT NULL,
	[receiveremail] [varchar](50) NULL
) ON [PRIMARY]
```

Visual Represenation:

Column Name	**Data Type**	**Allow Nulls**
Productname	varchar (50)	Not Null
strength	varchar (50)	Not Null
quantity	int	Not Null
receivername	varchar (50)	Not Null
receiverphone	varchar (50)	Not Null
region	varchar (50)	Not Null
city	varchar (50)	Not Null
area	varchar (50)	Not Null
address	varchar (50)	Not Null
receiveremail	varchar (50)	Not Null

<u>ProductTable Creation Query:</u>

```
USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ProductTable](
	[pID] [varchar](50) NOT NULL,
	[pName] [varchar](50) NULL,
	[price] [float] NULL,
	[pStatus] [varchar](50) NULL,
	[username] [varchar](50) NULL,
 CONSTRAINT [PK_ProductTable] PRIMARY KEY CLUSTERED
(
	[pID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Visual Representation:

Column Name	Data Type	Allow Nulls
pID (Primary Key)	varchar (50)	Not Null
pName	varchar (50)	Not Null
Price	float	Not Null
pStatus	varchar (50)	Not Null
Username	varchar (50)	Not Null

<u>ReviewTable Creation Query:</u>

```
 USE [Pharmacy Management System]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ReviewsTable](
	[username] [varchar](50) NOT NULL,
	[reviews] [varchar](max) NULL,
 CONSTRAINT [PK_ReviewsTable] PRIMARY KEY CLUSTERED
(
	[username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
ALTER TABLE [dbo].[ReviewsTable]  WITH CHECK ADD
CONSTRAINT [Fk4] FOREIGN KEY([username])
REFERENCES [dbo].[SignUpTable] ([username])
GO
ALTER TABLE [dbo].[ReviewsTable] CHECK CONSTRAINT [Fk4]
```

Visual Representation

Column Name	Data Type	Allow Nulls
username (Primary Key)	varchar(50)	Not Null
reviews	varchar (MAX)	Not Null

Implementation

• Programming Language & Tools

C# (.NET): The C# language is the most popular language for the .NET platform, a free, cross-platform, open source development environment. C# programs can run on many different devices, from Internet of Things (IoT) devices to the cloud and everywhere in between. You can write apps for phone, desktop, and laptop computers and servers.
C# is a cross-platform general purpose language that makes developers productive while writing highly performant code. With millions of developers, C# is the most popular .NET language. C# has broad support in the ecosystem and all .NET workloads. Based on object-oriented principles, it incorporates many features from other paradigms, not least functional programming. Low-level features support high-efficiency scenarios without writing unsafe code. Most of the .NET runtime and libraries are written in C#, and advances in C# often benefit all .NET developers.

SQL Server Management Studio : SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. Use SSMS to access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, Azure SQL Managed Instance, SQL Server on Azure VM, and Azure Synapse Analytics. SSMS provides a single comprehensive utility that combines a broad group of graphical tools with many rich script editors to provide access to SQL Server for developers and database administrators of all skill levels.

- ## Major Code Functionality

Discussing how different parts of code works

SessionManager.CS Class:
The `sessionManager` class is a static utility class used to manage user session information in the Pharmacy Management System (PMS). It provides a way to store and access session-related data throughout the application.

Properties:

1) IsLoggedIn (bool): Indicates whether the user is currently logged in. This helps in controlling access to different parts of the application based on the user's login status.

2) Username (string): Stores the username of the currently logged-in user. This can be used to display the user's name or perform user-specific operations.

3) productID (string): Holds the ID of the product currently being interacted with. This can be useful for operations related to that specific product, such as viewing or editing product details.

Usage:

- Checking Login Status: You can check if a user is logged in by accessing `sessionManager.IsLoggedIn`. If it's `true`, the user is logged in; otherwise, they are not.

- Storing User Information: After a user logs in, you can store their username in `sessionManager.Username` for use throughout the application.

- Managing Product Information: When working with a specific product, you can set `sessionManager.productID` to the ID of the product you're interested in. This allows you to perform operations related to that product.

Functions.CS class:
This code is a C# class designed to help manage database operations, particularly for a system called **Pharmacy Management System (PMS).** It uses **SQL Server** to interact with the database. The main goal of the class is to perform two types of operations:

1. **Retrieve data** from the database (e.g., get information from tables).

2. **Modify data** in the database (e.g., add, update, or delete information).

Let's break it down:

1. Class and Connections

Namespace `PMS`: It's a way to organize code, like a folder for this class.
Class `Functions`: Contains methods (functions) to interact with the database.
Private variables:
 - `con`: Represents a connection to the SQL database.
 - `cmd`: Holds SQL commands to be executed.
 - `sda`: Used to retrieve data.
 - `dt`: A table structure used to store data after it's retrieved.
 - `conStr`: The connection string that tells the program how to connect to the database, including the server name, database name, and login credentials.

2. Constructor

- The **`Functions()`** constructor sets up the initial connection to the SQL Server using a connection string (`conStr`). This includes:
 - **Server name:** `ZOBAER`.
 - **Database:** "Pharmacy Management System".
 - **Login credentials:** User ID (`sa`) and Password (`admin`).
 - **Security settings:** It ensures encryption is used and trusts the server certificate for secure communication.

3. Get Data Method

csharp
public DataTable GetData(string query, Dictionary<string, object> parameters = null)

This method **retrieves data** from the database based on a SQL query you provide. The result is stored in a **DataTable**, which is a table-like structure in C#.

How it works:
 1. It creates a new SQL connection (`con`) and command (`cmd`) based on the query provided.
 2. If the query requires parameters (e.g., filtering by certain conditions), they are added to the SQL command.
 3. **SqlDataAdapter** (`sda`) is used to execute the query and fill the `DataTable` with the data retrieved from the database.
 4. If something goes wrong (like an error in the query), it logs the error and returns an empty `DataTable`.

4. Set Data Method

csharp

public int setData(string query, Dictionary<string, object> parameters = null)

This method is **for executing SQL commands that modify data** (such as INSERT, UPDATE, or DELETE).

How it works:
1. If the connection is closed, it opens it.
2. It clears any previous parameters, adds the current ones (if provided), and runs the SQL command.
3. **ExecuteNonQuery** is used because we're not fetching data but changing it.
4. After execution, it closes the connection and returns the number of rows affected by the query.
5. If there's an error, it catches the exception and prints an error message.

Key Features:
Parameters: Both methods support SQL queries with parameters. This is important to avoid **SQL Injection attacks** (a security risk).

Error Handling: The program prints errors to the console, which helps in debugging.
- **Connection Management**: The class makes sure that the connection to the database is properly opened and closed to avoid issues

Example Usage:

Get data:

csharp

```
 DataTable dt = GetData("SELECT * FROM Medicines");
```

This would get all the data from the `Medicines` table.

Insert data:

csharp

```
 Dictionary<string, object> parameters = new Dictionary<string, object>()
 {
     { "@Name", "Aspirin" },
     { "@Price", 50 }
 };
 int rowsAffected = setData("INSERT INTO Medicines (Name, Price) VALUES
(@Name, @Price)", parameters);
```

This would insert a new record with the name "Aspirin" and price 50 into the `Medicines` table.

This class helps simplify database interaction by abstracting away some of the repetitive tasks, making it easier to work with the database in your program.

This class helps simplify database interaction by abstracting away some of the repetitive tasks, making it easier to work with the database in your program.

- **Challenges**
  One of the significant challenges I faced during the development of the Pharmacy Management System was working with the DataGridView control, particularly in designing and managing its functionality. I encountered issues when attempting to delete specific rows from the DataGridView, as the functionality did not work as expected. Additionally, when trying to delete employee and department records, I faced further difficulties due to the foreign key relationships between the tables. The Employee table was linked to the Department and Salary tables, which restricted the deletion of records due to referential integrity constraints. Resolving these database relationship issues required a deeper understanding of foreign key dependencies and how to manage cascading deletions effectively. Another challenge occurred while implementing the "Add to Cart" functionality. Although I had written the required code, the functionality was not working. Upon reviewing the code, I discovered that the issue was due to a case mismatch in the SQL query—while I had correctly written @pID in the query, I had mistakenly written @pid in the parameter, causing the logic to fail. Correcting this small but critical error resolved the issue.

Testing And Validation

Test Cases

To ensure the robustness of the Pharmacy Management System, we implemented several test cases across different components:

Login Tests:

- **Valid User Login:** Test with valid credentials for both Admin and User roles to confirm correct redirection to their respective dashboards.
- **Invalid User Login:** Enter invalid credentials (incorrect username, email, or password) to verify the system displays appropriate error messages.
- **Password Reset:** Test the password reset functionality by entering registered and non-registered emails.

Product Addition Tests:

- **Add Product to Cart:** Test the functionality of adding a product to the cart and checking if the product's status changes to "cart" in the database.
- **Product Availability:** Verify that only products with available stock can be added to the cart, and an appropriate message is displayed if a product is out of stock.
- **Product Removal from Cart:** Ensure that products can be successfully removed from the cart before checkout.

Cart Checkout Tests:

- **Valid Checkout:** Test the checkout process with different payment methods (bKash, Nagad, Rocket, Banking) to ensure successful order placement and correct payment processing.
- **Invalid Payment:** Attempt checkout with incomplete or invalid payment details to ensure the system prevents the order and displays a proper error message.
- **Order Status Update:** Confirm that upon successful payment, the product status changes from "cart" to "Order placed."

Test Results

The system was tested thoroughly both manually and through user feedback. In manual testing, the following components were validated:

- **Login Module:** Functioned correctly with proper error handling for incorrect inputs. Role-based redirection worked as expected.
- **Product Management:** Successfully added products, updated statuses in the ProductTable, and handled various edge cases such as attempting to add out-of-stock items.
- **Cart and Checkout:** The checkout process was smooth, with accurate order updates. Different payment methods were verified and worked as intended.

User testing was conducted to ensure the user experience was intuitive, and user feedback confirmed ease of use across all major features.

Bug Fixes

During testing, a few bugs were identified:

- **Bug in Login Redirection:** Admin and User dashboards were sometimes switched due to incorrect role checking logic. This was fixed by refining the role verification code in the session manager.

- **Product Removal Failure:** Initially, removing a product from the cart did not update the product's status correctly in the database. The SQL query was modified to resolve this issue.
- **Order Status Update Delay:** There was a delay in updating the product status after checkout. This was fixed by optimizing the database transaction process.

Result And Discussion

Outcome

The Pharmacy Management System was successfully implemented with the following key features:

- **Product Management:** Admins can efficiently manage product details, including adding, updating, and deleting products.
- **User and Admin Login:** Role-based authentication ensures that users and admins access their respective dashboards.
- **Cart and Checkout:** Users can add products to the cart, choose from multiple payment methods, and complete their orders, which are reflected in the system.
- **Salary Calculation:** The system calculates employee salaries accurately based on attendance and commissions, providing automated salary slips.

Benefits

The system offers several benefits for real-world pharmacy operations:

- **Automation:** Automates daily tasks like product management, salary calculation, and order processing, reducing manual errors and saving time.
- **Efficiency:** Streamlines pharmacy management by integrating inventory management, user accounts, and payment systems in a single platform.
- **Accuracy:** By automating salary calculations and order management, the system ensures high accuracy in record-keeping, minimizing the risk of human error.

Limitations

Some limitations were faced during the development process:

- **Limited Reports:** Due to time constraints, more advanced reporting features (e.g., monthly sales reports, profit analysis) were not implemented.

- **UI Enhancements:** Further improvements could be made to the user interface, particularly in the admin dashboard, to improve the user experience.
- **Scalability:** The current system is designed for small to mid-sized pharmacies. For larger pharmacy chains, database optimization and scaling strategies may need to be considered.

Future Work

Several areas of the system can be expanded or improved in future iterations:

- **Advanced Analytics:** Adding features for sales trend analysis, product demand forecasting, and profit reports.
- **Inventory Alerts:** Implementing automated low-stock alerts and reorder points to improve inventory management.
- **Mobile Integration:** Developing a mobile app version for customers and pharmacy staff to provide remote access to the system.
- **Multi-Language Support:** Adding support for multiple languages to cater to pharmacies in different regions.

Conclusion

In conclusion, the pharmacy management system successfully achieved its goal of automating key aspects of a pharmacy's daily operations. The system contributes to solving the inefficiencies of manual stock management, billing, and user authentication by streamlining these processes into an integrated, automated system.

The project offers significant value in terms of improving accuracy and reducing the workload of pharmacy staff. It also ensures that sales are processed more efficiently, which can lead to higher customer satisfaction. While some advanced features like reporting and mobile access remain to be developed, the current system forms a strong foundation for future enhancements. Ultimately, this project has demonstrated the potential of technology to improve operational efficiency and accuracy in the healthcare sector, particularly in pharmacy management.

References

Microsoft .NET Documentation: This was a crucial resource for understanding the development of the system using C# and the .NET framework. It provided official guidance on the tools and libraries used in the project.

- Microsoft Docs. ".NET Documentation." Available at: https://docs.microsoft.com/en-us/dotnet/.

SQL Server Documentation: This resource provided detailed information on SQL Server, helping with database design, query optimization, and management for the system.

- Microsoft Docs. "SQL Server Documentation." Available at: https://docs.microsoft.com/en-us/sql/.

Windows Forms Documentation: The official Windows Forms documentation was used extensively for creating the user interface of the system.

- Microsoft Docs. "Windows Forms Overview." Available at: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/.

Design Patterns: The principles of object-oriented design were applied in the architecture of the system, with particular reference to design patterns.

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Database Design: This reference was used to ensure the proper design of relational databases, which were integral to the product and inventory management system.

- Hernandez, M. J. (2013). *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. Addison-Wesley.

Pharmacy Management Systems in Practice: This article provided insight into real-world applications of pharmacy management systems, which informed the design and features of my project.

- Zaman, M. T., Ahmed, R., & Hossain, M. (2021). *"Pharmacy Management Systems: Automating and Optimizing Health Services in Pharmacies."* International Journal of Pharmacy Practice, 29(3), pp. 184-191.

E-Commerce in Pharmacy: This paper helped in designing the cart and checkout system, demonstrating how similar features have improved efficiency in pharmacy operations.

- Rashid, A., & Bin Saleem, M. (2020). *"Impact of E-Commerce on Pharmacy Sales and Inventory Management."* Journal of Health Informatics, 12(1), pp. 35-42.

Software Testing Techniques: This reference was used to guide the testing process of the system, particularly in unit and integration testing.

- Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons.

Agile Testing and Scrum: The testing and iterative development process followed in the project was aligned with Agile principles, with this book guiding test case creation and validation.

- Cohn, M. (2010). *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley.

User Interface Design: The user interface was designed following best practices in UI/UX design, ensuring the system was easy to use and accessible to non-technical users in a pharmacy setting.

- Johnson, J. (2020). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Morgan Kaufmann.

Real-world Pharmacy Case Studies: This report provided real-world context for pharmacy management systems, helping to design features that align with industry standards.

- Smith, K., & Jones, R. (2019). *"Implementing Pharmacy Management Systems: Challenges and Benefits."* Healthcare IT News, 18(2), pp. 25-29.

HIPAA Compliance: While not directly handling patient data, HIPAA guidelines were reviewed to ensure the system was designed with future healthcare integration in mind.

- US Department of Health and Human Services. *"Health Insurance Portability and Accountability Act (HIPAA)."* Available at: https://www.hhs.gov/hipaa/index.html.

Appendices

- **Appendix A:**
  - **User Sign Up Functionality:**
    This code handles user sign up by inserting the informations into the database.

```
using System;
using System.Configuration;
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.Text.RegularExpressions;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Data.SqlClient;

namespace PMS
{
    public partial class signUp : Form
    {
        public signUp()
        {
            InitializeComponent();
        }
        string mailFormat = "^([0-9a-zA-Z]([-\\.\\w]*[0-9a-zA-Z])*@([0-9a-zA-
Z][-\\w]*[0-9a-zA-Z]\\.)+[a-zA-Z]{2,9})$";
        private void button2_Click(object sender, EventArgs e)
        {
            OpenFileDialog openfd = new OpenFileDialog();
            openfd.Filter = "Image Files| *.jpg;*.jpeg;*.png;*.svg;*.bmp|All
Files|*.*";

            if(openfd.ShowDialog() == DialogResult.OK)
            {
                string selectedFilePath = openfd.FileName;
                PictureBox pb = new PictureBox();
                pb.Image = Image.FromFile(selectedFilePath);
            }

        }
        string conStr = "Data Source=ZOBAER;Initial Catalog=\"Pharmacy
Management System\";User
ID=sa;Password=admin;TrustServerCertificate=True";
        private void button3_Click(object sender, EventArgs e)
        {
            //First Name
            if (string.IsNullOrEmpty(txtFname.Text) == true)
            {
                txtFname.Focus();
                errorProvider1.SetError(this.txtFname, "Please fill First Name");
            }
            //Last Name
            else if (string.IsNullOrEmpty(txtLname.Text) == true)
            {
                txtLname.Focus();
                errorProvider2.SetError(this.txtLname, "Please fill Last Name");
            }
            //Username
            else if (string.IsNullOrEmpty(txtUsername.Text) == true)
            {
                txtLname.Focus();
                errorProvider3.SetError(this.txtUsername, "Please fill Username");
            }
            //Email Validation

            else if (Regex.IsMatch(txtEmail.Text, mailFormat) == false)
            {
                txtEmail.Focus();
                errorProvider4.SetError(this.txtEmail, "Invalid email");
            }
            //Contact No
            int ContactNo;
            if(!int.TryParse(txtConNo.Text, out ContactNo))
            {
                txtConNo.Focus();
                errorProvider5.SetError(this.txtConNo, "Contact number must be
numeric.");
            }
            //Set Password
            else if (string.IsNullOrEmpty(txtNPass.Text) == true)
            {
                txtNPass.Focus();
                errorProvider6.SetError(this.txtNPass, "Please fill Set Password");
            }
            //Confirm Password
            else if (string.IsNullOrEmpty(txtCPass.Text) == true)
            {
                txtCPass.Focus();
                errorProvider7.SetError(this.txtCPass, "Please fill Confirm
Password");
            }
            //Setting the confirm password = Set password
            else if (txtCPass.Text != txtNPass.Text)
            {
                errorProvider9.SetError(this.txtCPass, "Retype the password
again");
            }
            //Adress
            else if (string.IsNullOrEmpty(txtAddress.Text) == true)
            {
                txtAddress.Focus();
                errorProvider8.SetError(this.txtAddress, "Please fill Address");
            }
            //Dob
            else if (DoBpicker.Checked == false)
            {
                DoBpicker.Focus();
                errorProvider10.SetError(this.DoBpicker, "please check that you
provide the correct Birthday");
            }
            //Gender selection logic
            string SelectedGender = string.Empty;
            if(radioButton1.Checked && radioButton1.Enabled)
            {
                SelectedGender = radioButton1.Text;
            }
            else if(radioButton2.Checked && radioButton2.Enabled)
            {
                SelectedGender = radioButton2.Text;
            }
            else if(radioButton3.Checked && radioButton3.Enabled)
            {
                SelectedGender = radioButton3.Text;
            }
            if(string.IsNullOrEmpty(SelectedGender))
            {
                errorProvider9.SetError(this.lblGender, "Please select a gender");
            }
            else
            {
                //Clearing error providers
                errorProvider1.Clear();
                errorProvider2.Clear();
                errorProvider3.Clear();
                errorProvider4.Clear();
                errorProvider5.Clear();
                errorProvider6.Clear();
                errorProvider7.Clear();
                errorProvider9.Clear();
                errorProvider10.Clear();



                SqlConnection con = new SqlConnection(conStr);
                con.Open();
                SqlCommand cmd = new SqlCommand("INSERT INTO
dbo.SignUpTable(firstname, lastname, username, email, contactno, pass,
dateofbirth, gender, address)VALUES(@firstname, @lastname, @username,
@email, @contactno, @pass, @dateofbirth, @gender, @address)", con);
```

```
                cmd.Parameters.AddWithValue("@firstname", txtFname.Text);
                cmd.Parameters.AddWithValue("@lastname", txtLname.Text);
                cmd.Parameters.AddWithValue("@username", txtUsername.Text);
                cmd.Parameters.AddWithValue("@email", txtEmail.Text);
                cmd.Parameters.AddWithValue("@contactno", ContactNo);
                cmd.Parameters.AddWithValue("@pass", txtCPass.Text);
                DateTime DoB = DoBpicker.Value.Date;
                cmd.Parameters.AddWithValue("@dateofbirth", DoB);
                cmd.Parameters.AddWithValue("@gender", SelectedGender);
                cmd.Parameters.AddWithValue("@address", txtAddress.Text);
                cmd.ExecuteNonQuery();
                con.Close();

                MessageBox.Show("Sign Up Successful","Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);


            }
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            txtNPass.UseSystemPasswordChar = !checkBox1.Checked;
            txtCPass.UseSystemPasswordChar= !checkBox1.Checked;
        }

        private void label1_Click(object sender, EventArgs e)
        {
            Home home = new Home();
            this.Hide();
            home.ShowDialog();
            this.Show();

        }

        private void label1_Click_1(object sender, EventArgs e)
        {
            panel1.Visible = !panel1.Visible;
        }

        //Esc btn event
        protected override bool ProcessCmdKey(ref Message msg, Keys
keyData)
        {
```

```
    if(keyData == Keys.Escape)
    {
        if(Home.stack.Count>0)
        {
            Form previousForm = Home.stack.Pop();
            this.Hide();
            previousForm.Show();
        }
        return true;
    }
    return base.ProcessCmdKey(ref msg, keyData);
}


private void label2_Click(object sender, EventArgs e)
{
    Contacts cn = new Contacts();
    Home.stack.Push(this);
    this.Hide();
    cn.ShowDialog();
    this.Show();
}

private void label18_Click(object sender, EventArgs e)
{
    Home frm = new Home();
    Home.stack.Push(this);
    this.Hide();
    frm.ShowDialog();
    this.Show();
}

private void button13_Click(object sender, EventArgs e)
{
    prescriptionMedicine pm = new prescriptionMedicine();
    Home.stack.Push(this);
    this.Hide();
    pm.ShowDialog();
    this.Show();
}


private void label17_Click(object sender, EventArgs e)
{
    requestOrder ro = new requestOrder();
    Home.stack.Push(this);
    this.Hide();
    ro.ShowDialog();
    this.Show();
}

private void label16_Click(object sender, EventArgs e)
{
    offer of = new offer();
    Home.stack.Push(this);
    this.Hide();
    of.ShowDialog();
    this.Show();
}

private void label15_Click(object sender, EventArgs e)
{
    location lc = new location();
    Home.stack.Push(this);
    this.Hide();
    lc.ShowDialog();
    this.Show();
}

//First Name
private void txtFname_Leave(object sender, EventArgs e)
{
    if(string.IsNullOrEmpty(txtFname.Text) == true)
    {
        txtFname.Focus();
        errorProvider1.SetError(this.txtFname, "Please fill First Name");
    }
    else
    {
        errorProvider1.Clear();
    }
}


private void pictureBox2_Click(object sender, EventArgs e)
{
    if (Home.stack.Count > 0)
    {
        Form previousForm = Home.stack.Pop();
        this.Hide();
        previousForm.Show();
    }
}


private void label18_MouseEnter(object sender, EventArgs e)
{
    label18.ForeColor = Color.OrangeRed;
}

private void label18_MouseLeave(object sender, EventArgs e)
{
    label18.ForeColor = Color.Black;
}

private void label17_MouseEnter(object sender, EventArgs e)
{
    label17.ForeColor = Color.OrangeRed;
}

private void label17_MouseLeave(object sender, EventArgs e)
{
    label17.ForeColor = Color.Black;
}

private void label16_MouseEnter(object sender, EventArgs e)
{
    label16.ForeColor = Color.OrangeRed;
}

private void label16_MouseLeave(object sender, EventArgs e)
{
    label16.ForeColor= Color.Black;
}

private void label15_MouseEnter(object sender, EventArgs e)
{
    label15.ForeColor = Color.OrangeRed;
}

private void label15_MouseLeave(object sender, EventArgs e)
{
```

```
    label15.ForeColor = Color.Black;
}

private void label3_MouseEnter(object sender, EventArgs e)
{
    label3.ForeColor = Color.OrangeRed;
}

private void label3_MouseLeave(object sender, EventArgs e)
{
    label3.ForeColor = Color.Black;
}

private void label2_MouseEnter(object sender, EventArgs e)
{
    label2.ForeColor = Color.OrangeRed;
}

private void label2_MouseLeave(object sender, EventArgs e)
{
    label2.ForeColor = Color.Black;
}

private void label1_MouseEnter(object sender, EventArgs e)
{
    label1.ForeColor = Color.OrangeRed;
}

private void label1_MouseLeave(object sender, EventArgs e)
{
    label1.ForeColor = Color.Black;
}
private void label26_Click_1(object sender, EventArgs e)
{
    if (Home.stack.Count > 0)
    {
        Form previousForm = Home.stack.Pop();
        this.Hide();
        previousForm.Show();
    }
}

private void button1_Click(object sender, EventArgs e)
{
```

```
            if (sessionManager.IsLoggedIn)
            {
                Cart cart = new Cart();
                Home.stack.Push(this);
                this.Hide();
                cart.ShowDialog();
            }
            else
            {
                MessageBox.Show("Please login or Sign up", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            surgicalProduct su = new surgicalProduct();
            Home.stack.Push(this);
            this.Hide();
            su.ShowDialog();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            otcMedicine otcMedicine = new otcMedicine();
            Home.stack.Push(this);
            this.Hide();
            otcMedicine.ShowDialog();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            babyCare babyCare = new babyCare();
            Home.stack.Push(this);
            this.Hide();
            babyCare.ShowDialog();
        }

        private void label3_Click(object sender, EventArgs e)
        {
            About about = new About();
            Home.stack.Push(this);
            this.Hide();
            about.ShowDialog();
        }
    }
}
```

- **User Login Functionality**
  This code handles user login by validating the provided credentials against the database.

```
using System;
using System.Configuration;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Linq;
using System.Net.Mail;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PMS
{
    public partial class Home : Form
    {
        public static Stack<Form> stack = new Stack<Form>();
        Functions con;
        public Home()
        {
            InitializeComponent();
            con = new Functions();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
        }

        private void label1_Click(object sender, EventArgs e)
        {
            Home frm = new Home();
            this.Hide();
            frm.ShowDialog();
        }
        string conStr = "Data Source=ZOBAER;Initial Catalog=\"Pharmacy
Management System\";User
ID=sa;Password=admin;TrustServerCertificate=True";
        private void button3_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(txtUsername.Text) == true)
            {
                txtUsername.Focus();
                errorProvider1.SetError(this.txtUsername, "Please fill Username");
            }
            else if (string.IsNullOrEmpty(txtPass.Text) == true)
            {
                txtPass.Focus();
                errorProvider2.SetError(this.txtPass, "Please fill Password");
            }
            // Attempt to login as Admin
            using (SqlConnection con = new SqlConnection(conStr))
            {
                con.Open();
                SqlCommand adminCmd = new SqlCommand("SELECT name,
email, contactno, dateofbirth, gender, address FROM AdminTable WHERE
username=@username AND pass=@pass", con);
                adminCmd.Parameters.AddWithValue("@username",
txtUsername.Text);
                adminCmd.Parameters.AddWithValue("@pass", txtPass.Text);

                SqlDataReader adminReader = adminCmd.ExecuteReader();
                if (adminReader.Read())
                {
                    // Admin Login Success
                    string name = adminReader["name"].ToString();
                    string email = adminReader["email"].ToString();
                    string contactno = adminReader["contactno"].ToString();
                    string dateofbirth = adminReader["dateofbirth"].ToString();
                    string gender = adminReader["gender"].ToString();
                    string address = adminReader["address"].ToString();
                    MessageBox.Show($"Welcome back {name}", "Debug",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    //Login true
                    sessionManager.IsLoggedIn = true;
                adminDashboard adminDashboard = new adminDashboard(name,
email, contactno, dateofbirth, gender, address);
                Home.stack.Push(this);
                this.Hide();
                adminDashboard.ShowDialog();
                return; // Return early if admin login is successful
            }
            adminReader.Close(); // Close reader before reusing connection
        }

        // Attempt to login as User
        using (SqlConnection con = new SqlConnection(conStr))
        {
            con.Open();
            SqlCommand userCmd = new SqlCommand("SELECT firstname,
lastname, email, contactno, dateofbirth, gender, address FROM SignUpTable
WHERE username=@username AND pass=@pass", con);
            userCmd.Parameters.AddWithValue("@username",
txtUsername.Text);
            userCmd.Parameters.AddWithValue("@pass", txtPass.Text);

            SqlDataReader userReader = userCmd.ExecuteReader();
            if (userReader.Read())
            {
                // User Login Success
                string firstname = userReader["firstname"].ToString();
                string lastname = userReader["lastname"].ToString();
                string email = userReader["email"].ToString();
                string contactno = userReader["contactno"].ToString();
                string dateofbirth = userReader["dateofbirth"].ToString();
                string gender = userReader["gender"].ToString();
                string address = userReader["address"].ToString();
                MessageBox.Show($"Welcome {firstname} {lastname}",
"Welcome", MessageBoxButtons.OK, MessageBoxIcon.Information);

                sessionManager.IsLoggedIn = true;
                sessionManager.Username = txtUsername.Text;

                userDashboard userDashboard = new userDashboard(firstname,
lastname, email, contactno, dateofbirth, gender, address);
                Home.stack.Push(this);
                this.Hide();
                userDashboard.ShowDialog();
                return;
            }
            else
            {
                MessageBox.Show("Invalid Username or Password", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

    }



    private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
        signUp su = new signUp();
        Home.stack.Push(this);
        this.Hide();
        su.ShowDialog();
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        if(checkBox1.Checked)
        {
            txtPass.UseSystemPasswordChar = false;
        }
        else
        {
            txtPass.UseSystemPasswordChar=true;
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        prescriptionMedicine pm = new prescriptionMedicine();
        Home.stack.Push(this);
        this.Hide();
        pm.ShowDialog();
    }


    private void label1_Click_2(object sender, EventArgs e)
    {
        panel1.Visible = !panel1.Visible;
    }



    private void label13_Click(object sender, EventArgs e)
    {
        Contacts cn = new Contacts();
        Home.stack.Push(this);
        this.Hide();
        cn.ShowDialog();
    }

    private void label17_Click(object sender, EventArgs e)
    {
        requestOrder ro = new requestOrder();
        Home.stack.Push(this);
        this.Hide();
        ro.ShowDialog();
    }

    private void linkLabel2_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
        forgotPass fp = new forgotPass();
        this.Hide();
        Home.stack.Push(this);
        fp.ShowDialog();

    }

    private void label16_Click(object sender, EventArgs e)
    {
        offer of = new offer();
        Home.stack.Push(this);
        this.Hide();
        of.ShowDialog();

    }

    private void label15_Click(object sender, EventArgs e)
    {
        location lc = new location();
        Home.stack.Push(this);
```

```
    this.Hide();
    lc.ShowDialog();
}

private void label18_MouseHover(object sender, EventArgs e)
{
    label18.ForeColor = Color.Orange;
}

private void label18_MouseEnter(object sender, EventArgs e)
{
    label18.ForeColor = Color.Orange;
}

private void label18_MouseLeave(object sender, EventArgs e)
{
    label18.ForeColor = Color.Black;
}


private void label18_MouseEnter_1(object sender, EventArgs e)
{
    label18.ForeColor = Color.OrangeRed;
}

private void label18_MouseLeave_1(object sender, EventArgs e)
{
    label18.ForeColor = Color.Black;
}

private void label17_MouseEnter(object sender, EventArgs e)
{
    label17.ForeColor = Color.OrangeRed;
}

private void label17_MouseLeave(object sender, EventArgs e)
{
    label17.ForeColor= Color.Black;
}

private void label16_MouseEnter(object sender, EventArgs e)
{
    label16.ForeColor = Color.OrangeRed;
}
```

```
private void label16_MouseLeave(object sender, EventArgs e)
{
   label16.ForeColor= Color.Black;
}

private void label15_MouseEnter(object sender, EventArgs e)
{
   label15.ForeColor = Color.OrangeRed;
}

private void label15_MouseLeave(object sender, EventArgs e)
{
   label15.ForeColor= Color.Black;
}

private void label14_MouseEnter(object sender, EventArgs e)
{
   label14.ForeColor = Color.OrangeRed;
}

private void label14_MouseLeave(object sender, EventArgs e)
{
   label14.ForeColor= Color.Black;
}

private void label13_MouseEnter(object sender, EventArgs e)
{
   label13.ForeColor = Color.OrangeRed;
}

private void label13_MouseLeave(object sender, EventArgs e)
{
   label13.ForeColor= Color.Black;
}

private void label1_MouseEnter(object sender, EventArgs e)
{
   label1.ForeColor = Color.OrangeRed;
}

private void label1_MouseLeave(object sender, EventArgs e)
{
   label1.ForeColor= Color.Black;
```

```
}

private void label1_MouseHover(object sender, EventArgs e)
{

}

private void panel1_MouseLeave(object sender, EventArgs e)
{
   panel1.Visible = false;
}

private void button4_Click(object sender, EventArgs e)
{
   surgicalProduct surgicalProduct = new surgicalProduct();
   Home.stack.Push(this);
   this.Hide();
   surgicalProduct.ShowDialog();
}

private void button6_Click(object sender, EventArgs e)
{
   babyCare babycare = new babyCare();
   Home.stack.Push(this);
   this.Hide();
   babycare.ShowDialog();
}

private void button5_Click(object sender, EventArgs e)
{
   otcMedicine otcMedicine = new otcMedicine();
   Home.stack.Push(this);
   this.Hide();
   otcMedicine.ShowDialog();
}

private void cartBtn_Click(object sender, EventArgs e)
{
   if (sessionManager.IsLoggedIn)
   {
      Cart cart = new Cart();
      Home.stack.Push(this);
      this.Hide();
```

```
            cart.ShowDialog();
        }
        else
        {
            MessageBox.Show("Please login or Sign up", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void label14_Click(object sender, EventArgs e)
    {
        About about = new About();
        Home.stack.Push(this);
        this.Hide();
        about.ShowDialog();
    }
  }
}
```

- **Add To Cart Functionality**
  This code handles Add To Cart by inserting the provided credentials into the database.

```
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 static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Xml.Linq;
using System.Text.RegularExpressions;

namespace PMS
{
  public partial class Cart : Form
  {
    //FUnctions from Functions class where I define the database
    Functions con;
```

```
        int selectedPID = 0;
        public Cart()
        {
            InitializeComponent();
            ConfigureDataGridView();
            con = new Functions();
            showData();
        }

        string currentUsername = sessionManager.Username;
        private void ConfigureDataGridView()
        {
            cartList.CellContentClick += new
DataGridViewCellEventHandler(cartList_CellContentClick);

            // Set alternating row colors for readability
            cartList.AlternatingRowsDefaultCellStyle.BackColor =
Color.LightGray;

            // Set header styles
            cartList.EnableHeadersVisualStyles = false;
            cartList.ColumnHeadersDefaultCellStyle.BackColor =
Color.FromArgb(255, 128, 0);
            cartList.ColumnHeadersDefaultCellStyle.ForeColor = Color.Black;
            cartList.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI",
10, FontStyle.Bold);

            // Set grid line color
            cartList.GridColor = Color.Black;

            // Set default row styles
            cartList.DefaultCellStyle.BackColor = Color.White;
            cartList.DefaultCellStyle.ForeColor = Color.Black;
            cartList.DefaultCellStyle.Font = new Font("Segoe UI", 10);

            // Set selection styles
            cartList.DefaultCellStyle.SelectionBackColor = Color.DarkOrange;
            cartList.DefaultCellStyle.SelectionForeColor = Color.White;

            // Fit columns to the DataGridView
            cartList.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;

            // Adjust row height to fit content
            cartList.AutoSizeRowsMode =
DataGridViewAutoSizeRowsMode.AllCells;

            // Disable column resize by the user for a consistent layout
            cartList.AllowUserToResizeColumns = false;

            // Set row and column headers visibility if needed
            cartList.RowHeadersVisible = false;
            cartList.ColumnHeadersVisible = true;
        }
        private void showData()
        {
            string Query = "SELECT * FROM ProductTable WHERE username =
@username";
            var parameters = new Dictionary<string, object>
            {
                {"@username", currentUsername }
            };
            cartList.DataSource = con.GetData(Query, parameters);
        }
        int key = 0;
        private void cartList_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0 && e.ColumnIndex == 0) // Assuming the delete
button is in the first column
            {
                // Get the selected product ID
                var selectedRow = cartList.Rows[e.RowIndex];
                string productID = selectedRow.Cells["pID"].Value.ToString(); //
Assuming column name is "pID"

                // Store and format the product ID
                if (IsValidProductID(productID))
                {
                    selectedPID = int.Parse(productID.Replace("p-", ""));
                }
                else
                {
                    MessageBox.Show("Invalid product ID format.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        private void SumPrices()
        {
            double sum = 0;
            foreach (DataGridViewRow row in cartList.Rows)
            {
                if (row.Cells[2].Value != null)
                {
                    double value;
                    if (double.TryParse(row.Cells[2].Value.ToString(), out value))
                    {
                        sum += value;
                    }
                }
            }

            MessageBox.Show($"Your total bill is: {sum} bdt",
"Success",MessageBoxButtons.OK, MessageBoxIcon.Information);
        }



        private void deleteBtn_Click(object sender, EventArgs e)
        {
            try
            {
                if (selectedPID == 0)
                {
                    MessageBox.Show("No item selected for deletion.", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                string productID = FormatProductID(selectedPID); // Format ID as
'p-XXX
                string Query = "DELETE FROM ProductTable WHERE pID =
@pID";
                var parameters = new Dictionary<string, object>
                {
                    { "@pID", productID }
                };

                con.setData(Query, parameters);
                showData();
                //Debugging
                MessageBox.Show("Item deleted successfully", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                selectedPID = 0; // Reset
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        private string FormatProductID(int id)
        {
            return $"p-{id:D3}"; // Format integer as 'p-XXX' with leading zeros if
necessary
        }

        private bool IsValidProductID(string pID)
        {
            string pattern = @"^p-\d{3}$"; // Pattern for 'p-XXX' where XXX is
exactly three digits
            Regex regex = new Regex(pattern);
            return regex.IsMatch(pID);
        }
        private void label26_Click(object sender, EventArgs e)
        {
            if (Home.stack.Count > 0)
            {
                Form previousForm = Home.stack.Pop();
                this.Hide();
                previousForm.Show();
            }
        }

        private void orderBtn_Click(object sender, EventArgs e)
        {
            SumPrices();
            payment payment = new payment();
            this.Hide();
            payment.ShowDialog();
        }

        private void categoriesBtn_Click(object sender, EventArgs e)
        {
            ddMenu.Visible = !ddMenu.Visible;
        }

        private void button15_Click(object sender, EventArgs e)
        {
            prescriptionMedicine pm = new prescriptionMedicine();
            Home.stack.Push(this);
            this.Hide();
            pm.ShowDialog();
        }

        private void button14_Click(object sender, EventArgs e)
        {
            surgicalProduct sp = new surgicalProduct();
            Home.stack.Push(this);
            this.Hide();
            sp.ShowDialog();
        }

        private void button13_Click(object sender, EventArgs e)
        {
            otcMedicine om = new otcMedicine();
            Home.stack.Push(this);
            this.Hide();
            om.ShowDialog();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            babyCare bc = new babyCare();
            Home.stack.Push(this);
            this.Hide();
            bc.ShowDialog();
        }

        private void aboutBtn_Click(object sender, EventArgs e)
        {
            About about = new About();
            Home.stack.Push(this);
            this.Hide();
            about.ShowDialog();
        }
    }
}
```

- **User Dashboard Functionality**

  This code handles User Dashboard by retrieving data from the provided credentials from the database.

```
using System;
using System.Collections;
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 static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace PMS
{

    public partial class userDashboard : Form
    {

        Functions con;
        private string firstname;
        private string lastname;
        private string email;
        private string contactno;
        private string dateofbirth;
        private string gender;
        private string address;

        public userDashboard(string fname, string lname, string email, string
contactno, string dateofbirth, string gender, string adress)
        {
            InitializeComponent();
            this.firstname = fname;
            this.lastname = lname;
            this.email = email;
            this.contactno = contactno;
            this.dateofbirth = dateofbirth;
            this.gender = gender;
            this.address = adress;
            con = new Functions();
        }




        private void label6_Click(object sender, EventArgs e)
        {

        }
        //Esc Button
        protected override bool ProcessCmdKey(ref Message msg, Keys
keyData)
        {
            if (keyData == Keys.Escape)
            {
                if (Home.stack.Count > 0)
                {
                    Form previousForm = Home.stack.Pop();
                    this.Hide();
                    previousForm.Show();
                }
                return true;
            }
            return base.ProcessCmdKey(ref msg, keyData);
        }
        private void label18_Click(object sender, EventArgs e)
        {
            Home frm = new Home();
            Home.stack.Push(this);
            this.Hide();
            frm.ShowDialog();
            this.Show();
        }

        private void label17_Click(object sender, EventArgs e)
        {
            requestOrder ro = new requestOrder();
            Home.stack.Push(this);
            this.Hide();
            ro.ShowDialog();
            this.Show();
        }
```

```
private void label16_Click(object sender, EventArgs e)
{
   offer of = new offer();
   Home.stack.Push(this);
   this.Hide();
   of.ShowDialog();
   this.Show();
}

private void label15_Click(object sender, EventArgs e)
{
   location lc = new location();
   Home.stack.Push(this);
   this.Hide();
   lc.ShowDialog();
   this.Show();
}

private void label13_Click(object sender, EventArgs e)
{
   Contacts cn = new Contacts();
   Home.stack.Push(this);
   this.Hide();
   cn.ShowDialog();
   this.Show();
}



private void label26_Click_1(object sender, EventArgs e)
{
   if (Home.stack.Count > 0)
   {
      Form previousForm = Home.stack.Pop();
      this.Hide();
      previousForm.Show();
   }
}


//MenuBtn Color setting start
private void label18_MouseEnter(object sender, EventArgs e)
{
    label18.ForeColor = Color.OrangeRed;
}

private void label17_MouseEnter(object sender, EventArgs e)
{
    label17.ForeColor = Color.OrangeRed;
}

private void label16_MouseEnter(object sender, EventArgs e)
{
    label16.ForeColor = Color.OrangeRed;
}

private void label15_MouseEnter(object sender, EventArgs e)
{
    label15.ForeColor = Color.OrangeRed;
}

private void label14_MouseEnter(object sender, EventArgs e)
{
    label14.ForeColor = Color.OrangeRed;
}

private void label13_MouseEnter(object sender, EventArgs e)
{
    label13.ForeColor = Color.OrangeRed;
}

private void label18_MouseLeave(object sender, EventArgs e)
{
    label18.ForeColor = Color.Black;
}

private void label17_MouseLeave(object sender, EventArgs e)
{
    label17.ForeColor = Color.Black;
}

private void label16_MouseLeave(object sender, EventArgs e)
{
    label16.ForeColor = Color.Black;
}

private void label15_MouseLeave(object sender, EventArgs e)
```

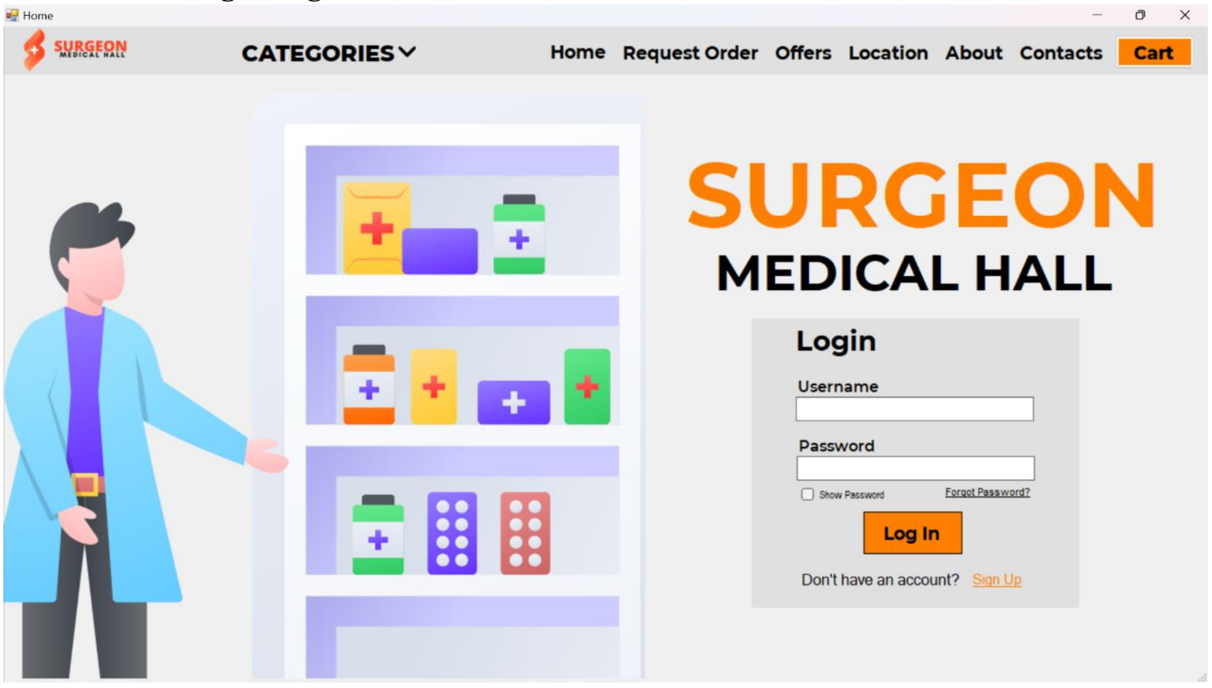
```
{
   label15.ForeColor = Color.Black;
}

private void label14_MouseLeave(object sender, EventArgs e)
{
   label14.ForeColor = Color.Black;
}

private void label13_MouseLeave(object sender, EventArgs e)
{
   label13.ForeColor = Color.Black;
}

private void button5_Click(object sender, EventArgs e)
{
   panel14.Visible=!panel14.Visible;
}


private void panel1_Paint(object sender, PaintEventArgs e)
{

}

private void panel4_Paint(object sender, PaintEventArgs e)
{

}

private void panel14_Paint(object sender, PaintEventArgs e)
{

}

private void panel5_Paint(object sender, PaintEventArgs e)
{

}

private void panel2_Paint(object sender, PaintEventArgs e)
{

}
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            sessionManager.IsLoggedIn = false;
            Home frm = new Home();
            this.Hide();
            frm.ShowDialog();
            this.Show();
        }

        private void userDashboard_Load(object sender, EventArgs e)
        {
            // Display all of the information according to database
            lbluserFullName.Text = $"{firstname} {lastname}";
            lblfullNameMain.Text = $"{firstname} {lastname}";
            lblPhoneNo.Text = contactno;
            lblphoneNumberMain.Text = contactno;
            lblEmailAdd.Text = email;
            lblemailMain.Text = email;
            lblGender.Text = gender;
            lblDoB.Text = dateofbirth;
            lblAdress.Text = address;
        }

        private void UpdatePassBtn_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text;
            string currentPassword = txtCurPass.Text;
            string newPassword = txtNewPass.Text;
            string confirmPassword = txtConPass.Text;

            if (validationInputs(username, currentPassword, newPassword,
confirmPassword))
            {
                if(updatePassword(username, currentPassword, newPassword))
                {
                    MessageBox.Show("Password changed successfully", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }
                else
                {
                    MessageBox.Show("Current Password is Incorrect !!!", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
                return;
            }
        }

    }
    private bool validationInputs(string username, string curPass, string
newPass, string conPass)
    {
        //String.IsNullOrWhiteSpace(value)
        if(username=="" || curPass =="" || newPass == "" || conPass == "")
        {
            MessageBox.Show("Please fill all fields", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            return false;
        }
        if(newPass != conPass)
        {
            MessageBox.Show("New Password and Confirm Password do not
match", "Failure", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return false;
        }
        return true;
    }

    private bool updatePassword(string username, string curPass, string
newPass)
    {
        string ValidationQuery = "SELECT pass FROM SignUpTable WHERE
username = @username";
        var validationParameters = new Dictionary<string, object>
        {
            {"username", username}
        };
        DataTable parametersResult =
con.GetData(ValidationQuery,validationParameters);

        //Update Password
        if (parametersResult.Rows.Count > 0 &&
parametersResult.Rows[0][0].ToString() == curPass)
        {
            string updatePassQuery = "UPDATE SignUpTable SET pass =
@newPass WHERE username = @username AND pass = @curPass";
            var updatePassParam = new Dictionary<string, object>
            {
                {"@username",username},
                {"@curPass", curPass},
                {"@newPass", newPass}
            };
            int rowsAffected = con.setData(updatePassQuery,updatePassParam);
            return rowsAffected > 0;
        }
        return false;
    }

    private void showPass_CheckedChanged(object sender, EventArgs e)
    {
        txtCurPass.UseSystemPasswordChar = !showPass.Checked;
        txtNewPass.UseSystemPasswordChar = !showPass.Checked;
        txtConPass.UseSystemPasswordChar = !showPass.Checked;
    }

    private void viewCartBtn_Click(object sender, EventArgs e)
    {
        if (sessionManager.IsLoggedIn)
        {
            Cart cart = new Cart();
            Home.stack.Push(this);
            this.Hide();
            cart.ShowDialog();
        }
        else
        {
            MessageBox.Show("Please login or Sign up", "Failure",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void categoriesBtn_Click(object sender, EventArgs e)
    {
        ddMenu.Visible = !ddMenu.Visible;
    }

    private void button15_Click(object sender, EventArgs e)
    {
        prescriptionMedicine pm = new prescriptionMedicine();
        Home.stack.Push(this);
        this.Hide();
        pm.ShowDialog();
        }

        private void button14_Click(object sender, EventArgs e)
        {
            surgicalProduct sp = new surgicalProduct();
            Home.stack.Push(this);
            this.Hide();
            sp.ShowDialog();
        }

        private void button13_Click(object sender, EventArgs e)
        {
            otcMedicine om = new otcMedicine();
            Home.stack.Push(this);
            this.Hide();
            om.ShowDialog();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            babyCare bc = new babyCare();
            Home.stack.Push(this);
            this.Hide();
            bc.ShowDialog();
        }

        private void label14_Click(object sender, EventArgs e)
        {
            About about = new About();
            Home.stack.Push(this);
            this.Hide();
            about.ShowDialog();
        }
    }
}
```

- ## Appendix B

Screenshot of login Page

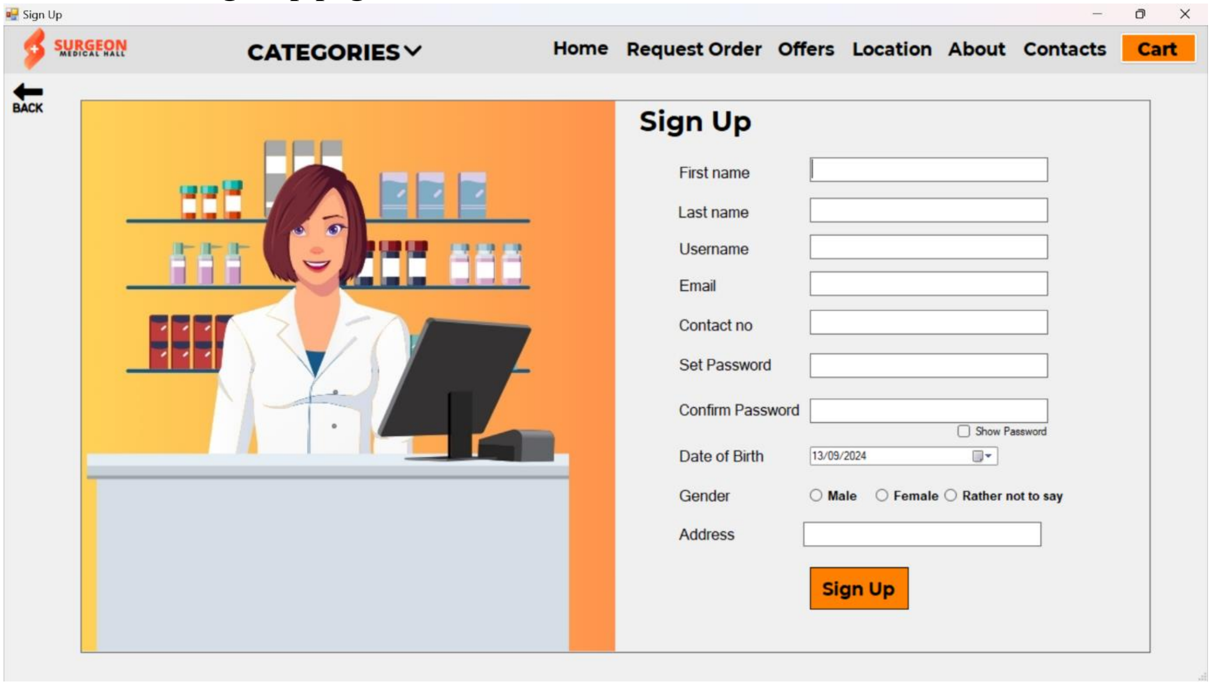


Login

Username	
Password	

Don't have an account? Sign Up

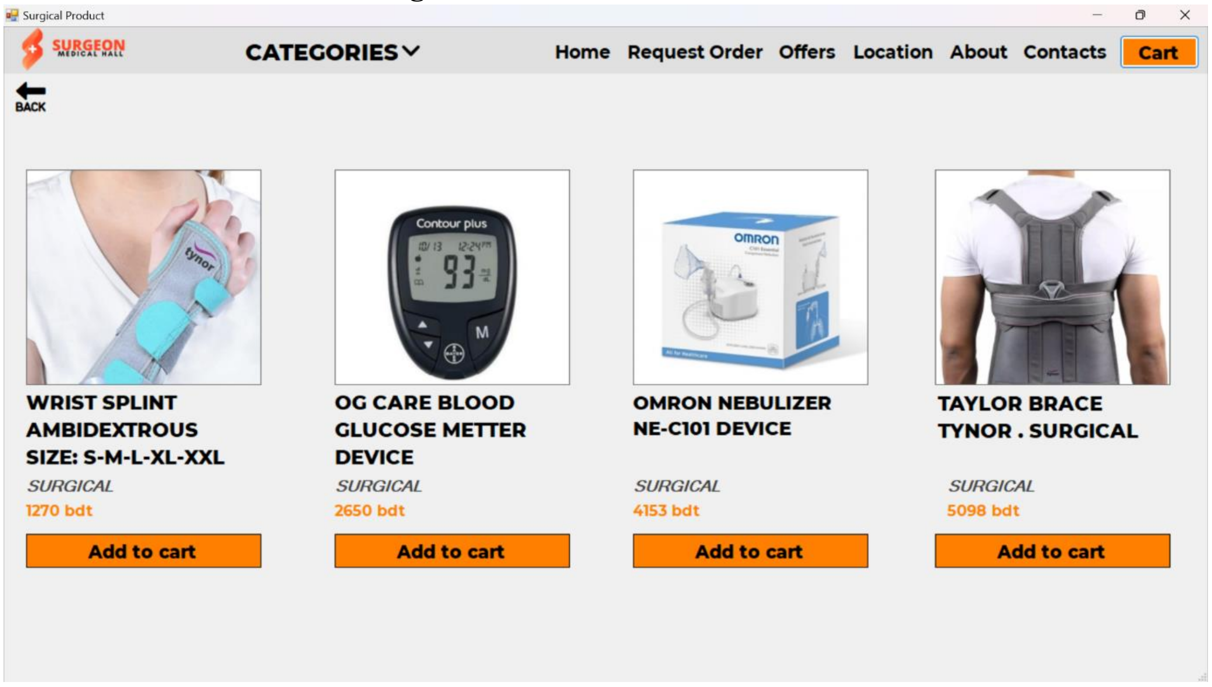
Screenshot of Sign-Up page



Sign Up

First name	
Last name	
Username	
Email	
Contact no	
Set Password	
Confirm Password	
Date of Birth	13/09/2024
Gender	Male / Female / Rather not to say
Address	

Screenshot of Add To Cart Page



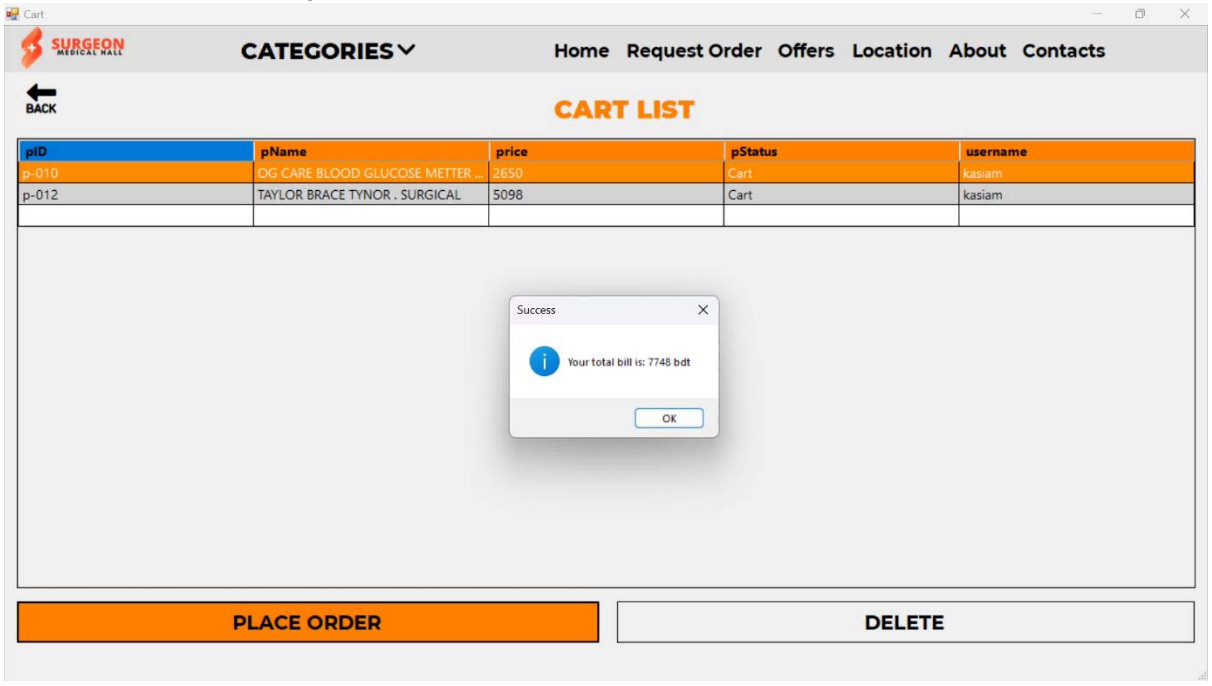
WRIST SPLINT AMBIDEXTROUS SIZE: S-M-L-XL-XXL
SURGICAL
1270 bdt

OG CARE BLOOD GLUCOSE METTER DEVICE
SURGICAL
2650 bdt

OMRON NEBULIZER NE-C101 DEVICE
SURGICAL
4153 bdt

TAYLOR BRACE TYNOR . SURGICAL
SURGICAL
5098 bdt

Screenshot of Cart Page



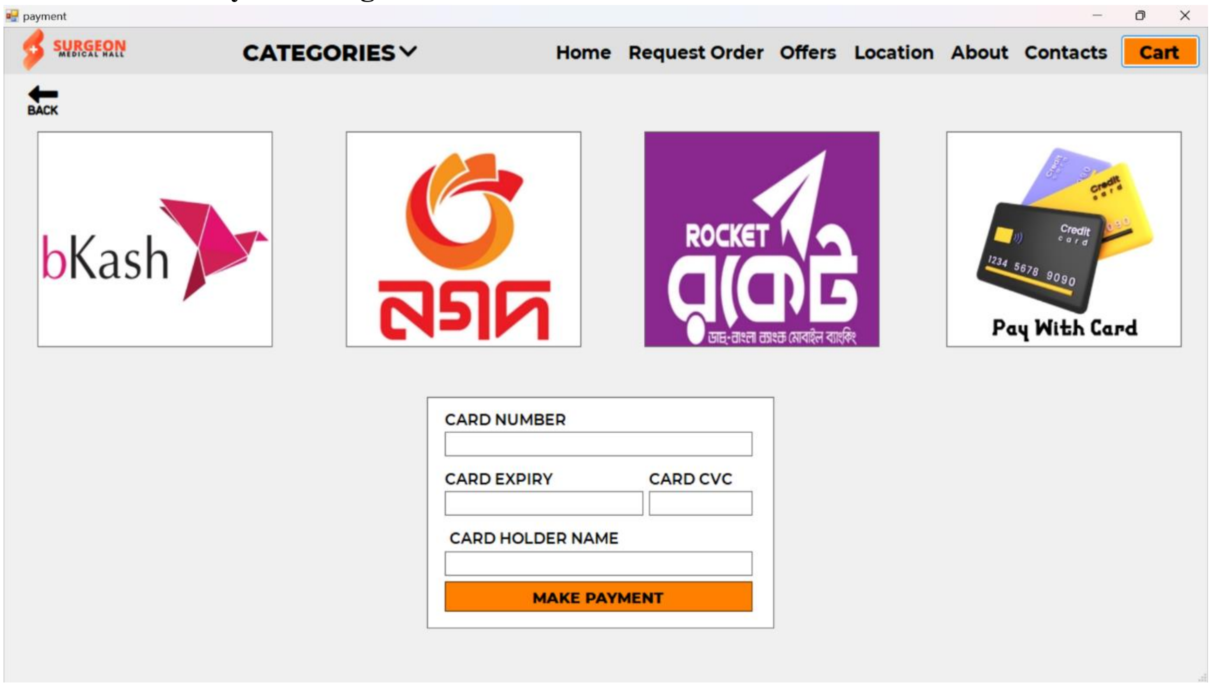
CART LIST

pID	pName	price	pStatus	username
p-010	OG CARE BLOOD GLUCOSE METTER ...	2650	Cart	kasiam
p-012	TAYLOR BRACE TYNOR . SURGICAL	5098	Cart	kasiam

Success

Your total bill is: 7748 bdt

Screenshot of Payment Page

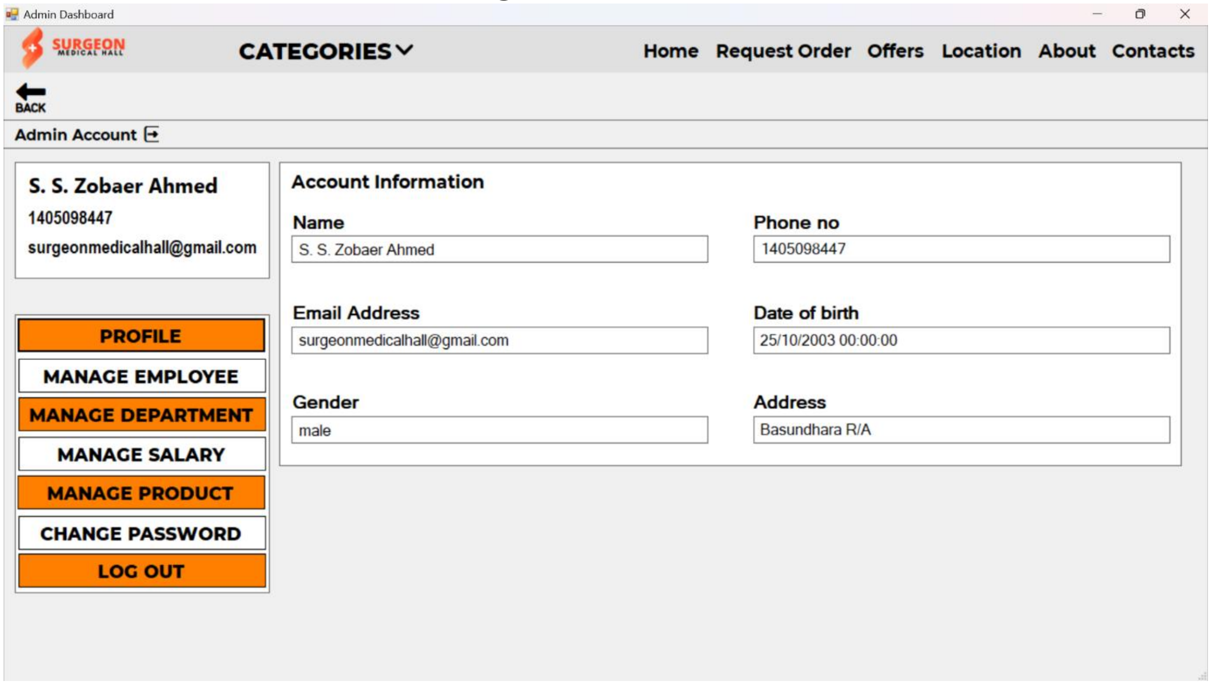


CARD NUMBER

CARD EXPIRY | CARD CVC

CARD HOLDER NAME

ScreenShot of Admin Dashboard Page

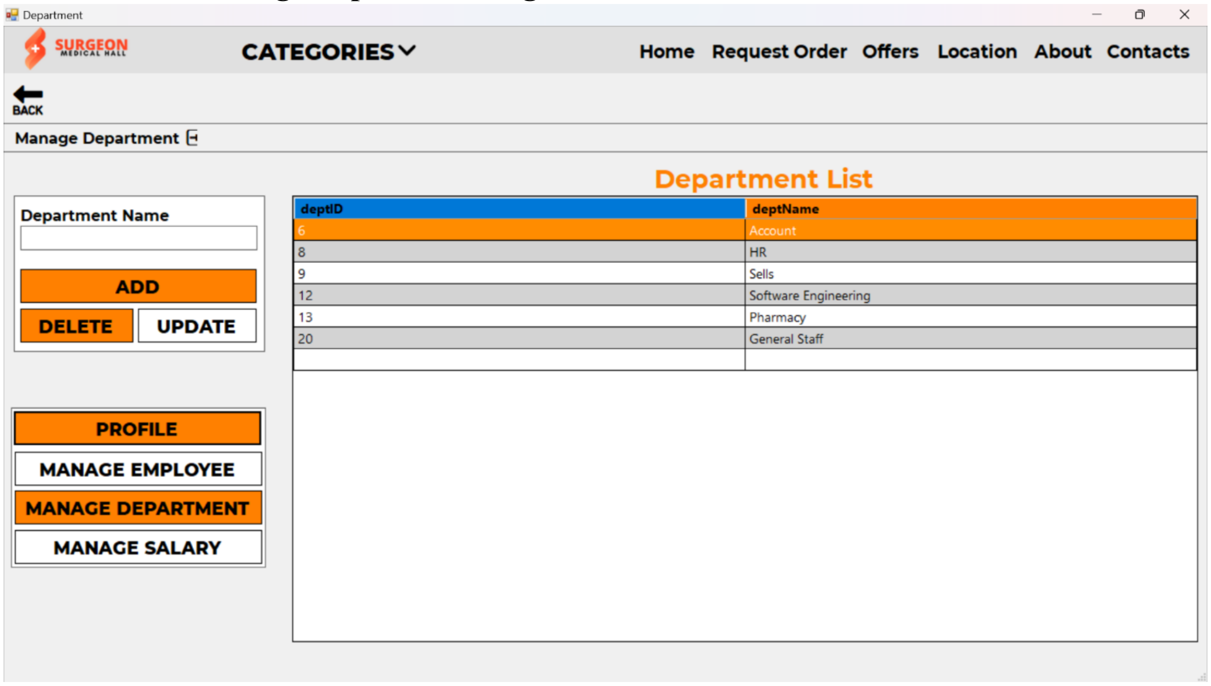


S. S. Zobaer Ahmed
1405098447
surgeonmedicalhall@gmail.com

Account Information

Name: S. S. Zobaer Ahmed	Phone no: 1405098447
Email Address: surgeonmedicalhall@gmail.com	Date of birth: 25/10/2003 00:00:00
Gender: male	Address: Basundhara R/A

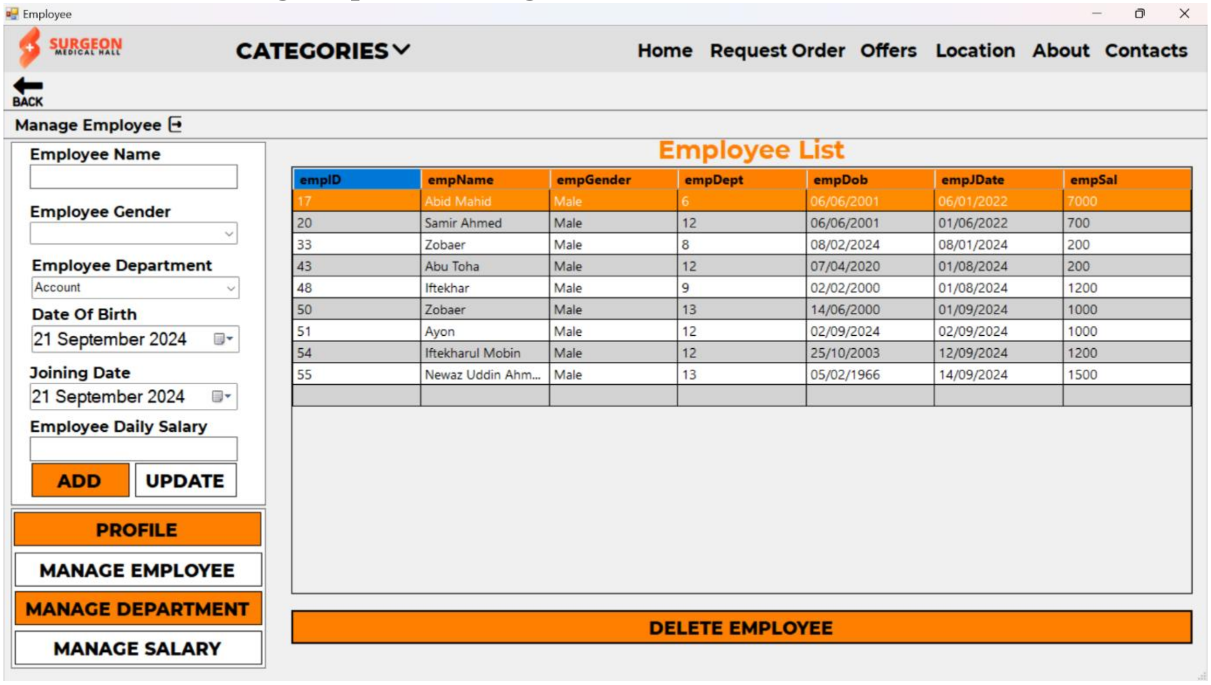
Screenshot of Manage Department Page



Department List

deptID	deptName
6	Account
8	HR
9	Sells
12	Software Engineering
13	Pharmacy
20	General Staff

Screenshot of Manage Department Page



Employee Name

Employee Gender

Employee Department: Account

Date Of Birth: 21 September 2024

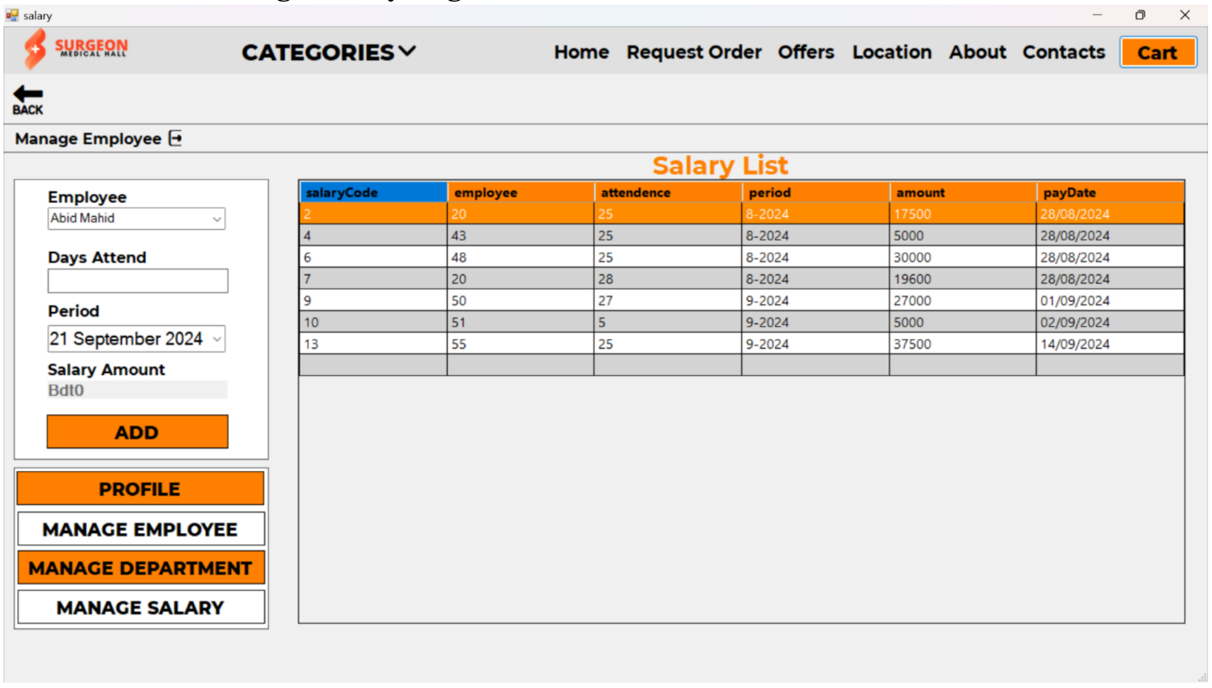
Joining Date: 21 September 2024

Employee Daily Salary

Employee List

empID	empName	empGender	empDept	empDob	empJDate	empSal
17	Abid Mahid	Male	6	06/06/2001	06/01/2022	7000
20	Samir Ahmed	Male	12	06/06/2001	01/06/2022	700
33	Zobaer	Male	8	08/02/2024	08/01/2024	200
43	Abu Toha	Male	12	07/04/2020	01/08/2024	200
48	Iftekhar	Male	9	02/02/2000	01/08/2024	1200
50	Zobaer	Male	13	14/06/2000	01/09/2024	1000
51	Ayon	Male	12	02/09/2024	02/09/2024	1000
54	Iftekharul Mobin	Male	12	25/10/2003	12/09/2024	1200
55	Newaz Uddin Ahm...	Male	13	05/02/1966	14/09/2024	1500

Screenshot of Manage Salary Page



Employee: Abid Mahid

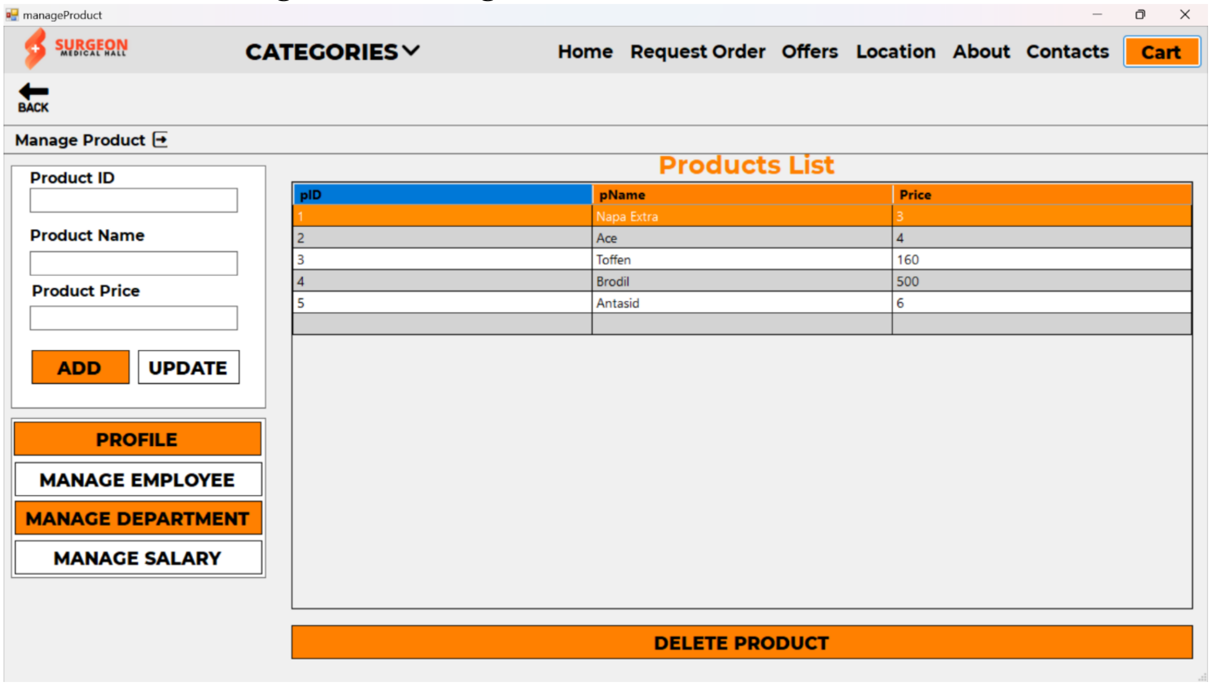
Days Attend:

Period: 21 September 2024

Salary Amount: Bdt0

salaryCode	employee	attendence	period	amount	payDate
2	20	25	8-2024	17500	28/08/2024
4	43	25	8-2024	5000	28/08/2024
6	48	25	8-2024	30000	28/08/2024
7	20	28	8-2024	19600	28/08/2024
9	50	27	9-2024	27000	01/09/2024
10	51	5	9-2024	5000	02/09/2024
13	55	25	9-2024	37500	14/09/2024

Screenshot of Manage Products Page



Product ID:

Product Name:

Product Price:

pID	pName	Price
1	Napa Extra	3
2	Ace	4
3	Toffen	160
4	Brodil	500
5	Antasid	6

- # Appendix C:

User Manual

Version: 1.0
Prepared by: S. S. Zobaer Ahmed

1. Introduction
The Pharmacy Management System (PMS) is designed to streamline pharmacy operations for both users and admins. It provides features such as product management, cart and orders, multiple payment methods, employee management, and salary calculation based on attendance. The system also includes separate login functionalities for users and admins, offering a tailored experience based on roles.

2. System Requirements

- **Operating System:** Windows 7 or higher
- **Database:** SQL Server
- **Framework:** .NET Framework 4.7 or higher
- **Hardware:**
  - Processor: Intel Core i3 or higher
  - RAM: 4GB or higher
  - Hard Disk: 500MB free space

3.Getting Started

Installation:

1. Download the Pharmacy Management System installer.
2. Run the installer and follow the on-screen instructions to complete the installation.
3. Ensure SQL Server is properly set up and connected.
   Sign Up / Login

- **Sign Up:** New users must register by providing username, email, password, and confirming the password. Admins need to be pre-registered by the system admin.
- **Login:** Users and Admins can log in using their credentials. After login, the appropriate dashboard (User/Admin) will be displayed.

4. **System Features**

User Features

Dashboard Overview:

The user dashboard provides an overview of available products, current orders, and account settings.

Product Management:
Users can browse through the product catalog and add products to the cart by selecting items from the product list.

Cart and Orders:
The cart displays products added by the user with details like price and quantity. Orders can be placed from the cart, and the product status will change to "Order placed."

Payment Methods:
The system supports multiple payment methods:

- bKash
- Nagad
- Rocket
- Bank Transfer

Order History:
Users can view their past orders, including details of the products and the status (completed, pending, etc.).
Password Management:

- Users can change their passwords via the account settings. The new password must match the confirmation password.
- The "Forgot Password" feature helps users reset their password via email. Viewing Offers, Location, and Contact Info
- Users can see pharmacy offers, locations, and contact details from the dashboard. Admin Features

Employee Management:

- Admins can manage employees, including adding, updating, and removing employees.
- Employee details like Name, ID, Date of Birth, Daily Salary, and Joining Date are tracked.

Department Management:
Admins can create and manage different pharmacy departments (e.g., Sales, Inventory, HR).

Salary Management:
Admins can calculate and process employee salaries based on attendance and commissions.

5. Navigation and Layout

Main Menu:
The drop-down menu provides easy navigation to all system features, including Product Management, Cart, Orders, Payment Methods, and Admin-specific modules like Employee and Salary Management.

User/ Admin Role Separation:
The interface dynamically adjusts based on whether the logged-in account is a user or an admin. For instance, only admins have access to employee and salary management.

6. Troubleshooting

- **Cannot connect to the database:** Ensure that the SQL Server is properly configured and running.
- **Forgot password not working:** Check your email spam folder for the reset link.
- **Error during order placement**: Ensure all required fields are filled in before placing an order.

8. Contact and Support
For any issues or support, please contact the system administrator at [Your Pharmacy Support Contact] or visit the help desk at [Your Pharmacy Location].

Application Link

https://github.com/sszobaer/PharmacyApplicationManagementSystem

---THE END---