OXFORD
COLLEGE OF ENGINEERING & MANAGEMENT
(AFFILIATED TO POKHARA UNIVERSITY)

qualityaustria
SYSTEM CERTIFIED
ISO 9001:2015 No.18386/0
An ISO 9001:2015 Certified College

Gaindakot-2, Nawalparasi
Nepal

CANTEEN MANAGEMENT SYSTEM

PROJECT REPORT

PREPARED BY:

CRF CREW

Citiz Shrestha
Regina Gharti Magar
Farhan Alam

A Project Report

On

Project II: PRJ 151
CANTEEN MANAGEMENT SYSTEM

Submitted by:

Farhan Alam	**22530193**
Regina Gharti Magar	**22530222**
Citiz Shrestha	**22530191**

Under the Guidance
of
Mr. Dipendra Silwal

Submitted to the Faculty of Science and Technology,

OCEM

in partial fulfilment of the requirements for

Fourth Semester Bachelor of Computer Application (BCA)

Year: 2021

OXFORD
COLLEGE OF ENGINEERING & MANAGEMENT
(AFFILIATED TO POKHARA UNIVERSITY)

qualityaustria
SYSTEM CERTIFIED
ISO 9001:2015 No.18386/0
An ISO 9001:2015 Certified College

Gaindakot-2, Nawalparasi
Nepal

CERTIFICATE OF AUTHENTICATED WORK

This is to certify that the project report entitled **CANTEEN MANAGEMENT SYSTEM** submitted to **Department of Bachelor of Computer Application, Oxford College of Engineering and Management** in partial fulfilment of second semester is an original work carried out by:

Farhan Alam	**22530193**
Regina Gharti Magar	**22530222**
Citiz Shrestha	**22530191**

Under our guidance and supervision. The matter embodied in this project is authentic and genuine work done by the group and has not been submitted in other college/institution/University for the fulfilment of the requirement of any course of study.

……………
Dipendra Silwal

………………..
Name of the External Examiner
External Evaluator

…………………
Sushant Tiwari
HoD, BCA

Acknowledgements

We would like to express our sincere gratitude to all those who contributed to the successful completion of the Canteen Management System project. This endeavor would not have been possible without the support and collaboration of many individuals and groups.

First and foremost, we extend our heartfelt thanks to our project supervisor, **Dipendra Silwal Sir**. Your guidance, expertise, and continuous support have been invaluable throughout this journey. Your insightful feedback and constructive criticism motivated us to strive for excellence, encouraging us to think critically and innovatively. Your unwavering belief in our abilities kept us focused on our goals, even during challenging times. We are deeply grateful for the time and effort you invested in our learning.

We also wish to thank our fellow classmates in the **BCA 4th semester**. The collaborative spirit and enthusiasm that you brought to this project have made the experience both enjoyable and enriching. The brainstorming sessions, discussions, and collective problem-solving allowed us to share diverse ideas and refine our approach, ultimately leading to a more robust and effective system. Your camaraderie and shared commitment to this project have made this journey memorable.

Additionally, we appreciate the cooperation and assistance of the **canteen staff and management**. Your willingness to share your experiences and challenges provided us with a real-world perspective that was crucial in identifying the key areas for improvement. The insights you offered informed the design of our system and helped us tailor our solutions to better meet the needs of both the staff and customers.

We would also like to acknowledge the support of the **Oxford College of Engineering and Management** for providing us with the necessary resources and environment to pursue our project. Access to technology, study materials, and a conducive learning atmosphere has played a significant role in our success.

Lastly, we would like to express our heartfelt gratitude to our families and friends for their unwavering support and encouragement. Your understanding during the long hours of work and study made this project possible. Your patience and belief in our capabilities have been a source of strength, helping us overcome obstacles and stay motivated throughout this journey.

This project is a culmination of collective efforts, and we are grateful to everyone who played a role in bringing our vision to life. It is a testament to what can be achieved through collaboration, perseverance, and shared passion. Thank you all for your contributions and support.

Roles and Responsibility

CANTEEN MANAGEMENT SYSTEM 1st October, 2024

Name of the team Member	ROLE	Tasks and Responsibilities
1. Farhan Alam	Overall Operation (Project Leader)	- - Coordinating the project, managing deadlines, assisting with design and coding
2. Regina Gharti Magar	Developer / Designer	- - Coding, designing, ensuring the user interface and user experience meet project goals
3. Citiz Shrestha	Developer / Researcher	- - Developing core functionalities, conducting research to support project development

Name and Signature of the Project Team Member/s:

Signature

1. Farhan Alam

2. Regina Gharti Magar

3. Citiz Shrestha

Signature of the Project Guide :…………………….. 24 October 2024

Abstract/Executive Summary

The **Canteen Management System** project, undertaken by students of the BCA 4th semester, seeks to modernize and optimize the daily operations of our campus canteen by addressing the persistent challenges faced by both customers and staff. Motivated by the frequent frustrations of long waiting lines, slow service, and inefficient payment methods that hinder the overall experience, our project aims to replace these outdated manual processes with a streamlined, automated system that leverages modern technology. By integrating features such as digital order placement, real-time inventory tracking, and cashless transactions, our system not only significantly reduces waiting times but also enhances service efficiency, ultimately leading to improved customer satisfaction. Moreover, the design of this system focuses on minimizing human errors and creating a user-friendly interface, ensuring that both customers and staff can easily navigate and utilize its features. By providing faster and more convenient access to meals and snacks, the system aspires to transform the way the canteen operates, fostering a smoother, more efficient, and enjoyable environment for all users. Through this innovative approach, we aim to set a new standard for campus dining experiences and contribute to the overall modernization of our canteen services, ensuring that they meet the expectations of today's tech-savvy users while accommodating future growth and scalability.

Table of Contents

Chapter 1: Introduction 1

1.1Introduction: 1

1.3 Objectives: 1

1.4 Scope and Limitation: 2

Scope: 2

Features: 2

Limitations: 2

Chapter 2 : Requirement Analysis and Feasibility Study 3

2.1 Related works 3

2.2 Requirement Analysis 3

2.2.1 Functional requirements 4

Use Case Diagram 5

2.2.2 Non functional requirements 7

2.3 Feasibility Analysis 8

2.3.1 Economic Feasibility 8

2.3.2 Operational Feasibility 8

2.3.3 Technical Feasibility 8

2.4 Structuring system requirements 9

Hardware Requirement: 9

Software Requirement 9

3.1 Data Flow Diagram (DFD) 10

3.2 UML Class Diagram 15

3.3 Detailed Budget 17

3.4 Detailed Timeline 18

Chapter 4 : Coding of the System 19

4.1 Welcome Page 19

4.2 Login Page 22

4.3 Register Page 27

4.4 Home Page 35

4.5 Admin Page 40

Chapter 5: Implementation and Testing 46

5.1 Implementation 46

5.1.1 Tools Used 46

4.1.2 Description/Listing of major classes/methods 48

4.2 Testing 49

Chapter 6: Conclusion and Recommendation 51

6.1 Conclusion: 51

6.2 Recommendations 52

References / Bibliography 53

Chapter 1: Introduction

1.1Introduction:

The **Canteen Management System (CMS)** project was developed to modernize and automate the traditional manual operations of a canteen. Built using Visual Basic in the .NET framework, this system seeks to resolve common issues such as long waiting times, manual billing errors, and inefficiencies in inventory management. CMS enhances the overall user experience by introducing digital ordering, real-time inventory tracking, and automated payment processes, all within a user-friendly interface. The project reflects a growing trend toward automation and digitization in service-based industries, aiming to streamline the canteen's operations, improve customer satisfaction, and provide valuable insights through data analytics. Through the CMS, canteen administrators gain access to comprehensive dashboards that monitor sales, customer behaviours, and inventory levels, while customers enjoy a seamless and efficient ordering experience.

This introduction serves as the foundation of the project report, providing readers with a clear understanding of the problem being addressed, the innovative solutions being implemented, and the broader impact of the system on canteen management. The subsequent sections will explore the project objectives and define the scope and limitations of the system.

1.3 Objectives:

The objectives of the **Canteen Management System** focus on solving the identified problems of manual processes and inefficiencies in the canteen. The following objectives have been established to guide the development of the system:

- **Automate Order and Payment Processes**: Provide customers with a digital platform to place orders and make payments, reducing queue times and improving convenience.
- **Implement Instant Bill Printing**: Ensure quick and accurate billing by allowing customers to print invoices directly from the system after payment.
- **Enhance User Experience with a Friendly UI**: Develop a simple, interactive user interface that makes ordering easy for customers and system management efficient for administrators.
- **Provide Real-Time Dashboard Analytics**: Equip administrators with a comprehensive dashboard that offers real-time insights into sales, order statuses, customer behaviors, and inventory levels.
- **Integrate Inventory & Supplier Management**: Automate inventory tracking with real-time monitoring, enabling admins to manage stock levels, receive low-stock alerts, and ensure continuous availability of menu items.
- **Enable Order and Cart Management**: Allow customers to modify their orders via the "View Plate" feature, ensuring flexibility before finalizing their purchases.

1.4 Scope and Limitation:

The **Canteen Management System** encompasses several key technologies, features, and functionalities that define the scope of the project. However, there are also certain limitations to be considered:

Scope:

- **Technologies Used**:
    - Developed using **Visual Basic in .NET framework** for robust and scalable system architecture.
    - Integration of **instant bill printing** and **feedback collection** directly within the application.

Features:

- **User-Friendly Interface** for customers and admins to navigate the system effortlessly.
- **Cart & Order Management** with a "View Plate" feature allowing customers to review and adjust their orders before checkout.
- **Comprehensive Admin Dashboard** providing real-time analytics on sales, orders, customer preferences, and inventory management.
- **Inventory & Supplier Management** with real-time tracking, stock alerts, and order processing automation.
- **Instant Invoice Printing** to streamline the billing process and enhance customer experience.
- **Order and Payment Automation** to facilitate faster and more efficient transaction processing.

Limitations:

- **Limited Scalability**: The system is designed specifically for small to medium-sized canteens and may require significant upgrades for use in larger, more complex environments.
- **Internet Dependency**: Some features, such as real-time updates and online payments, rely on a stable internet connection.
- **Geographic Restrictions on Payment Integration**: Integration with certain payment gateways may be limited by geographical region, impacting the availability of specific payment options.
- **Hardware Dependency**: Requires compatible devices for administrators to manage orders, payments, and inventory, which may incur additional costs for the canteen.

Chapter 2 : Requirement Analysis and Feasibility Study

2.1 Related works

Before delving deeper into the development and features of the **Canteen Management System (CMS)**, it is essential to review existing research and related works in this domain. This project builds on a foundation of previous studies and projects that have addressed similar challenges in canteen management. The following literature and resources provide valuable insights into the current state of canteen management systems and their application in academic and commercial settings.

Reddy, GS 2023, "Canteen Management System," IJERT,
This article presents a comprehensive overview of the development and implementation of a canteen management system, focusing on automating manual processes within a canteen environment. Reddy's research highlights the effectiveness of integrating technology to reduce human error, speed up service times, and improve customer satisfaction. The system developed in this work offers features such as digital ordering, payment gateways, and inventory management, aligning closely with the objectives of our project. The article also discusses the challenges faced during system deployment and offers solutions to overcome common obstacles in canteen automation.

Project report on Canteen Management System | Academic Projects,
This project report explores the design and architecture of a canteen management system developed for academic purposes. It outlines the key components of the system, including order management, payment processing, and inventory control. The report also delves into the technical specifications required for implementing such a system in a real-world scenario, making it a useful reference for understanding the practical aspects of system development. It provides insights into user interface design and backend database management, which were valuable for shaping our own project.

These related works and resources provided us with a solid foundation to develop our **Canteen Management System**. By reviewing previous research and leveraging existing tools, we were able to refine our approach, enhance system functionality, and anticipate potential challenges. The insights gained from these works informed our decision-making processes and contributed to the overall feasibility and success of the project.

2.2 Requirement Analysis

Requirement analysis is a critical step in the development of the **Canteen Management System (CMS)**. It ensures that the system meets the functional and non-functional needs of users and administrators alike. This section outlines the functional and non-functional requirements that guide the system's design and implementation.

2.2.1 Functional requirements

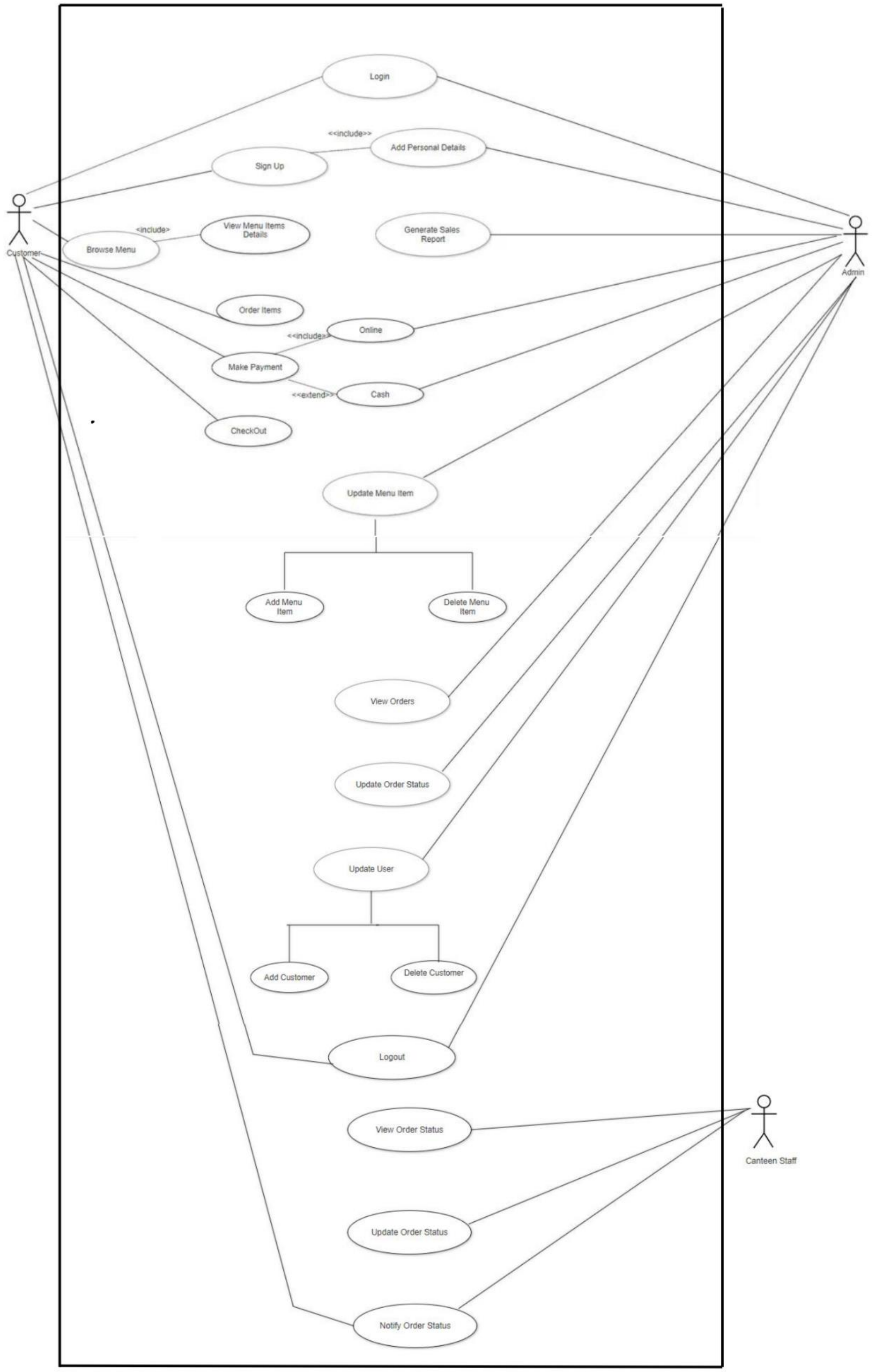
1. **User Management:**
    - Ability to create, edit, and delete user accounts for canteen staff.
    - Option to assign different access levels to user accounts (e.g., cashier, inventory manager).
2. **Item Management:**
    - Capability to add, edit, and delete items offered in the canteen, including name, description, price, and category.
    - Functionality to maintain stock levels for each item and set up low-stock alerts.
3. **Customer Management:**
    - Optional feature to create and manage customer profiles.
4. **Order Management:**
    - Capability for customers to place orders, either in person
    - Option to customize orders (e.g., adding special instructions).
    - Display of order details including item names, quantities, and total price.
    - Ability to modify orders before confirmation.
5. **Billing:**
    - Automated calculation of order totals.
    - Provision of various payment options (e.g., cash, card).
    - Generation of receipts for orders.
6. **Reporting:**
    - Generation of reports on sales, inventory levels, and other relevant data.
    - Ability to filter and export reports in different formats

Use Case Diagram

A Use Case Diagram is a visual representation that illustrates how users interact with a system and its functionalities. It depicts the various use cases, actors (users or external systems), and their relationships within the system. Use cases represent specific interactions between users (actors) and the system to achieve particular goals. Actors are individuals, external systems, or other entities that interact with the system, depicted as stick figures or blocks outside the system boundary. Relationships between actors and use cases show which actors are involved in each use case, clarifying system behavior and user roles. Use Case Diagrams are invaluable tools in system analysis and design, aiding in requirements gathering, communication among stakeholders, and validation of system functionality.

Components of a Use Case Diagram:

1. **Actors**:
   - Represent users, external systems, or other entities interacting with the system.
   - Shown outside the system boundary as stick figures or blocks.
   - Example: *Customer*, *Administrator*.
2. **Use Cases**:
   - Represent specific functionalities or actions the system performs to achieve a goal for an actor.
   - Shown inside the system boundary as ovals.
   - Example: *Place Order*, *Generate Report*.
3. **Relationships**:
   - Connect actors with use cases to indicate which actors are involved in each use case.
   - Arrowed lines show the direction of interaction.
   - Example: *Customer* interacts with *Place Order*.
4. **System Boundary**:
   - Represents the scope and boundary of the system under consideration.
   - Use cases and actors are placed inside this boundary.
5. **Include and Extend Relationships** (optional):
   - **Include**: Indicates that one use case includes the functionality of another.
   - **Extend**: Shows optional or exceptional behavior that extends a base use case.


Login
<<include>>
Add Personal Details
Sign Up
Customer
<include>
View Menu Items Details
Browse Menu
Generate Sales Report
Admin
Order Items
<<include>>
Online
Make Payment
<<extend>>
Cash
CheckOut
Update Menu Item
Add Menu Item
Delete Menu Item
View Orders
Update Order Status
Update User
Add Customer
Delete Customer
Logout
View Order Status
Canteen Staff
Update Order Status
Notify Order Status

2.2.2 Non functional requirements

1. **Security:**
   - Implementation of robust user authentication and authorization mechanisms.
   - Secure storage of sensitive data such as customer information and financial data.
2. **Performance:**
   - Responsive system handling user requests efficiently.
   - Minimal downtime to ensure continuous operation.
3. **Usability:**
   - Intuitive user interface that is easy to navigate for users with varying technical skills.
   - Accessibility features to accommodate users with disabilities.
4. **Reliability:**
   - Reliable system operation to minimize disruptions to canteen operations.
   - Data integrity measures to prevent loss or corruption of critical information.
5. **Maintainability:**
   - Well-documented codebase to facilitate ease of maintenance and future enhancements.
   - Modular design to enable scalability and flexibility in accommodating changes or updates.

By adhering to these high-level requirements, the proposed Canteen Management System (CMS) will strive to meet the functional needs of canteen operations while also addressing critical non-functional aspects such as security, performance, usability, reliability, and maintainability. These requirements form the foundation for the development of a robust and user-centric solution that aims to enhance efficiency, accuracy, and customer satisfaction in canteen management

2.3 Feasibility Analysis

The feasibility analysis evaluates whether the **Canteen Management System** is viable from various perspectives, including economic, operational, and technical feasibility.

2.3.1 Economic Feasibility

The economic feasibility of the CMS project assesses whether the expected benefits outweigh the costs involved in developing, implementing, and maintaining the system. In this case, the costs of development are primarily associated with software tools, hardware equipment (e.g., computers, printers), and any required licensing for payment gateway integration. The economic benefits include increased operational efficiency, reduced labor costs through automation, and improved customer satisfaction, which can potentially lead to higher revenue for the canteen. The ROI is expected to be positive within the first year of operation as the system significantly reduces order processing time and minimizes billing errors.

2.3.2 Operational Feasibility

Operational feasibility evaluates the ability of the canteen staff and customers to effectively use the CMS. The system is designed with a user-friendly interface for both customers and admins, making it easy to adopt without the need for extensive training. The automated order management and payment processing features ensure that the system integrates seamlessly into existing workflows, reducing manual effort and improving the speed and accuracy of operations. Feedback from potential users has indicated that the system meets operational needs and provides a more efficient and satisfying user experience.

2.3.3 Technical Feasibility

The technical feasibility assesses whether the technology required for the CMS is available and appropriate for the project's requirements. The system is developed using Visual Basic in the .NET framework, both of which are well-established and widely supported technologies. The use of cloud-based databases for real-time inventory tracking and order management ensures that the system can handle large volumes of data with minimal latency. The payment gateway integration relies on secure and reliable third-party services that are compatible with the CMS architecture. Additionally, the hardware requirements for the system (such as printers for invoices and computers for order management) are readily available and affordable, making the technical implementation straightforward.

2.4 Structuring system requirements

This includes the development environment and the operating system in which this Banking Management system is build. The type of processors required in the hardware to develop this application ranging from front end to back end.

Hardware Requirement:

Processor: Intel(R) Core

Installed Memory: 8.00GB

Speed: 1.70GHz or faster

Operating System: 64-Bit Operating system, x64- based processor

Software Requirement

Operating System: Windows 11

IDE & Tools: Microsoft Visual Studio , .Net Framework 4.1 , Visual Basic Language

Chapter 3 : System Design

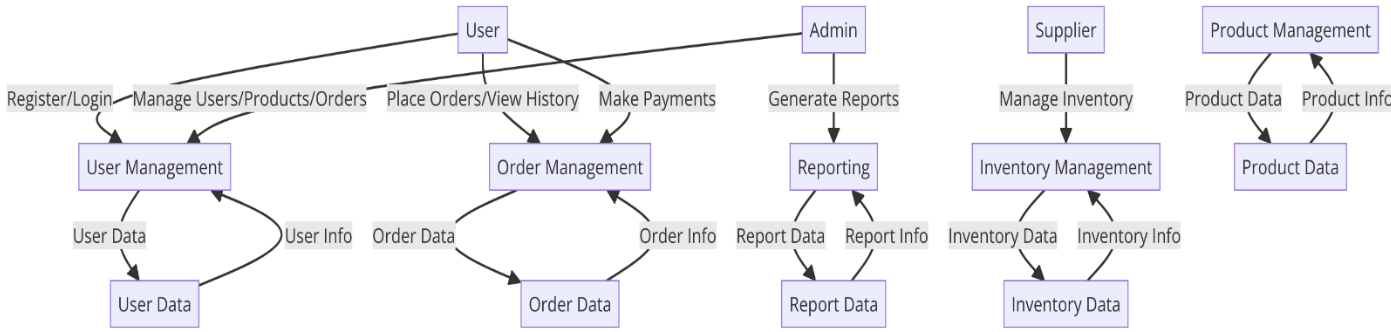
3.1 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a visual representation of how data flows through a system and how it is processed. It illustrates the movement of data between processes, data stores, and external entities. Processes represent functions or transformations that occur within the system, manipulating data inputs to produce outputs. Data stores depict where data is persisted within the system. External entities are sources or destinations of data outside the system boundary. Arrows indicate the direction of data flow, showing how data moves through the system and undergoes transformation. DFDs are instrumental in system analysis and design, facilitating a clear understanding of data flow and enabling stakeholders to identify inefficiencies, redundancies, and opportunities for improvement within the system architecture.

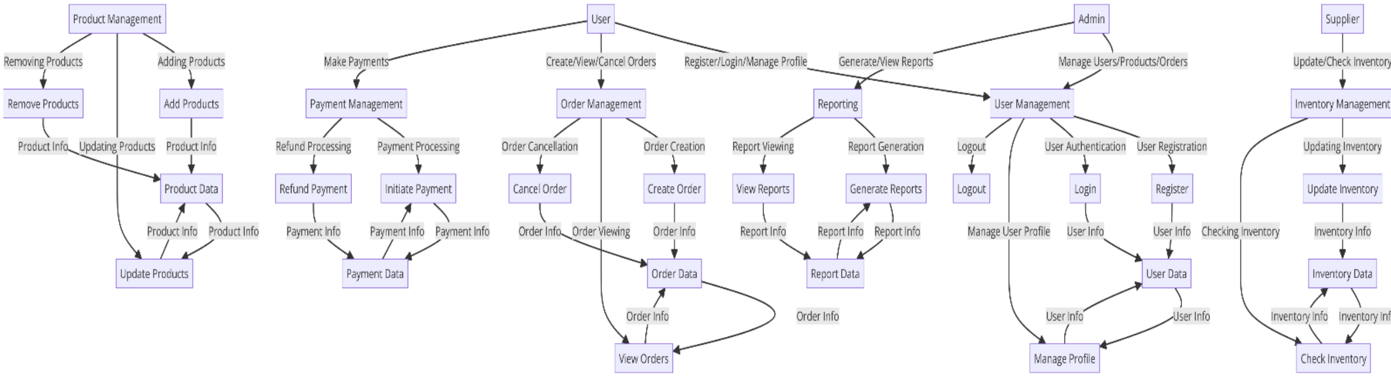
Components of a Data Flow Diagram:

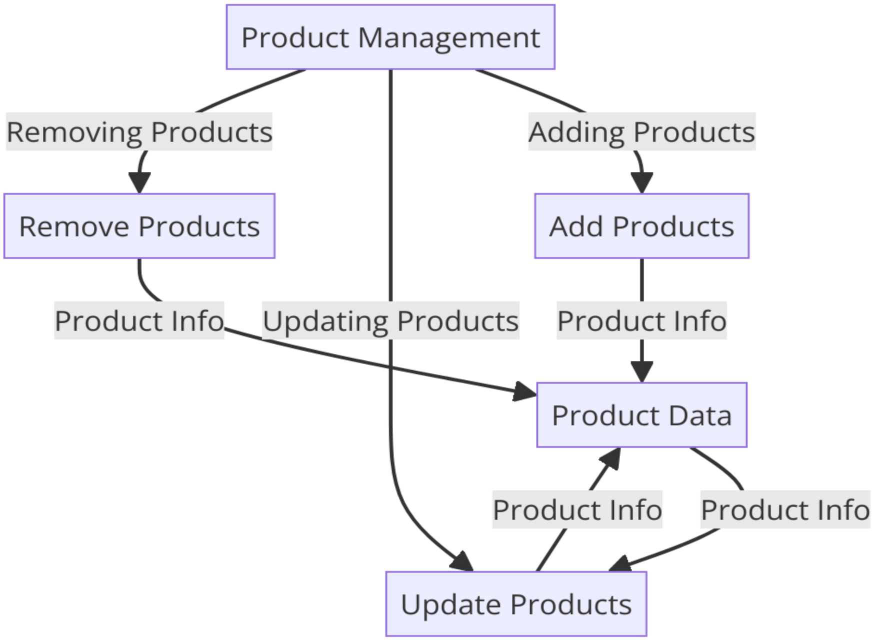
1. **Processes**:
   - Represent activities or transformations that manipulate data inputs to produce outputs.
   - Shown as circles or ovals with meaningful labels.
   - Example: *Process Order*, *Calculate Payroll*.
2. **Data Flows**:
   - Represent the movement of data between processes, data stores, and external entities.
   - Arrows indicate the direction of data flow.
   - Example: Data flowing from *Customer* (external entity) to *Place Order* (process).
3. **Data Stores**:
   - Represent where data is stored or persisted within the system.
   - Shown as rectangles.
   - Example: *Customer Database*, *Inventory Database*.
4. **External Entities**:
   - Represent sources or destinations of data outside the system boundary.
   - Shown as squares or rectangles.
   - Example: *Customer*, *Supplier*.

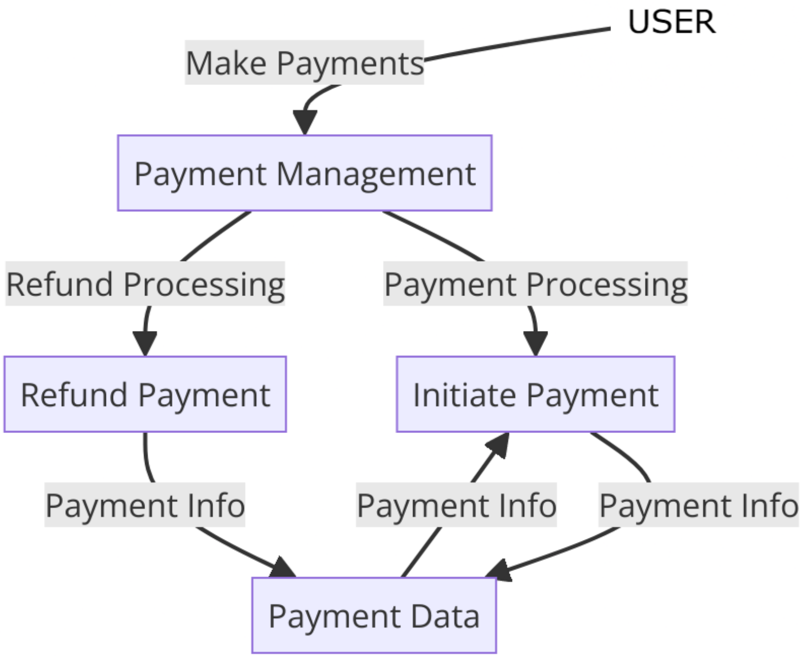
Level 0 DFD

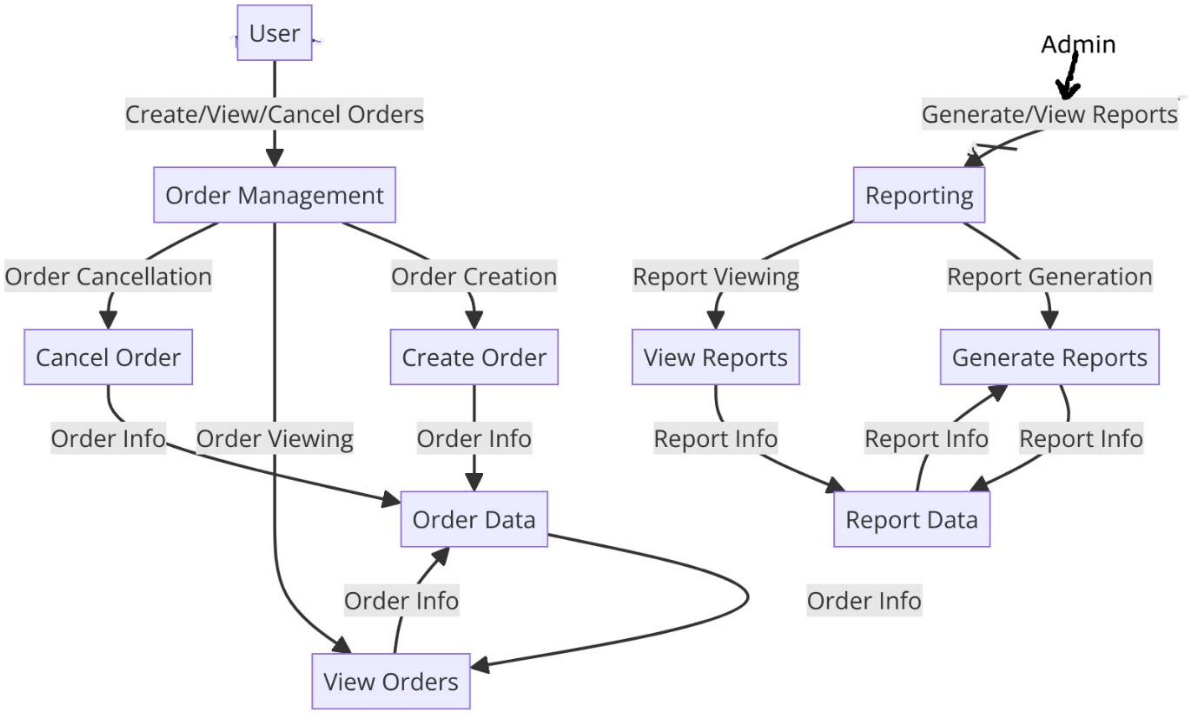


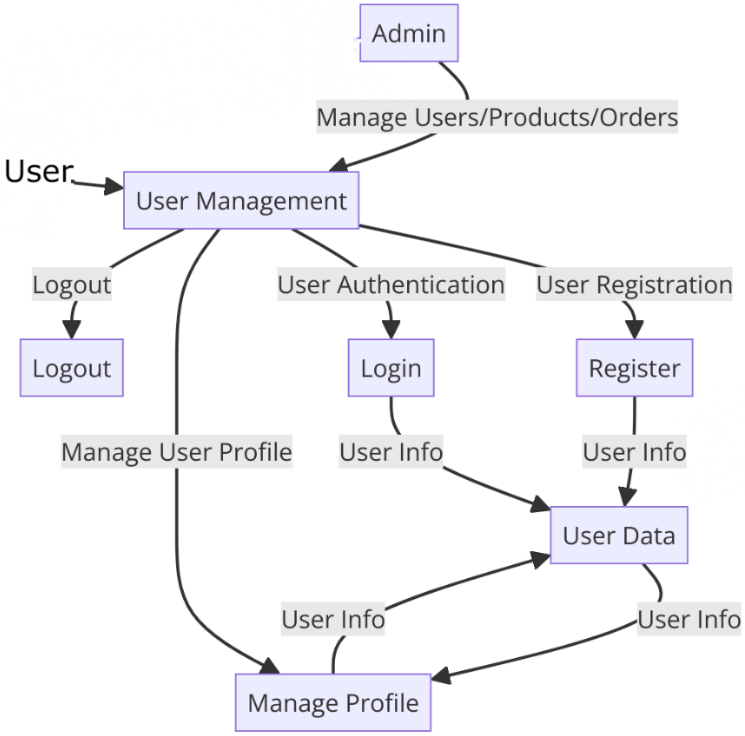
Level 1 DFD

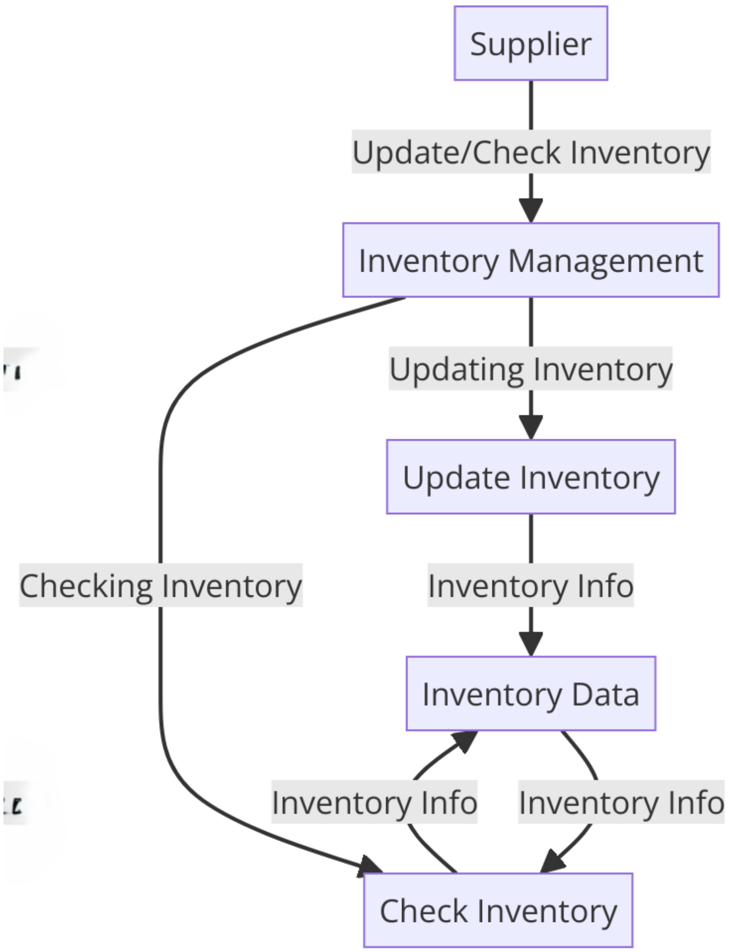



Product Management
Removing Products
Adding Products
Remove Products
Add Products
Product Info
Updating Products
Product Info
Product Data
Product Info
Product Info
Update Products


USER
Make Payments
Payment Management
Refund Processing
Payment Processing
Refund Payment
Initiate Payment
Payment Info
Payment Info
Payment Info
Payment Data


User
Create/View/Cancel Orders
Order Management
Order Cancellation
Order Creation
Cancel Order
Create Order
Order Info
Order Viewing
Order Info
Order Data
Order Info
View Orders
Admin
Generate/View Reports
Reporting
Report Viewing
Report Generation
View Reports
Generate Reports
Report Info
Report Info
Report Info
Report Data
Order Info


Admin
Manage Users/Products/Orders
User
User Management
Logout
User Authentication
User Registration
Logout
Login
Register
Manage User Profile
User Info
User Info
User Data
User Info
User Info
Manage Profile

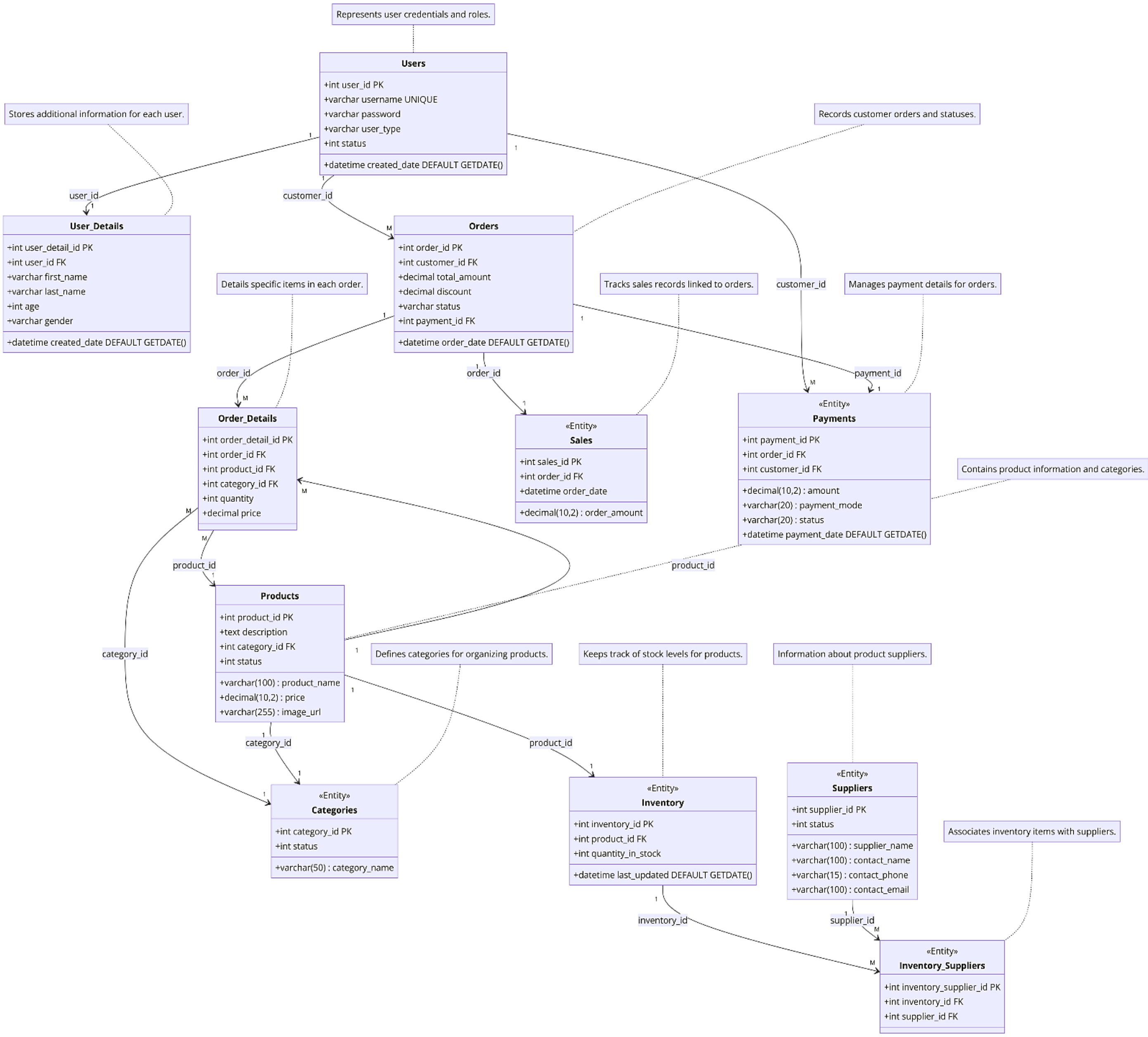

Supplier
Update/Check Inventory
Inventory Management
Updating Inventory
Update Inventory
Checking Inventory
Inventory Info
Inventory Data
Inventory Info
Inventory Info
Check Inventory

3.2 UML Class Diagram

A UML Class Diagram is a type of static structure diagram in the Unified Modeling Language (UML) that represents the structure and relationships of classes and interfaces within a system. It provides a blueprint of the system's classes, their attributes, methods, and the relationships among objects. Classes represent blueprints for objects, encapsulating data (attributes) and behaviors (methods) related to a specific entity or concept. Relationships between classes illustrate how objects collaborate and interact in the system. UML Class Diagrams are instrumental in system design and analysis, aiding in understanding class structure, identifying class responsibilities, and facilitating communication among stakeholders.

Components of a UML Class Diagram:

1. **Classes**:
   - Represent templates for objects that share common structure, behavior, and relationships.
   - Shown as rectangles with three compartments: class name, attributes, and methods.
   - Example: *Customer*, *Order*, *Product*.
2. **Attributes**:
   - Represent data or properties of a class.
   - Shown in the second compartment of the class rectangle.
   - Example: *CustomerID, Name, Address*.
3. **Methods**:
   - Represent behaviors or operations that a class can perform.
   - Shown in the third compartment of the class rectangle.
   - Example: *placeOrder(), calculateTotal()*.
4. **Relationships**:
   - Types of relationships include:
     - **Association**: Represents a relationship where objects of one class are connected to objects of another class.
     - **Aggregation**: Represents a "whole-part" relationship where one class (whole) contains objects of another class (part), but parts can exist independently.
     - **Composition**: Represents a stronger form of aggregation where the parts cannot exist without the whole.
     - **Inheritance**: Represents an "is-a" relationship where one class (subclass or derived class) inherits attributes and methods from another class (superclass or base class).
     - **Dependency**: Represents a relationship where one class relies on another class.
5. **Multiplicities**:
   - Indicate how many instances of one class are related to instances of another class in associations.
   - Shown near association lines (e.g., 1, *, 0..1).


Represents user credentials and roles.
Users
+int user_id PK
+varchar username UNIQUE
+varchar password
+varchar user_type
+int status
+datetime created_date DEFAULT GETDATE()
Stores additional information for each user.
Records customer orders and statuses.
user_id
customer_id
User_Details
+int user_detail_id PK
+int user_id FK
+varchar first_name
+varchar last_name
+int age
+varchar gender
+datetime created_date DEFAULT GETDATE()
Orders
+int order_id PK
+int customer_id FK
+decimal total_amount
+decimal discount
+varchar status
+int payment_id FK
+datetime order_date DEFAULT GETDATE()
Details specific items in each order.
Tracks sales records linked to orders.
customer_id
Manages payment details for orders.
order_id
order_id
payment_id
Order_Details
+int order_detail_id PK
+int order_id FK
+int product_id FK
+int category_id FK
+int quantity
+decimal price
«Entity»
Sales
+int sales_id PK
+int order_id FK
+datetime order_date
+decimal(10,2) : order_amount
«Entity»
Payments
+int payment_id PK
+int order_id FK
+int customer_id FK
+decimal(10,2) : amount
+varchar(20) : payment_mode
+varchar(20) : status
+datetime payment_date DEFAULT GETDATE()
Contains product information and categories.
product_id
product_id
Products
+int product_id PK
+text description
+int category_id FK
+int status
+varchar(100) : product_name
+decimal(10,2) : price
+varchar(255) : image_url
category_id
Defines categories for organizing products.
Keeps track of stock levels for products.
Information about product suppliers.
category_id
product_id
«Entity»
Categories
+int category_id PK
+int status
+varchar(50) : category_name
«Entity»
Inventory
+int inventory_id PK
+int product_id FK
+int quantity_in_stock
+datetime last_updated DEFAULT GETDATE()
«Entity»
Suppliers
+int supplier_id PK
+int status
+varchar(100) : supplier_name
+varchar(100) : contact_name
+varchar(15) : contact_phone
+varchar(100) : contact_email
Associates inventory items with suppliers.
inventory_id
supplier_id
«Entity»
Inventory_Suppliers
+int inventory_supplier_id PK
+int inventory_id FK
+int supplier_id FK
1
1
1
1
M
1
M
1
M
1
M
1
M
M
M
1
1
1
1
1
1
1
1
1
M
M

3.3 Detailed Budget

A budget is a financial plan that outlines expected income and expenses over a specific period, typically aligned with the duration of a project or fiscal year. It serves as a crucial tool for financial management and decision-making, ensuring resources are allocated efficiently to achieve organizational goals. Budgets typically include both revenue (income) and expenditures (costs), providing a comprehensive overview of financial activities

A well-structured budget not only helps in financial planning and management but also serves as a tool for tracking progress, evaluating performance, and making informed decisions to achieve financial goals effectively.

Development Phase	Estimated Cost (NRS)
Analysis and Planning	
Project Manager (1 month)	35000
Miscellaneous Expenses	2500
Contingency (10% of Total)	3850
Total for Analysis and Planning	**41,350**
Development	
Software Developers (3 months)	90,000
Database Administrators (3 months)	50,000
Miscellaneous Expenses	2,500
Contingency (10% of Total)	14,250
Total for Development	**1,56,750**
Testing and Quality Assurance	
Quality Assurance/Testers (2 months)	40,000
Miscellaneous Expenses	2,500
Contingency (10% of Total)	4,250
Total for Testing and QA	**46,750**
Deployment and Maintenance	
Miscellaneous Expenses	25,000
Contingency (10% of Total)	2500
Total for Deployment and Maintenance	**27,500**
Grand Total Estimated Budget	**2,31,000**

3.4 Detailed Timeline

A timeline is a visual representation of the sequence of events or activities within a project, outlining key milestones and deadlines. It serves as a roadmap for project management, helping to organize tasks, allocate resources, and track progress over time. A well-defined timeline ensures that project objectives are met within specified timeframes, facilitating efficient project execution and coordination among team members and stakeholders.

Phase	Duration
Requirement Analysis	2 weeks
Design	3 weeks
Development	8 weeks
Testing	4 weeks
Deployment	2 weeks
Documentation	2 weeks
Total	**21 weeks**

Chapter 4 : Coding of the System

4.1 Welcome Page

```
Imports System.Data.SqlClient
Imports System.Drawing

Public Class Welcome_Page

    ' Form load event to center the form on the screen with error handling
    Private Sub Welcome_Page_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Try
            CenterForm()
        Catch ex As Exception
            MessageBox.Show("An error occurred while loading the page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Center the form manually on the screen
    Private Sub CenterForm()
        Dim x As Integer
        Dim y As Integer
        Dim r As Rectangle

        If Not Parent Is Nothing Then
            r = Parent.ClientRectangle
            x = r.Width - Me.Width + Parent.Left
            y = r.Height - Me.Height + Parent.Top
        Else
            r = Screen.PrimaryScreen.WorkingArea
            x = r.Width - Me.Width
            y = r.Height - Me.Height
        End If

        x = CInt(x / 2)
        y = CInt(y / 2)

        Me.StartPosition = FormStartPosition.Manual
        Me.Location = New Point(x, y)
    End Sub

    ' Event handler for Sign In button click
    Private Sub BtnSignIn_Click(sender As Object, e As EventArgs) Handles
btnSignIn.Click
        Try
            Dim loginpage As New Login_Page
            loginpage.ShowDialog()
            Me.Close()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the login page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
    ' Event handler for Sign In button click
    Private Sub btnRegistration_Click(sender As Object, e As EventArgs) Handles
btnRegistration.Click
        Try
            Dim registerpage As New Register
            registerpage.ShowDialog()
            Me.Close()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the login page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub


    ' Event handler for Guest button click (Male)
    Private Sub btnGuestSignInMale_Click(sender As Object, e As EventArgs)
Handles btnGuestSignInMale.Click
        Try
            Dim homepage As New Home_Page(1011) ' Assuming 1011 is an identifier
for guest user
            homepage.ShowDialog()
            Me.Close()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the guest page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Event handler for Guest button click (Female)
    Private Sub btnSignInFemale_Click(sender As Object, e As EventArgs) Handles
btnGuestSignInFemale.Click
        Try
            Dim homepage As New Home_Page(1013) ' Assuming 1013 is an identifier
for female guest user
            homepage.ShowDialog()
            Me.Close()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the female guest
page: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' MouseEnter and MouseLeave events to change button appearance (Hover
effects)

    ' Changes the background color of the Sign In button when hovered
    Private Sub BtnSignIn_MouseEnter(sender As Object, e As EventArgs) Handles
btnSignIn.MouseEnter
        btnSignIn.BackColor = Color.FromArgb(0, 175, 99) ' Darker shade for
hover
    End Sub

    Private Sub btnSignIn_MouseLeave(sender As Object, e As EventArgs) Handles
btnSignIn.MouseLeave
        btnSignIn.BackColor = Color.FromArgb(0, 191, 99) ' Original color
    End Sub

    ' Changes the background color of the Guest button (Male) when hovered
    Private Sub btnGuest_MouseEnter(sender As Object, e As EventArgs) Handles
btnGuestSignInMale.MouseEnter
        btnGuestSignInMale.BackColor = Color.Gray
        btnGuestSignInMale.ForeColor = Color.White
    End Sub

    Private Sub btnGuest_MouseLeave(sender As Object, e As EventArgs) Handles
btnGuestSignInMale.MouseLeave
        btnGuestSignInMale.BackColor = Color.Black ' Original color
        btnGuestSignInMale.ForeColor = Color.White
    End Sub

    ' Changes the background color of the Sign Up button when hovered
    Private Sub btnRegistration_MouseEnter(sender As Object, e As EventArgs)
Handles btnRegistration.MouseEnter
        btnRegistration.BackColor = Color.FromArgb(0, 191, 99) ' Highlighted
green
        btnRegistration.ForeColor = Color.White
    End Sub

    Private Sub btnRegistration_MouseLeave(sender As Object, e As EventArgs)
Handles btnRegistration.MouseLeave
        btnRegistration.BackColor = Color.White ' Original color
        btnRegistration.ForeColor = Color.FromArgb(0, 191, 99)
    End Sub

    ' Changes the background color of the Guest button (Female) when hovered
    Private Sub btnSignInFemale_MouseEnter(sender As Object, e As EventArgs)
Handles btnGuestSignInFemale.MouseEnter
        btnGuestSignInFemale.BackColor = Color.Gray
        btnGuestSignInFemale.ForeColor = Color.White
    End Sub

    Private Sub btnSignInFemale_MouseLeave(sender As Object, e As EventArgs)
Handles btnGuestSignInFemale.MouseLeave
        btnGuestSignInFemale.BackColor = Color.Black ' Original color
        btnGuestSignInFemale.ForeColor = Color.White
    End Sub

End Class
```

4.2 Login Page

```
Imports System.Configuration
Imports System.Data.SqlClient

Public Class Login_Page

    ' Database Connection
    Private connectionString As String =
ConfigurationManager.ConnectionStrings("CMS.My.MySettings.CMSConnectionString").
ConnectionString

    Private loginAttempts As Integer = 3
    Private isPasswordVisible As Boolean = False

    ' Form load event to initialize and center the login page
    Private Sub Login_Page_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        lblmsg.Visible = True

        ' Center the form on the screen
        Dim x As Integer
        Dim y As Integer
        Dim r As Rectangle

        If Not Parent Is Nothing Then
            r = Parent.ClientRectangle
            x = r.Width - Me.Width + Parent.Left
            y = r.Height - Me.Height + Parent.Top
        Else
            r = Screen.PrimaryScreen.WorkingArea
            x = r.Width - Me.Width
            y = r.Height - Me.Height
        End If

        x = CInt(x / 2)
        y = CInt(y / 2)

        Me.StartPosition = FormStartPosition.Manual
        Me.Location = New Point(x, y)

        ' Initialize password field to hide characters
        rtxtboxpassword.PasswordChar = "●"c
    End Sub

    ' Login button click event
    Private Sub btnLogin_Click(sender As Object, e As EventArgs) Handles
btnLogin.Click
        If loginAttempts <= 0 Then
            lblmsg.Text = "No login attempts remaining."
            lblmsg.ForeColor = Color.Red
            btnLogin.Enabled = False
            btnLogin.BackColor = Color.Gray ' Disable button and change color
            Exit Sub
        End If
        Dim email As String = rtxtboxusername.Text
        Dim password As String = rtxtboxpassword.Text

        ' Input Validation
        If String.IsNullOrWhiteSpace(email) OrElse
String.IsNullOrWhiteSpace(password) Then
            MessageBox.Show("Please enter both email and password.", "Input
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        Try
            Using connection As New SqlConnection(connectionString)
                connection.Open()

                ' Check if email exists
                If Not CheckEmailExists(email, connection) Then
                    MessageBox.Show("Invalid Email!", "Login Failed",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                    DecrementLoginAttempts()
                    Exit Sub
                End If

                ' Validate email and password
                If Not ValidateUser(email, password, connection) Then
                    DecrementLoginAttempts()
                    MessageBox.Show("Incorrect Password!", "Login Failed",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Exit Sub
                End If
            End Using
        Catch ex As Exception
            MessageBox.Show($"Database error: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Check if the email exists in the database
    Private Function CheckEmailExists(email As String, connection As
SqlConnection) As Boolean
        Dim query As String = "SELECT COUNT(*) FROM Users WHERE email = @Email"
        Using command As New SqlCommand(query, connection)
            command.Parameters.AddWithValue("@Email", email)
            Dim result As Integer = Convert.ToInt32(command.ExecuteScalar())
            Return result > 0
        End Using
    End Function

    ' Validate user by checking email and password
    Private Function ValidateUser(email As String, password As String,
connection As SqlConnection) As Boolean
        Dim query As String = "SELECT user_id, user_type, status FROM Users
WHERE email = @Email AND password = HASHBYTES('SHA2_512', @Password)"
        Using command As New SqlCommand(query, connection)
            command.Parameters.AddWithValue("@Email", email)
            command.Parameters.AddWithValue("@Password", password)

            Using reader As SqlDataReader = command.ExecuteReader()
```

```
                If reader.Read() Then
                    Dim userId As Integer = Convert.ToInt32(reader("user_id")) '
Retrieve user ID
                    Dim userType As String = reader("user_type").ToString()
                    Dim status As Integer = Convert.ToInt32(reader("status"))

                    If status = 0 Then
                        MessageBox.Show("Your account is inactive. Please
contact the administrator.", "Account Inactive", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
                        Return False
                    Else
                        ShowSuccessMessage(userType, userId) ' Display success
message and navigate to the appropriate page
                        Return True
                    End If
                Else
                    Return False ' Invalid credentials
                End If
            End Using
        End Using
    End Function

    ' Show success message and redirect based on user type
    Private Sub ShowSuccessMessage(userType As String, userId As Integer)
        If userType = "admin" Then
            Dim adminPage As New Admin_Page()
            adminPage.ShowDialog()
            Me.Close()
        ElseIf userType = "customer" Then
            Dim homePage As New Home_Page(userId) ' Pass user ID to home page
            homePage.ShowDialog()
            Me.Close()
        End If
    End Sub

    ' Decrement login attempts and disable login if no attempts are left
    Private Sub DecrementLoginAttempts()
        loginAttempts -= 1
        If loginAttempts > 0 Then
            lblmsg.Text = $"You have {loginAttempts} attempt(s) remaining."
            lblmsg.ForeColor = Color.Orange
        Else
            lblmsg.Text = "No login attempts remaining."
            lblmsg.ForeColor = Color.Red
            btnLogin.Enabled = False
            btnLogin.BackColor = Color.Gray ' Disable button and change color
        End If
    End Sub

    ' Toggle password visibility
    Private Sub btnshowpassword_Click(sender As Object, e As EventArgs) Handles
btnTogglePassword.Click
        If isPasswordVisible Then
            ' Hide password
            rtxtboxpassword.PasswordChar = "●"c
            btnTogglePassword.BackgroundImage = My.Resources.Eye_Open
            isPasswordVisible = False
        Else
            ' Show password
            rtxtboxpassword.PasswordChar = ControlChars.NullChar
            btnTogglePassword.BackgroundImage = My.Resources.Eye_Close
            isPasswordVisible = True
        End If
    End Sub

    ' Cancel button to close the login page
    Private Sub btncancel_Click(sender As Object, e As EventArgs) Handles
btnCancel.Click
        Me.Close()
    End Sub

    ' Registration button click event
    Private Sub btnregister_Click(sender As Object, e As EventArgs) Handles
btnregister.Click
        Dim registerForm As New Register()
        registerForm.ShowDialog()
        Me.Close()
    End Sub

    ' Handle Forgot Password click event
    Private Sub lblforgetpassword_Click(sender As Object, e As EventArgs)
Handles lblforgetpassword.Click
        Dim forgetpassword As New ForgetPassword()
        forgetpassword.ShowDialog()
        Me.Close()
    End Sub

    ' MouseEnter and MouseLeave events to change the appearance of the "Forgot
Password" label
    Private Sub lblforgetpassword_MouseEnter(sender As Object, e As EventArgs)
Handles lblforgetpassword.MouseEnter
        lblforgetpassword.ForeColor = Color.FromArgb(0, 191, 99) ' Highlight the
text on hover
    End Sub

    Private Sub lblforgetpassword_MouseLeave(sender As Object, e As EventArgs)
Handles lblforgetpassword.MouseLeave
        lblforgetpassword.ForeColor = Color.DarkGray ' Revert the text color
when the mouse leaves
    End Sub

    ' MouseEnter and MouseLeave events for the Login button
    Private Sub btnLogin_MouseEnter(sender As Object, e As EventArgs) Handles
btnLogin.MouseEnter
        btnLogin.BackColor = Color.FromArgb(0, 175, 99) ' Darker green for hover
effect
    End Sub

    Private Sub btnLogin_MouseLeave(sender As Object, e As EventArgs) Handles
btnLogin.MouseLeave
        btnLogin.BackColor = Color.FromArgb(0, 191, 99) ' Original color
    End Sub

    ' MouseEnter and MouseLeave events for the Cancel button
    Private Sub btncancel_MouseEnter(sender As Object, e As EventArgs) Handles
btnCancel.MouseEnter
        btnCancel.BackColor = Color.FromArgb(0, 191, 99) ' Highlight text on
hover
        btnCancel.ForeColor = Color.White
    End Sub

    Private Sub btncancel_MouseLeave(sender As Object, e As EventArgs) Handles
btnCancel.MouseLeave
        btnCancel.BackColor = Color.White
        btnCancel.ForeColor = Color.FromArgb(0, 191, 99) ' Revert text color
    End Sub

    ' MouseEnter and MouseLeave events for the Register button
    Private Sub btnregister_MouseEnter(sender As Object, e As EventArgs) Handles
btnregister.MouseEnter
        btnregister.ForeColor = Color.FromArgb(0, 191, 99) ' Highlight text on
hover
    End Sub

    Private Sub btnregister_MouseLeave(sender As Object, e As EventArgs) Handles
btnregister.MouseLeave
        btnregister.ForeColor = Color.FromArgb(0, 175, 99) ' Revert text color
    End Sub
End Class
```

4.3 Register Page

```
Imports System.Configuration
Imports System.Data.SqlClient
Imports System.Text.RegularExpressions

Public Class Register

    Private isAdminMode As Boolean

    ' Modify the constructor to accept the "isAdminMode" flag
    Public Sub New(Optional isAdminMode As Boolean = False)
        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        Me.isAdminMode = isAdminMode
    End Sub
    ' Database connection string
    Dim connectionString As String =
ConfigurationManager.ConnectionStrings("CMS.My.MySettings.CMSConnectionString").
ConnectionString
    Private originalPassword As String = ""
    Private originalConfirmPassword As String = ""

    Private Sub Register_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        ' Check if the form is opened in "admin mode"
        If isAdminMode Then
            ' Disable or hide the Login button
            btnLogin.Visible = False ' Or you could set Enabled = False to just
disable it
            lblfootermsg.Visible = False
        Else
            ' Ensure the Login button is visible when opened normally
            btnLogin.Visible = True
            lblfootermsg.Visible = True
        End If

        ' Centering the form in the middle of the parent form
        Dim x As Integer
        Dim y As Integer
        Dim r As Rectangle

        If Not Parent Is Nothing Then
            r = Parent.ClientRectangle
            x = r.Width - Me.Width + Parent.Left
            y = r.Height - Me.Height + Parent.Top
        Else
            r = Screen.PrimaryScreen.WorkingArea
            x = r.Width - Me.Width
            y = r.Height - Me.Height
        End If

        x = CInt(x / 2)
        y = CInt(y / 2)
        Me.StartPosition = FormStartPosition.Manual
        Me.Location = New Point(x, y)

        ' Mask passwords on load
        txtboxpassword.PasswordChar = "●"c
        txtboxconfirmpassword.PasswordChar = "●"c
    End Sub

    ' Event handler for the Register button click event
    Private Sub btnSignIn_Click(sender As Object, e As EventArgs) Handles
btnSignIn.Click
        ' Retrieve form input values
        Dim email As String = rtxtboxemail.Text.Trim()
        Dim password As String = txtboxpassword.Text.Trim()
        Dim confirmPassword As String = txtboxconfirmpassword.Text.Trim()
        Dim firstName As String = rtxtboxfirstname.Text.Trim()
        Dim lastName As String = rtxtboxlastname.Text.Trim()
        Dim age As String = rtxtboxage.Text.Trim()
        Dim gender As String = If(radiobtnmale.Checked, "Male",
If(radiobtnfemale.Checked, "Female", "Other"))
        Dim phoneNo As String = rtxtboxphoneno.Text.Trim()

        ' Input validation
        If Not ValidateInputs(email, password, confirmPassword, firstName,
lastName, age, phoneNo) Then
            Exit Sub
        End If

        ' Insert new user data into the database
        Try
            Using connection As New SqlConnection(connectionString)
                connection.Open()

                ' Check if email already exists
                Dim emailQuery As String = "SELECT COUNT(*) FROM Users WHERE
email = @Email"
                Using emailCommand As New SqlCommand(emailQuery, connection)
                    emailCommand.Parameters.AddWithValue("@Email", email)
                    Dim emailExists As Integer =
Convert.ToInt32(emailCommand.ExecuteScalar())

                    If emailExists > 0 Then
                        MessageBox.Show("Email already exists. Please use a
different email.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
                        HighlightField(rtxtboxemail, True)
                        Exit Sub
                    End If
                End Using

                ' Insert into Users table
                Dim insertUserQuery As String = "INSERT INTO Users (email,
password, user_type, status) VALUES (@Email, HASHBYTES('SHA2_512', @Password),
@UserType, @Status); SELECT SCOPE_IDENTITY();"
                Using insertUserCommand As New SqlCommand(insertUserQuery,
connection)
                    insertUserCommand.Parameters.AddWithValue("@Email", email)
                    insertUserCommand.Parameters.AddWithValue("@Password",
password)
                    insertUserCommand.Parameters.AddWithValue("@UserType",
"customer")
                    insertUserCommand.Parameters.AddWithValue("@Status", 1)

                    ' Get the inserted user_id
                    Dim newUserId As Integer =
Convert.ToInt32(insertUserCommand.ExecuteScalar())

                    ' Insert into User_Details table
                    Dim insertUserDetailsQuery As String = "INSERT INTO
User_Details (user_id, first_name, last_name, age, gender, phone_no) VALUES
(@UserId, @FirstName, @LastName, @Age, @Gender, @PhoneNo)"
                    Using insertUserDetailsCommand As New
SqlCommand(insertUserDetailsQuery, connection)

insertUserDetailsCommand.Parameters.AddWithValue("@UserId", newUserId)

insertUserDetailsCommand.Parameters.AddWithValue("@FirstName", firstName)

insertUserDetailsCommand.Parameters.AddWithValue("@LastName", lastName)
                        insertUserDetailsCommand.Parameters.AddWithValue("@Age",
age)

insertUserDetailsCommand.Parameters.AddWithValue("@Gender", gender)

insertUserDetailsCommand.Parameters.AddWithValue("@PhoneNo",
If(String.IsNullOrEmpty(phoneNo), "Not Provided", phoneNo))

                        insertUserDetailsCommand.ExecuteNonQuery()
                    End Using
                End Using
            End Using

            ' Registration successful
            MessageBox.Show("Registration successful!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            Me.Close()

        Catch ex As Exception
            MessageBox.Show($"An error occurred while registering:
{ex.Message}", "Database Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

' Function to validate form inputs
Private Function ValidateInputs(email As String, password As String,
confirmPassword As String, firstName As String, lastName As String, age As
String, phoneNo As String) As Boolean
    Dim isValid As Boolean = True

    If String.IsNullOrWhiteSpace(email) Then
        HighlightField(rtxtboxemail, True)
        isValid = False
    Else
        HighlightField(rtxtboxemail, False)
    End If
    If String.IsNullOrWhiteSpace(firstName) Then
        HighlightField(rtxtboxfirstname, True)
        isValid = False
    Else
        HighlightField(rtxtboxfirstname, False)
    End If

    If String.IsNullOrWhiteSpace(lastName) Then
        HighlightField(rtxtboxlastname, True)
        isValid = False
    Else
        HighlightField(rtxtboxlastname, False)
    End If

    If String.IsNullOrWhiteSpace(phoneNo) Then
        HighlightField(rtxtboxphoneno, True)
        isValid = False
    ElseIf phoneNo.Length <> 10 Then
        MessageBox.Show("Phone number must be 10 digits long.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        HighlightField(rtxtboxphoneno, True)
        isValid = False
    Else
        HighlightField(rtxtboxphoneno, False)
    End If

    If String.IsNullOrWhiteSpace(age) OrElse Not IsNumeric(age) OrElse
Convert.ToInt32(age) <= 0 OrElse Convert.ToInt32(age) >= 150 Then
        MessageBox.Show("Please enter a valid age.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        HighlightField(rtxtboxage, True)
        isValid = False
    Else
        HighlightField(rtxtboxage, False)
    End If

    ' Validate gender selection
    If Not radiobtnmale.Checked AndAlso Not radiobtnfemale.Checked Then
        MessageBox.Show("Please select a gender.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        isValid = False
    End If

    ' Password validation (based on your new requirements)
    If Not IsPasswordValid(password) Then
        MessageBox.Show("Password does not meet the required criteria.",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        HighlightField(txtboxpassword, True)
        isValid = False
    Else
        HighlightField(txtboxpassword, False)
    End If

    ' Confirm password validation
    If String.IsNullOrWhiteSpace(confirmPassword) Then
        HighlightField(txtboxconfirmpassword, True)
        isValid = False
    ElseIf password <> confirmPassword Then
        MessageBox.Show("Passwords do not match.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        HighlightField(txtboxpassword, True)
        HighlightField(txtboxconfirmpassword, True)
        isValid = False
    Else
        HighlightField(txtboxconfirmpassword, False)
    End If

    If Not IsValidEmail(email) Then
        MessageBox.Show("Please enter a valid email address.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        HighlightField(rtxtboxemail, True)
        isValid = False
    End If

    If Not isValid Then
        MessageBox.Show("Please correct the highlighted fields.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If

    Return isValid
End Function



' Function to validate email format
Private Function IsValidEmail(email As String) As Boolean
    Dim emailPattern As String = "^[^@\s]+@[^@\s]+\.[^@\s]+$"
    Return Regex.IsMatch(email, emailPattern)
End Function


' Function to highlight required fields
Private Sub HighlightField(field As Control, highlight As Boolean)
    If highlight Then
        field.BackColor = Color.FromArgb(252, 73, 73)
    Else
        field.BackColor = Color.White
    End If
End Sub

' Event handler for the Cancel button click event
Private Sub btncancel_Click(sender As Object, e As EventArgs) Handles
btncancel.Click
    Me.Close()
End Sub

' Toggle password visibility for password field
Private Sub btntogglepassword_Click(sender As Object, e As EventArgs) Handles
btnTogglePassword.Click
    TogglePasswordVisibility(txtboxpassword, btnTogglePassword,
originalPassword)
End Sub

' Toggle password visibility for confirm password field
```

```
Private Sub btntogglecpassword_Click(sender As Object, e As EventArgs) Handles
btnTogglecPassword.Click
    TogglePasswordVisibility(txtboxconfirmpassword, btnTogglecPassword,
originalConfirmPassword)
End Sub

' Function to toggle password visibility for TextBox
Private Sub TogglePasswordVisibility(passwordBox As TextBox, toggleButton As
Button, ByRef originalText As String)
    ' Store the current text in originalText if it's the first time toggling
    If String.IsNullOrEmpty(originalText) Then
        originalText = passwordBox.Text
    End If

    ' Check if the password is currently visible (not masked)
    If passwordBox.PasswordChar = ControlChars.NullChar Then
        ' Mask the password
        passwordBox.PasswordChar = "●"c
        toggleButton.BackgroundImage = My.Resources.Eye_Open
    Else
        ' Unmask the password
        passwordBox.PasswordChar = ControlChars.NullChar
        toggleButton.BackgroundImage = My.Resources.Eye_Close
    End If
End Sub

    ' Event handler for KeyPress event of rtxtboxphoneno
    Private Sub rtxtboxphoneno_KeyPress(sender As Object, e As
KeyPressEventArgs) Handles rtxtboxphoneno.KeyPress
        ' Allow control keys (backspace, delete)
        If Char.IsControl(e.KeyChar) Then
            Return
        End If

        ' Allow only numeric input
        If Not Char.IsDigit(e.KeyChar) Then
            e.Handled = True
        End If

        ' Restrict the length to 10 characters
        If rtxtboxphoneno.Text.Length >= 10 Then
            e.Handled = True
        End If
    End Sub

    'Function to valid the password
    Private Function IsPasswordValid(password As String) As Boolean
        Dim specialChars As String = "@$!%*?&"
        ' Check if the password meets all criteria:
        Return password.Length >= 8 AndAlso password.Length <= 16 AndAlso
           password.Any(Function(c) Char.IsUpper(c)) AndAlso
           password.Any(Function(c) Char.IsLower(c)) AndAlso
           password.Any(Function(c) Char.IsDigit(c)) AndAlso
           password.Any(Function(c) specialChars.Contains(c))
    End Function


    ' Event handler for password input text change
    Private Sub txtboxpassword_TextChanged(sender As Object, e As EventArgs)
Handles txtboxpassword.TextChanged
        Dim password As String = txtboxpassword.Text

        ' Validate password length
        If password.Length >= 8 AndAlso password.Length <= 16 Then
            lblLength.ForeColor = Color.Green
        Else
            lblLength.ForeColor = Color.Red
        End If

        ' Validate at least one uppercase letter
        If password.Any(Function(c) Char.IsUpper(c)) Then
            lblUppercase.ForeColor = Color.Green
        Else
            lblUppercase.ForeColor = Color.Red
        End If

        ' Validate at least one lowercase letter
        If password.Any(Function(c) Char.IsLower(c)) Then
            lblLowercase.ForeColor = Color.Green
        Else
            lblLowercase.ForeColor = Color.Red
        End If

        ' Validate at least one digit
        If password.Any(Function(c) Char.IsDigit(c)) Then
            lblNumber.ForeColor = Color.Green
        Else
            lblNumber.ForeColor = Color.Red
        End If

        ' Validate at least one special character
        Dim specialChars As String = "@$!%*?&"
        If password.Any(Function(c) specialChars.Contains(c)) Then
            lblSpecialChar.ForeColor = Color.Green
        Else
            lblSpecialChar.ForeColor = Color.Red
        End If
    End Sub

    'Event handler for bthlogin to redirect user to login page
    Private Sub btnlogin_Click(sender As Object, e As EventArgs) Handles
btnLogin.Click
        ' Create a new instance of the Login_Page form.
        Dim login_page As New Login_Page()

        ' Show the Login_Page form as a modal dialog.
        ' This will block interaction with the current form until the login form
is closed.
        login_page.ShowDialog()

        ' Close the current form (the registration form) after showing the login
page.
        Me.Close()
    End Sub

    ' MouseEnter and MouseLeave events for the Login button
    Private Sub btnLogin_MouseEnter(sender As Object, e As EventArgs) Handles
btnSignIn.MouseEnter
        btnSignIn.BackColor = Color.FromArgb(0, 175, 99) ' Darker green for
hover effect
    End Sub

    Private Sub btnLogin_MouseLeave(sender As Object, e As EventArgs) Handles
btnSignIn.MouseLeave
        btnSignIn.BackColor = Color.FromArgb(0, 191, 99) ' Original color
    End Sub

    ' MouseEnter and MouseLeave events for the Cancel button
    Private Sub btncancel_MouseEnter(sender As Object, e As EventArgs) Handles
btncancel.MouseEnter
        btncancel.BackColor = Color.FromArgb(0, 191, 99) ' Highlight text on
hover
        btncancel.ForeColor = Color.White
    End Sub

    Private Sub btncancel_MouseLeave(sender As Object, e As EventArgs) Handles
btncancel.MouseLeave
        btncancel.BackColor = Color.White
        btncancel.ForeColor = Color.FromArgb(0, 191, 99) ' Revert text color
    End Sub

End Class
```

4.4 Home Page

```
Public Class Home_Page

    Private userId As Integer

    ' Constructor to accept user ID
    Public Sub New(Optional userId As Integer = 1011)
        InitializeComponent()
        Me.userId = userId
        ' You can now use the userId to track the customer
    End Sub

    'Default Constructor
    Public Sub New()
        InitializeComponent()
        Me.userId = 1011
        ' You can now use the userId to track the customer
    End Sub

    ' Mouse Enter/Leave events to change colors for labels and buttons for
visual appeal
    Private Sub LblHome_MouseEnter(sender As Object, e As EventArgs) Handles
lblHome.MouseEnter
        ChangeLabelColor(lblHome, Color.FromArgb(0, 191, 99))
    End Sub

    Private Sub LblHome_MouseLeave(sender As Object, e As EventArgs) Handles
lblHome.MouseLeave
        ChangeLabelColor(lblHome, Color.Black)
    End Sub

    Private Sub LblMenu_MouseEnter(sender As Object, e As EventArgs) Handles
lblMenu.MouseEnter
        ChangeLabelColor(lblMenu, Color.FromArgb(0, 191, 99))
    End Sub

    Private Sub LblMenu_MouseLeave(sender As Object, e As EventArgs) Handles
lblMenu.MouseLeave
        ChangeLabelColor(lblMenu, Color.Black)
    End Sub

    Private Sub LblAboutUs_MouseEnter(sender As Object, e As EventArgs) Handles
lblAboutUs.MouseEnter
        ChangeLabelColor(lblAboutUs, Color.FromArgb(0, 191, 99))
    End Sub

    Private Sub LblAboutUs_MouseLeave(sender As Object, e As EventArgs) Handles
lblAboutUs.MouseLeave
        ChangeLabelColor(lblAboutUs, Color.Black)
    End Sub

    Private Sub LblContact_MouseEnter(sender As Object, e As EventArgs) Handles
lblContact.MouseEnter
        ChangeLabelColor(lblContact, Color.FromArgb(0, 191, 99))
    End Sub
    Private Sub LblContact_MouseLeave(sender As Object, e As EventArgs) Handles
lblContact.MouseLeave
        ChangeLabelColor(lblContact, Color.Black)
    End Sub

    ' Social media label color changes for Google, Facebook, etc.
    Private Sub LblGoogle_MouseEnter(sender As Object, e As EventArgs) Handles
lblGoogle.MouseEnter
        ChangeLabelColor(lblGoogle, Color.Black)
    End Sub

    Private Sub LblGoogle_MouseLeave(sender As Object, e As EventArgs) Handles
lblGoogle.MouseLeave
        ChangeLabelColor(lblGoogle, Color.DimGray)
    End Sub

    Private Sub LblFacebook_MouseEnter(sender As Object, e As EventArgs) Handles
lblFacebook.MouseEnter
        ChangeLabelColor(lblFacebook, Color.Blue)
    End Sub

    Private Sub LblFacebook_MouseLeave(sender As Object, e As EventArgs) Handles
lblFacebook.MouseLeave
        ChangeLabelColor(lblFacebook, Color.DimGray)
    End Sub

    Private Sub LblYoutube_MouseEnter(sender As Object, e As EventArgs) Handles
lblYoutube.MouseEnter
        ChangeLabelColor(lblYoutube, Color.Red)
    End Sub

    Private Sub LblYoutube_MouseLeave(sender As Object, e As EventArgs) Handles
lblYoutube.MouseLeave
        ChangeLabelColor(lblYoutube, Color.DimGray)
    End Sub

    Private Sub LblInstagram_MouseEnter(sender As Object, e As EventArgs)
Handles lblInstagram.MouseEnter
        ChangeLabelColor(lblInstagram, Color.Red)
    End Sub

    Private Sub LblInstagram_MouseLeave(sender As Object, e As EventArgs)
Handles lblInstagram.MouseLeave
        ChangeLabelColor(lblInstagram, Color.DimGray)
    End Sub

    ' Changes the background color of the Sign In button when hovered
    Private Sub BtnSignIn_MouseEnter(sender As Object, e As EventArgs) Handles
btnLogin.MouseEnter
        btnLogin.BackColor = Color.FromArgb(0, 175, 99) ' Darker shade for hover
        btnLogin.ForeColor = Color.White
    End Sub

    Private Sub btnSignIn_MouseLeave(sender As Object, e As EventArgs) Handles
btnLogin.MouseLeave
        btnLogin.BackColor = Color.FromArgb(0, 191, 99) ' Original color
        btnLogin.ForeColor = Color.White
    End Sub

    Private Sub BtnSignUp_MouseEnter(sender As Object, e As EventArgs) Handles
btnSignUp.MouseEnter
        btnSignUp.BackColor = Color.FromArgb(0, 175, 99) ' shade for hover
        btnSignUp.ForeColor = Color.White
    End Sub

    Private Sub btnSignUp_MouseLeave(sender As Object, e As EventArgs) Handles
btnSignUp.MouseLeave
        btnSignUp.BackColor = Color.White
        btnSignUp.ForeColor = Color.FromArgb(0, 191, 99) ' Original color
    End Sub

    Private Sub BtnOrderNow_MouseEnter(sender As Object, e As EventArgs) Handles
btnOrderNow.MouseEnter
        btnOrderNow.BackColor = Color.FromArgb(0, 175, 99) ' Darker shade for
hover
        btnOrderNow.ForeColor = Color.White
    End Sub

    Private Sub btnOrderNow_MouseLeave(sender As Object, e As EventArgs) Handles
btnOrderNow.MouseLeave
        btnOrderNow.BackColor = Color.FromArgb(0, 191, 99) ' Original color
        btnOrderNow.ForeColor = Color.White
    End Sub

    Private Sub BtnViewPlate_MouseEnter(sender As Object, e As EventArgs)
Handles btnViewPlate.MouseEnter
        btnViewPlate.BackColor = Color.FromArgb(0, 175, 99) ' shade for hover
        btnViewPlate.ForeColor = Color.White
    End Sub

    Private Sub btnViewPlate_MouseLeave(sender As Object, e As EventArgs)
Handles btnViewPlate.MouseLeave
        btnViewPlate.BackColor = Color.White
        btnViewPlate.ForeColor = Color.FromArgb(0, 191, 99) ' Original color
    End Sub

    ' Click event for navigating to the Menu_Page and passing userId
    Private Sub LblMenu_Click(sender As Object, e As EventArgs) Handles
lblMenu.Click
        Try
            Dim menuPage As New Menu_Page(userId)
            menuPage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the menu page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Click event for navigating to Contact_Us_Page
    Private Sub lblContact_Click(sender As Object, e As EventArgs) Handles
lblContact.Click
        Try
            Dim contactUsPage As New Contact_Us_Page
            contactUsPage.Show()
        Catch ex As Exception
```

```
            MessageBox.Show("An error occurred while opening the contact page: "
& ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Click event for navigating to About_Us_Page
    Private Sub lblAboutUs_Click(sender As Object, e As EventArgs) Handles
lblAboutUs.Click
        Try
            Dim aboutUsPage As New About_Us_Page
            aboutUsPage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the about page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Click event for the Sign Up button
    Private Sub btnSignUp_Click(sender As Object, e As EventArgs) Handles
btnSignUp.Click
        Try
            Dim signUpPage As New Register()
            signUpPage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the sign-up page: "
& ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Click event for the Login button
    Private Sub btnLogin_Click(sender As Object, e As EventArgs) Handles
btnLogin.Click
        Try
            Dim loginPage As New Login_Page()
            loginPage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the login page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Event to navigate to Plate_Page when the "View Plate" button is clicked
    Private Sub btnViewPlate_Click(sender As Object, e As EventArgs) Handles
btnViewPlate.Click
        Try
            Dim platePage As New Plate_Page(userId)
            platePage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the plate page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Event to navigate to Order_Page when the "Order Now" button is clicked
    Private Sub btnOrderNow_Click(sender As Object, e As EventArgs) Handles
btnOrderNow.Click
        Try
            Dim billPage As New Billing_Page(userId)
            billPage.Show()
        Catch ex As Exception
            MessageBox.Show("An error occurred while opening the order page: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Helper function to change label color dynamically
    Private Sub ChangeLabelColor(label As Label, color As Color)
        label.ForeColor = color
    End Sub

    ' Helper function to change buttin color dynamically
    Private Sub ChangeButtonColor(btn As Label, color As Color)
        btn.ForeColor = color
    End Sub
    ' Event to handle the Logout button click - close all open forms
    Private Sub btnLogout_Click(sender As Object, e As EventArgs) Handles
btnLogout.Click
        Try
            For Each frm As Form In Application.OpenForms.Cast(Of Form).ToList()
                frm.Close()
            Next
        Catch ex As Exception
            MessageBox.Show("An error occurred while logging out: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    Private Sub Home_Page_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

    End Sub
End Class
```

4.5 Admin Page

```
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms

Public Class Admin_Page
    ' Track the sidebar state (collapsed or expanded)
    Private isCollapsed As Boolean = False
    Private WithEvents sidebarTransitionTimer As New Timer With {.Interval = 18}
    Private targetWidth As Integer ' Width to transition to

    ' Constants for sidebar dimensions and colors
    Private Const EXPAND_WIDTH As Integer = 200
    Private Const COLLAPSED_WIDTH As Integer = 80
    Private ReadOnly DefaultBackClr As Color = Color.Black
    Private ReadOnly PanelHoverColor As Color = Color.WhiteSmoke
    Private ReadOnly ButtonHoverColor As Color = PanelHoverColor
    Private ReadOnly ButtonTextHoverColor As Color = Color.Black

    ' Radius for rounded corners
    Private rad As Integer = 50

    ' Dictionaries to store original colors of UI elements
    Private originalPanelColors As New Dictionary(Of Panel, Color)
    Private originalButtonColors As New Dictionary(Of Button, Color)
    Private originalButtonTextColors As New Dictionary(Of Button, Color)
    Private originalPictureBoxColors As New Dictionary(Of PictureBox, Color)

    ' Admin_Page Load event
    Private Sub Admin_Page_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        ' Apply rounded corners and default dashboard on load
        SetRoundedCorners()
        Try
            ' Load the default control (Dashboard) on startup
            LoadForm(New Dashboard_Page())

            ' Attach hover and click events for sidebar panels
            AddSidebarHoverAndClickEvents()

        Catch ex As Exception
            MessageBox.Show("An error occurred during page load: " & ex.Message)
        End Try
    End Sub

    ' Toggle sidebar (expand or collapse) on double-click
    Private Sub pnlSidebar_DoubleClick(sender As Object, e As EventArgs) Handles
pnlSidebar.DoubleClick
        Try
            If isCollapsed Then
                ' Expand the sidebar
                StartSidebarTransition(EXPAND_WIDTH)
                ToggleButtonText(True)
            Else
                ' Collapse the sidebar
                StartSidebarTransition(COLLAPSED_WIDTH)
                ToggleButtonText(False)
            End If
            isCollapsed = Not isCollapsed
        Catch ex As Exception
            MessageBox.Show("An error occurred while toggling sidebar: " &
ex.Message)
        End Try
    End Sub

    ' Start the sidebar transition animation
    Private Sub StartSidebarTransition(target As Integer)
        targetWidth = target
        sidebarTransitionTimer.Start()
    End Sub

    ' Smooth sidebar transition using Timer
    Private Sub sidebarTransitionTimer_Tick(sender As Object, e As EventArgs)
Handles sidebarTransitionTimer.Tick
        Try
            If pnlSidebar.Width <> targetWidth Then
                ' Smoothly adjust the width step by step
                Dim stepSize As Integer = Math.Sign(targetWidth -
pnlSidebar.Width) * 10
                pnlSidebar.Width += stepSize

                ' Stop timer once the target width is reached
                If Math.Abs(pnlSidebar.Width - targetWidth) < Math.Abs(stepSize)
Then
                    pnlSidebar.Width = targetWidth
                    sidebarTransitionTimer.Stop()
                End If
            Else
                sidebarTransitionTimer.Stop()
            End If

            ' Reapply rounded corners after resizing
            SetRoundedCorners()

        Catch ex As Exception
            MessageBox.Show("An error occurred during sidebar transition: " &
ex.Message)
        End Try
    End Sub

    ' Toggle button text visibility based on sidebar state
    Private Sub ToggleButtonText(showText As Boolean)
        For Each pnl As Panel In pnlSidebar.Controls.OfType(Of Panel)()
            For Each ctrl In pnl.Controls
                If TypeOf ctrl Is Button Then
                    Dim button = DirectCast(ctrl, Button)
                    If showText Then
                        button.Text = button.Tag.ToString() ' Restore button
text
                    Else
                        button.Tag = button.Text ' Store button text in Tag
                        button.Text = "" ' Hide button text
                    End If
                End If
            Next
        Next
    End Sub

    ' Load the correct form based on the clicked panel
    Private Sub LoadPanelForm(pnl As Panel)
        Try
            Select Case pnl.Name
                Case "pnlDashboard"
                    LoadForm(New Dashboard_Page())
                Case "pnlAnalytics"
                    LoadForm(New Analytics_Page())
                Case "pnlPayments"
                    LoadForm(New Payments_Page())
                Case "pnlUsers"
                    LoadForm(New Users_Page())
                Case "pnlProducts"
                    LoadForm(New Products_Page())
                Case "pnlSuppliers"
                    LoadForm(New Suppliers_Page())
                Case "pnlInventory"
                    LoadForm(New Inventory_Page())
                Case "pnlOrders"
                    LoadForm(New Orders_Page())
                Case "pnlMenu"
                    LoadForm(New MenuManage_Page())
            End Select
        Catch ex As Exception
            MessageBox.Show("An error occurred while loading panel form: " &
ex.Message)
        End Try
    End Sub

    ' Load the form inside the main panel
    Private Sub LoadForm(form As Form)
        Try
            pnlMain.Controls.Clear() ' Clear existing controls
            form.FormBorderStyle = FormBorderStyle.None ' Remove form border
            form.TopLevel = False ' Make it a child control
            form.Dock = DockStyle.Fill ' Fill the panel
            pnlMain.Controls.Add(form)
            form.Show() ' Display the form
        Catch ex As Exception
            MessageBox.Show("An error occurred while loading form: " &
ex.Message)
        End Try
    End Sub

    ' Attach hover and click events for all relevant sidebar panels and controls
    Private Sub AddSidebarHoverAndClickEvents()
        ' Add hover effects and click events for each panel, picture box, and
button
        AddHoverEffectAndClick(pnlDashboard, PicboxDashboard, btnDashboard)
        AddHoverEffectAndClick(pnlAnalytics, PicboxAnalytics, btnAnalytics)
        AddHoverEffectAndClick(pnlPayments, picboxPayments, btnPayments)
        AddHoverEffectAndClick(pnlUsers, picboxUsers, btnUsers)
        AddHoverEffectAndClick(pnlProducts, picboxProducts, btnProducts)
        AddHoverEffectAndClick(pnlSuppliers, picboxSuppliers, btnSuppliers)
        AddHoverEffectAndClick(pnlInventory, picboxInventory, btnInventory)
        AddHoverEffectAndClick(pnlOrders, picboxOrders, btnOrders)
        AddHoverEffectAndClick(pnlMenu, picboxMenu, btnMenu)
    End Sub

    ' Add hover effects and click events for panels, picture boxes, and buttons
    Private Sub AddHoverEffectAndClick(pnl As Panel, picbox As PictureBox, btn
As Button)
        ' Store original background and text colors if not already stored
        If Not originalPanelColors.ContainsKey(pnl) Then
            originalPanelColors(pnl) = pnl.BackColor
        End If
        If Not originalPictureBoxColors.ContainsKey(picbox) Then
            originalPictureBoxColors(picbox) = picbox.BackColor
        End If
        If Not originalButtonColors.ContainsKey(btn) Then
            originalButtonColors(btn) = btn.BackColor
        End If
        If Not originalButtonTextColors.ContainsKey(btn) Then
            originalButtonTextColors(btn) = btn.ForeColor
        End If

        ' Attach hover and leave events for both picture boxes and buttons
        AddHandler picbox.MouseEnter, Sub(sender, e) ApplyHoverEffect(pnl,
picbox, btn)
        AddHandler btn.MouseEnter, Sub(sender, e) ApplyHoverEffect(pnl, picbox,
btn)

        AddHandler picbox.MouseLeave, Sub(sender, e) RevertHoverEffect(pnl,
picbox, btn)
        AddHandler btn.MouseLeave, Sub(sender, e) RevertHoverEffect(pnl, picbox,
btn)

        ' Attach click events
        AddHandler picbox.Click, Sub(sender, e) HandleClickEffect(picbox, pnl)
        AddHandler btn.Click, Sub(sender, e) HandleClickEffect(btn, pnl)
    End Sub

    ' Apply hover effect for buttons and picture boxes
    Private Sub ApplyHoverEffect(pnl As Panel, picbox As PictureBox, btn As
Button)
        pnl.BackColor = PanelHoverColor
        picbox.BackColor = PanelHoverColor
        btn.BackColor = ButtonHoverColor
        btn.ForeColor = ButtonTextHoverColor
    End Sub

    ' Revert hover effect on panel, picture box, and button
    Private Sub RevertHoverEffect(pnl As Panel, picbox As PictureBox, btn As
Button)
        pnl.BackColor = originalPanelColors(pnl)
        picbox.BackColor = originalPictureBoxColors(picbox)
        btn.BackColor = originalButtonColors(btn)
        btn.ForeColor = originalButtonTextColors(btn)
    End Sub

    ' Handle click effect and load the corresponding user control
    Private Sub HandleClickEffect(ctrl As Control, pnl As Panel)
        Try
            ApplyHoverEffect(pnl, GetAssociatedPictureBox(pnl),
GetAssociatedButton(pnl))

            ' Temporary click effect timer
            Using clickFeedbackTimer As New Timer With {.Interval = 200}
                AddHandler clickFeedbackTimer.Tick, Sub(sender, e)
                                                        RevertHoverEffect(pnl,
GetAssociatedPictureBox(pnl), GetAssociatedButton(pnl))

clickFeedbackTimer.Stop()
                                                    End Sub
                clickFeedbackTimer.Start()
            End Using

            ' Load the corresponding form based on the clicked panel
            LoadPanelForm(pnl)

        Catch ex As Exception
            MessageBox.Show("An error occurred while handling click effect: " &
ex.Message)
        End Try
    End Sub

    ' Helper function to get the associated PictureBox from a panel
    Private Function GetAssociatedPictureBox(pnl As Panel) As PictureBox
        Return pnl.Controls.OfType(Of PictureBox)().FirstOrDefault()
    End Function

    ' Helper function to get the associated Button from a panel
    Private Function GetAssociatedButton(pnl As Panel) As Button
        Return pnl.Controls.OfType(Of Button)().FirstOrDefault()
    End Function

    ' Function to create a rounded rectangle for custom painting
    Private Function CreateRoundedRectangle(rect As Rectangle, radius As
Integer) As GraphicsPath
        Dim path As New GraphicsPath()
        path.AddArc(rect.Left, rect.Top, radius, radius, 180, 90)
        path.AddArc(rect.Right - radius, rect.Top, radius, radius, 270, 90)
        path.AddArc(rect.Right - radius, rect.Bottom - radius, radius, radius,
0, 90)
        path.AddArc(rect.Left, rect.Bottom - radius, radius, radius, 90, 90)
        path.CloseAllFigures()
        Return path
    End Function

    ' Apply rounded corners to the sidebar panel
    Private Sub SetRoundedCorners()
        pnlSidebar.Region = New
Region(CreateRoundedRectangle(pnlSidebar.ClientRectangle, rad))
    End Sub

    ' Control hover effects (both for buttons and picture boxes)
    Private Sub Control_MouseEnter(sender As Object, e As EventArgs)
        Dim control = DirectCast(sender, Control)
        If control IsNot Nothing Then
```

```
            ApplyHoverEffect(DirectCast(control.Parent, Panel),
GetAssociatedPictureBox(control.Parent), GetAssociatedButton(control.Parent))
        End If
    End Sub

    Private Sub Control_MouseLeave(sender As Object, e As EventArgs)
        Dim control = DirectCast(sender, Control)
        If control IsNot Nothing Then
            RevertHoverEffect(DirectCast(control.Parent, Panel),
GetAssociatedPictureBox(control.Parent), GetAssociatedButton(control.Parent))
        End If
    End Sub

    ' Event to handle the Logout button click - close all open forms
    Private Sub btnLogout_Click(sender As Object, e As EventArgs) Handles
btnLogout.Click
        Try
            For Each frm As Form In Application.OpenForms.Cast(Of Form).ToList()
                frm.Close()
            Next
        Catch ex As Exception
            MessageBox.Show("An error occurred while logging out: " &
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

End Class
```

Chapter 5: Implementation and Testing

5.1 Implementation

The implementation of the **Canteen Management System (CMS)** followed an agile development approach, which allowed for iterative development, frequent testing, and continuous improvements. The project was divided into manageable sprints, each focusing on key aspects of the system such as user interface design, order and inventory management, payment processing, and data visualization. Key tasks included setting up the database, integrating front-end and back-end systems, and ensuring real-time synchronization between customer orders and inventory levels. Each phase of the project was rigorously tested to ensure that the system performed reliably under various use cases.

5.1.1 Tools Used

The **Canteen Management System** employed a range of tools and technologies to ensure a smooth and efficient implementation. Below is a description of these tools, including their specific applications within the project.

- **Visual Basic and .NET Framework**:
  Visual Basic, paired with the .NET framework, formed the backbone of the system's development. The strong integration between front-end and back-end services within the .NET environment enabled us to efficiently build the core features of the system. Event-driven programming in Visual Basic facilitated the creation of a responsive and user-friendly interface, ensuring seamless user interactions, while the .NET libraries supported functionalities like instant billing and secure payment processing.
- **Microsoft SQL Server**:
  Microsoft SQL Server was selected for managing the database of the CMS. It played a vital role in storing and managing data such as customer orders, inventory levels, and transaction records. SQL Server's real-time data processing capabilities ensured that the system maintained up-to-date information on stock levels, which helped admins manage supplies effectively.
- **LiveCharts**:
  LiveCharts was integrated into the CMS to provide interactive and real-time data visualizations within the admin dashboard. LiveCharts offered rich, dynamic charts that displayed key analytics, such as sales trends, popular items, inventory levels, and customer order patterns. These visual representations made it easier for canteen managers to track performance at a glance and make data-driven decisions. By using LiveCharts, the system could provide admins with instant feedback through pie charts, bar graphs, and line charts, reflecting sales performance and inventory fluctuations in real-time.
- **Regex (Regular Expressions)**:
  Regular expressions (Regex) were used throughout the CMS for data validation and pattern matching, particularly in user input fields. For example, Regex was employed to validate customer email addresses, phone numbers, and payment information, ensuring that the data entered met the required format before submission. This helped

to minimize errors and maintain data integrity across the system. Regex also facilitated efficient search functionality within the system, allowing admins to quickly locate specific orders or inventory items based on keyword patterns.

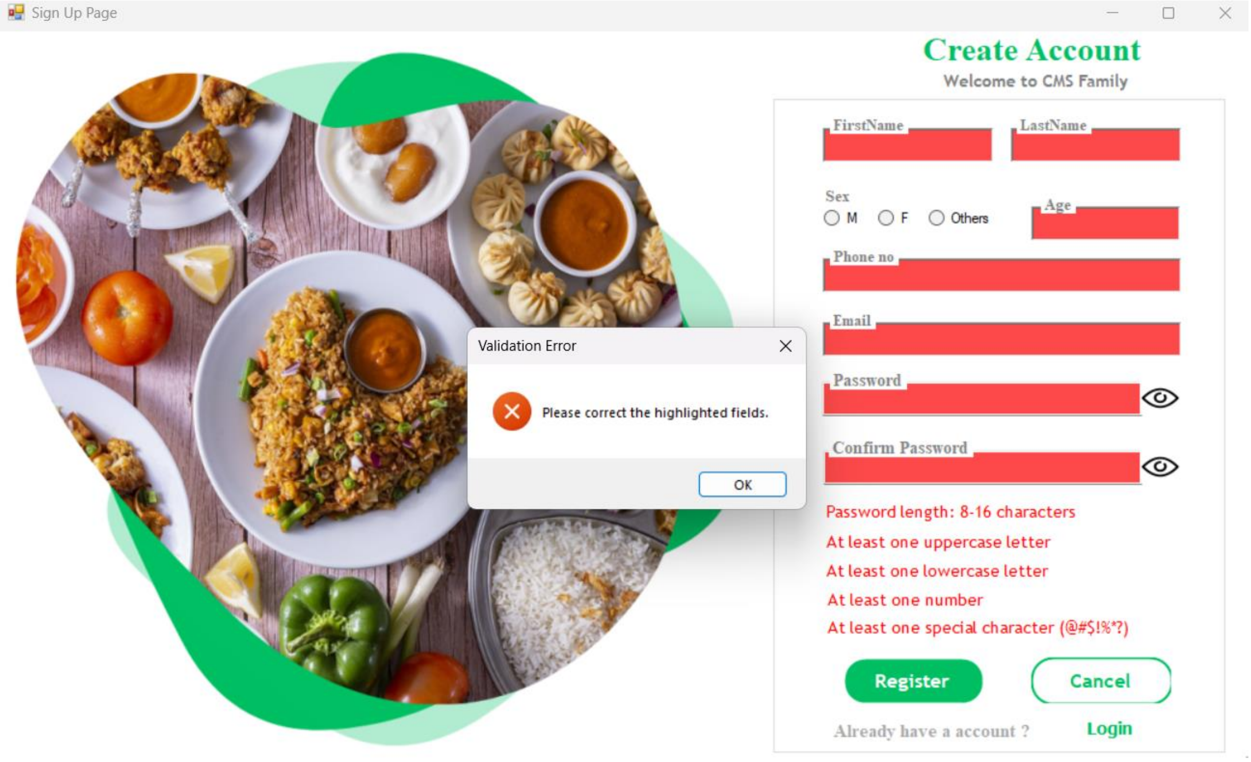
- **Diagrams.helpful.dev**:
  This tool was used to create and modify system diagrams during the planning and design phases. It helped the team visualize the architecture of the CMS, including the flow of data between customer interfaces, the database, and the admin dashboard. The diagrams were crucial in ensuring that all components of the system were correctly integrated.

By utilizing these tools and technologies, the **Canteen Management System** was successfully implemented with a focus on performance, usability, and real-time data management. Each tool played a unique role in delivering a cohesive, feature-rich system that met both customer and administrative needs, from order placement to billing and inventory management.

4.1.2 Description/Listing of major classes/methods

Module	Description
Login Page	Authenticates users and directs them to the appropriate dashboard (Admin/User).
Register Page	Allows new users to create accounts and validates their inputs using Regex.
Home Page	Provides quick navigation to the menu, cart (plate), and other relevant pages.
Menu Page	Displays available food items with details, allowing users to add items to cart.
Plate Page	Shows selected items in the cart and allows quantity adjustment or item removal.
Billing Page	Generates the final invoice and processes payments for the selected items.
Thank You Page	Displays a confirmation message after a successful purchase.
Online Payment Page	Integrates third-party payment gateway for secure online transactions.
Contact Us Page	Provides contact information and a feedback form for customer inquiries.
About Us Page	Shares details about the canteen and the CMS project.
Admin Dashboard	Centralized hub for admins to view analytics, manage orders, and monitor sales.
Analytics Page	Displays real-time charts and graphs using LiveCharts to show performance data.
Orders Page	Allows admins to view, update, and track customer orders.
Payment Page	Shows payment records and tracks transactions made by customers.
Users Page	Manages user accounts, including creation, updates, and deletion.
Suppliers Page	Manages suppliers and tracks incoming inventory from vendors.
Products Page	Manages the items offered in the menu, including adding, updating, and deleting.
Inventory Page	Tracks inventory levels in real-time and sends alerts for low stock.

4.2 Testing



Validation Error

Please correct the highlighted fields.

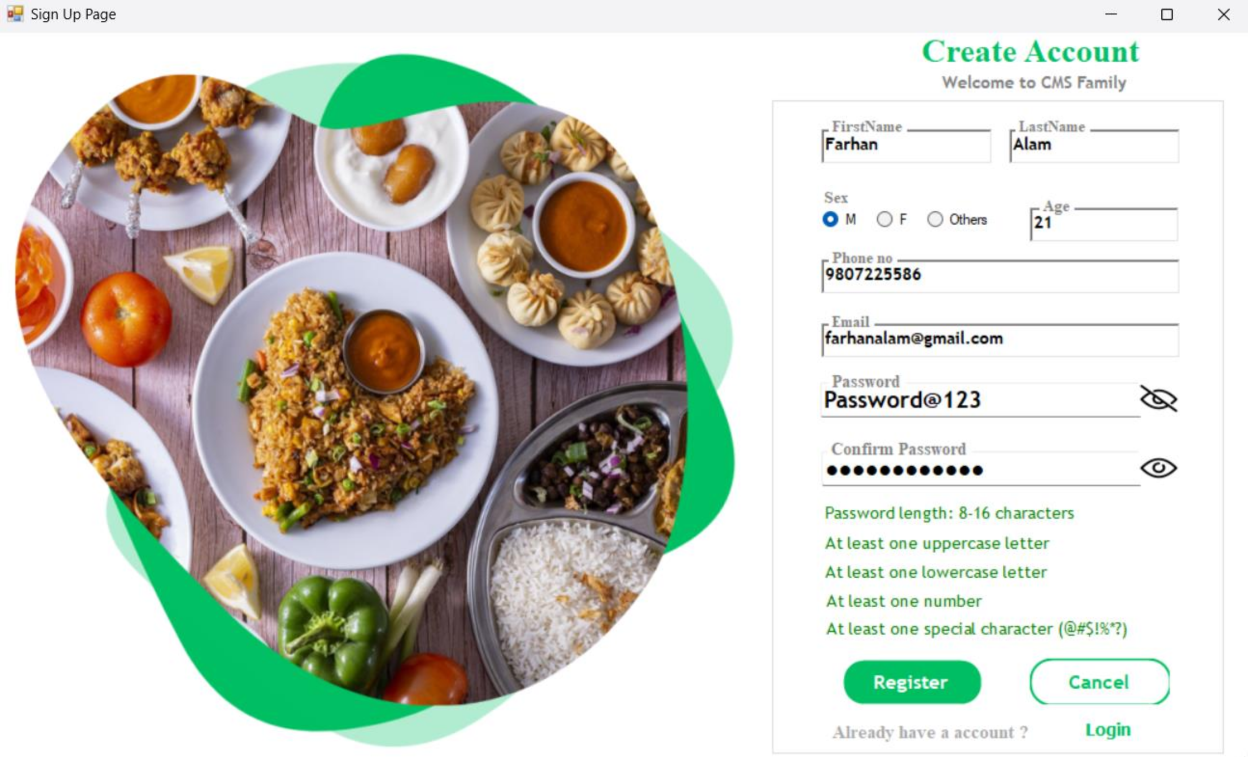
Password length: 8-16 characters

At least one uppercase letter

At least one lowercase letter

At least one number

At least one special character (@#$!%*?)



FirstName: Farhan

LastName: Alam

Sex: ● M ○ F ○ Others

Age: 21

Phone no: 9807225586

Email: farhanalam@gmail.com

Password: Password@123

Confirm Password: ●●●●●●●●●●●●

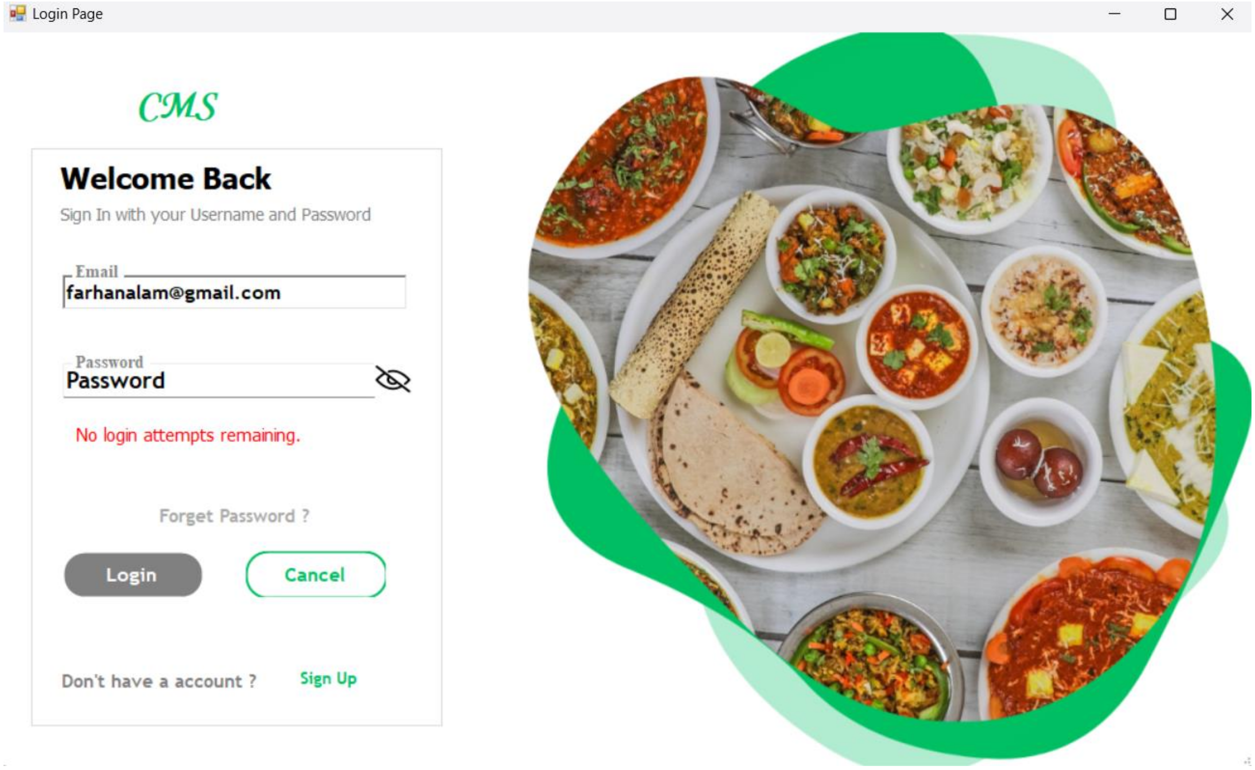
Password length: 8-16 characters

At least one uppercase letter

At least one lowercase letter

At least one number

At least one special character (@#$!%*?)



Login Page

CMS

Welcome Back

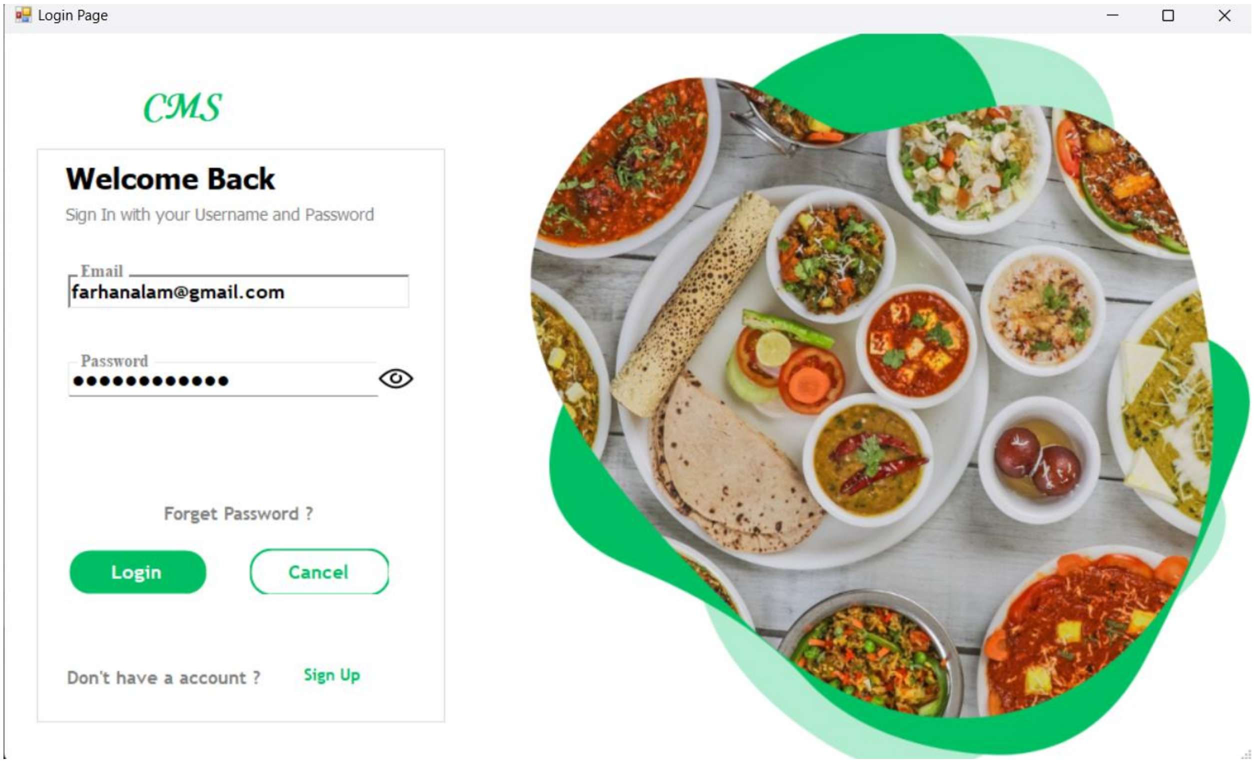
Sign In with your Username and Password

Email: farhanalam@gmail.com

Password: Password

No login attempts remaining.

Don't have a account ?



Login Page

CMS

Welcome Back

Sign In with your Username and Password

Email: farhanalam@gmail.com

Password: ●●●●●●●●●●●●

Don't have a account ?

Chapter 6: Conclusion and Recommendation

6.1 Conclusion:

The **Canteen Management System (CMS)** project successfully achieved its primary goal of automating and streamlining the operations of a campus canteen. The project focused on addressing key challenges such as long queues, manual order management, and inefficient payment processes. By implementing features like digital ordering, instant bill printing, real-time inventory tracking, and cashless payment integration, the system significantly improved both customer experience and operational efficiency.

The project contributed to modernizing the canteen's workflow, reducing human errors, and enabling faster service delivery. It met its initial aims by providing a user-friendly interface for customers and a comprehensive dashboard for administrators, which included sales analytics, order management, and inventory control. The integration of tools such as LiveCharts for data visualization and regular expressions (Regex) for input validation enhanced the system's functionality and security.

Although the project largely met its objectives, minor limitations were encountered, such as potential scalability issues for larger environments and reliance on stable internet connectivity for full functionality. Nonetheless, the major findings showed that automation improved overall efficiency, customer satisfaction, and data accuracy within the canteen environment. The CMS has laid the groundwork for further enhancements, with the potential for future upgrades to support larger-scale operations.

6.2 Recommendations

Based on the successful implementation of the Canteen Management System (CMS) and its impact on operational efficiency and customer satisfaction, several recommendations can be made to further enhance the system and its offerings:

User Experience Enhancements: Continue to prioritize the user interface (UI) design by conducting user testing sessions to gather feedback. This can help identify pain points and ensure that the system is intuitive and easy to navigate for both customers and staff.

Scalability Solutions: To address potential scalability issues, consider exploring cloud-based solutions that can handle increased user loads. This will ensure that the CMS can effectively support additional clients and larger canteen environments without compromising performance.

Internet Connectivity Options: Investigate alternative methods for ensuring reliable internet connectivity. This could include offline functionality for essential features or the use of local servers to minimize dependency on internet access.

Feature Expansion: In alignment with future plans, prioritize the integration of an in-app e-wallet and additional online payment options. This will enhance convenience for users and streamline the payment process.

Client Acquisition Strategy: Develop a strategic marketing plan to secure the first 50 clients. Focus on building partnerships with educational institutions and other organizations to showcase the benefits of the CMS and encourage adoption.

Regular Updates and Support: Establish a schedule for regular system updates that include new features, security enhancements, and performance improvements. Additionally, create a dedicated support team to assist clients with any issues they may encounter.

Data Security Measures: Implement robust data security protocols to protect user information and ensure compliance with relevant regulations. This is crucial in maintaining trust and confidence among users.

Feedback Mechanism: Introduce a formal feedback mechanism for users to share their experiences and suggestions for improvement. This can guide future developments and ensure the system continues to meet the evolving needs of its users.

By following these recommendations, the Canteen Management System can build on its initial success and adapt to the growing demands of the market, ultimately leading to enhanced functionality, improved customer satisfaction, and increased operational efficiency.

References / Bibliography

i. Reddy, GS 2023, "Canteen Management System," *IJERT*, https://doi.org/10.17577/IJERTV12IS040301.

ii. *Project report on Canteen Management System | Academic Projects*, https://www.freeprojectz.com/project-report/6811.

iii. *Edit diagram*, https://diagrams.helpful.dev/