**ORDER MANAGEMENT SYSTEM FOR KAPE KALINAW**

A Design Document Presented to the

Faculty of Datamex College of Saint Adeline, Inc

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Information Technology

By:

Lorenzana,Joshua

**INTRODUCTION**

The Kape Kalinaw Order Management System (OMS) is a desktop-based application created in VB.NET 2010 to improve the everyday operations of coffee shops. The main purpose of the system is to take order handling faster and easier, to minimize mistakes caused by manual recording, and to provide real time updates on sales and orders. With this system, the café can serve customers more efficiently while keeping accurate and reliable records for business management.

Many small and medium coffee shops experience challenges such as mis recorded orders, difficulties in tracking revenue, manual calculation errors, and the lack of clear sales reports. The Kape Kalinaw OMS is designed to solve these problems by offering a centralized and automated solution. The system makes sure that every order is recorded correctly, payments are computed automatically, and all transactions are stored securely in the database. This not only saves time for the staff but also helps the management to avoid problems with money records.

The system provides role-based access for two types of users: Admin and Cashier. Admin users have full control of the system, including adding, updating, and deleting products, adjusting prices, and monitoring the dashboard with real-time sales information. They can also review the order history to check records and correct mistakes if necessary. Cashier users, on the other hand, are focused on customer transactions. They can select products, choose the right size and type, add them to the customer’s order list, process the payment, and automatically compute the customer’s change. This separation of roles ensures smooth workflow, protects sensitive business data, and prevents unauthorized actions.

Another important feature of the OMS is the dashboard and reporting tools. The dashboard displays real-time data such as the total number of orders, total revenue, products available, and the number of users. It also includes detailed transaction history with information like order date, time, product name, size, quantity, and payment details. Additionally, the system provides a chart to show revenue trends for the last seven days, which helps the management understand business performance at a glance. These reports are valuable for making data-driven decisions, identifying best-selling products, and planning future improvements for the café.

**SYSTEM ARCHITECTURE**

The Order Management System for Kape Kalinaw is built using a client-server architecture. The client application is a Windows desktop program developed in VB.NET, which provides the user interface for both Admin and Cashier users. Through this interface, users can manage orders, view products, process payments, and see real-time dashboard updates. The client application also validates inputs and applies business rules to ensure that all operations are accurate and secure.

**High-Level Components and Interactions**

* **Database:** Stores and manages all essential information, including users, products, orders, and order history. The business logic layer communicates with the database to save, update, and retrieve data in real time, ensuring accurate records for transactions and reports.
* **User Interface:** Provides the forms and screens for Admin and Cashier users. Admins can manage products, view dashboards, and handle order history, while Cashiers can select products, record sales, update quantities, and process payments. This layer also ensures inputs are validated and provides visual feedback to reduce errors.
* **Business Logic Layer:** Handles core operations such as calculating totals, managing stock, enforcing user roles, and processing orders. It acts as a bridge between the database and the user interface, ensuring that rules and calculations are consistently applied.
* **Reporting and Dashboard:** Generates detailed summaries of sales, revenue, products sold, and order counts. Dashboards update automatically with each transaction, and reports can display totals for specific days, weeks, or custom date ranges to help Admins make informed business decisions.

**Deployment Architecture:**

The system follows a **client-server architecture,** where multiple client applications can connect to a centralized SQL Server database. This ensures data consistency, allows cashier to work at the same time, and centralizes admin controls for product. The deployment is fully Windows-based, requiring .NET runtime on desktop or laptop and a SQL Server instance accessible by all client PCs on the network.

**Communication Protocols and Interfaces:**

Communication between the VB.NET client and SQL Server database occurs through **ADO.NET** connections. The system uses **parameterized SQL queries** and stored procedures to securely retrieve and manipulate data. All data transactions are handled over the local network using TCP/IP, ensuring secure and reliable communication between clients and the database server.

**Scalability Considerations:**  
The architecture allows for easy scaling by adding additional desktop or laptop or upgrading the SQL Server instance to handle higher transaction loads. Future extensions, such as integrating online ordering or multi-branch support, can be accommodated without significant redesign.

**DATABASE DESIGN**

The database design for the Kape Kalinaw Order Management System focuses on organizing orders and products to ensure data accuracy, consistency, and reliability. The system uses SQL Server as the centralized storage for all application data. The primary components of the system include Users, Products, Categories,MenuItems for products and OrderHistory. Each records has its own table, and the relationships between tables maintain data consistency and enable smooth operations.

The Users table stores login credentials and role information for Admins and Cashiers, controlling access to system functions. The Products table contains details about each product such as name, category (Coffee or Non-Coffee), size (Hot or Iced), price, and stock quantity. The Categories table groups products into logical types, making it easier to filter and manage items. The OrderHistory table records every order transaction, including OrderID, category, Hot/Iced, product, size, quantity, price, total, order date, grand total, payment, change, and order time. This table allows tracking of sales, revenue, and inventory usage in real time.

Relationships are designed so that each order can contain multiple products, and each product can appear in multiple orders. Users have roles that define what actions they can perform, ensuring proper access control. Data normalization techniques are applied to reduce redundancy and maintain consistency. For example, product details are stored in a single Products table rather than repeating them in each order, while categories are referenced by ID to maintain uniformity.

**Entity Relationship Diagram (ERD)**

Users

-----

UserID (PK)

Username

Password

Role

OrderHistory

------------

OrderID (PK)

Product

Category

HotIced

Size

Quantity

Price

Total

OrderDate

GrandTotal

Payment

Change

OrderTime

MenuItems

Products

Size

Price

**Data Normalization Techniques**

The Kape Kalinaw OMS is the process of organizing the database to reduce duplicate information and ensure accuracy. Each piece of data is stored in one place only, so updates are easier and mistakes are minimized. The system uses clear relationships between tables, such as Users, Products, and OrderHistory, so that each table focuses on a single type of information.

1. **First Normal Form (1NF)**

* Each column must hold atomic (indivisible) values.
* Each row must be unique.  
  Example: In OrderHistory, instead of storing multiple products in one row, each product sold gets its own row with a reference to the OrderID.

2. **Second Normal Form (2NF)**

* Ensures that all non-key attributes depend on the entire primary key (applies when using composite keys).
* Example: If OrderDetails table had OrderID + ProductID as a composite key, the Price of a product depends only on ProductID, not the full composite key. So, Price stays in Products table to avoid duplication.

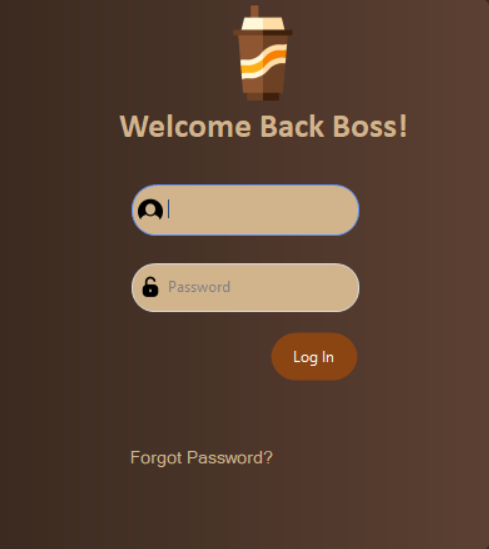
3. **Third Normal Form (3NF)**

* Removes transitive dependencies, meaning non-key attributes shouldn’t depend on other non-key attributes.
* Example: In OrderHistory, GrandTotal is derived from Price \* Quantity. Storing it separately is acceptable for performance, but you avoid storing redundant details like product category that’s already in Products.

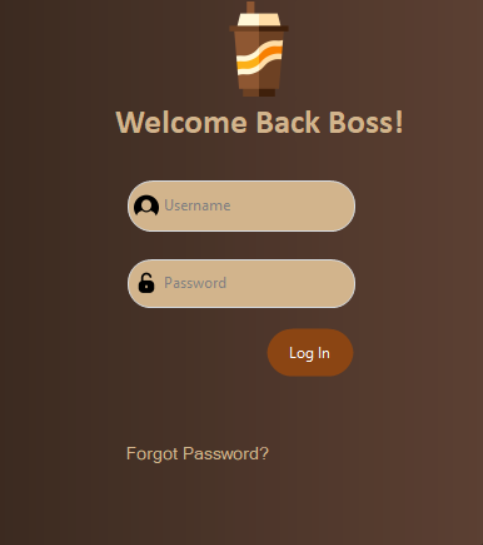
**USER INTERFACE DESIGN**

The user interface of the Kape Kalinaw Order Management System is designed to be simple, clean, and intuitive for both Admin and Cashier users. The system uses desktop forms built in VB.NET, where the Admin interface includes a sidebar with access to the Dashboard, Product Management, and Order History. The dashboard displays real-time metrics such as total revenue, total orders, total products sold, and total users. Admins can add, update, or delete products, update prices, and manage order history through clear buttons and input forms.

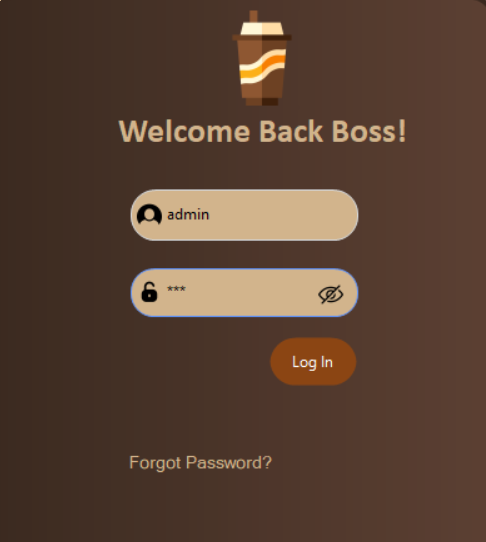
**Login Form**



*Figure 1*



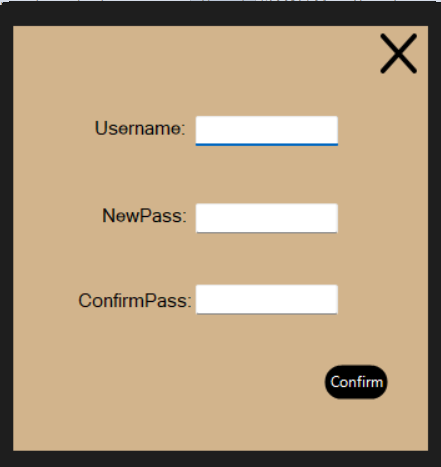
*Figure1.2*



*Figure 1.3*

The login interface allows both Admin and Cashier users to enter their username and password to access the system. It includes a "Forgot Password" option in case users forget their credentials. There is also an eye icon that enables users to toggle the visibility of the password, allowing them to either show or hide the password as they type.

**ForgotPassword Form**

****

*Figure 2*

In this form, the Admin or Cashier can reset their password if they forget it.

**Admin Form**

****

*Figure 1*

In the Admin form, the sidebar contains a menu button, a cashier button, a dashboard button, stocks button and a log-out button. The Cashier form is also included, where the cashier can select the products that the customer orders.

**Menu Form**

****

*Figure 2*

This is the Menu form, where pictures of the available menu items are displayed. The pictures can be updated if a new product is added.

**Dashboard Form**

****

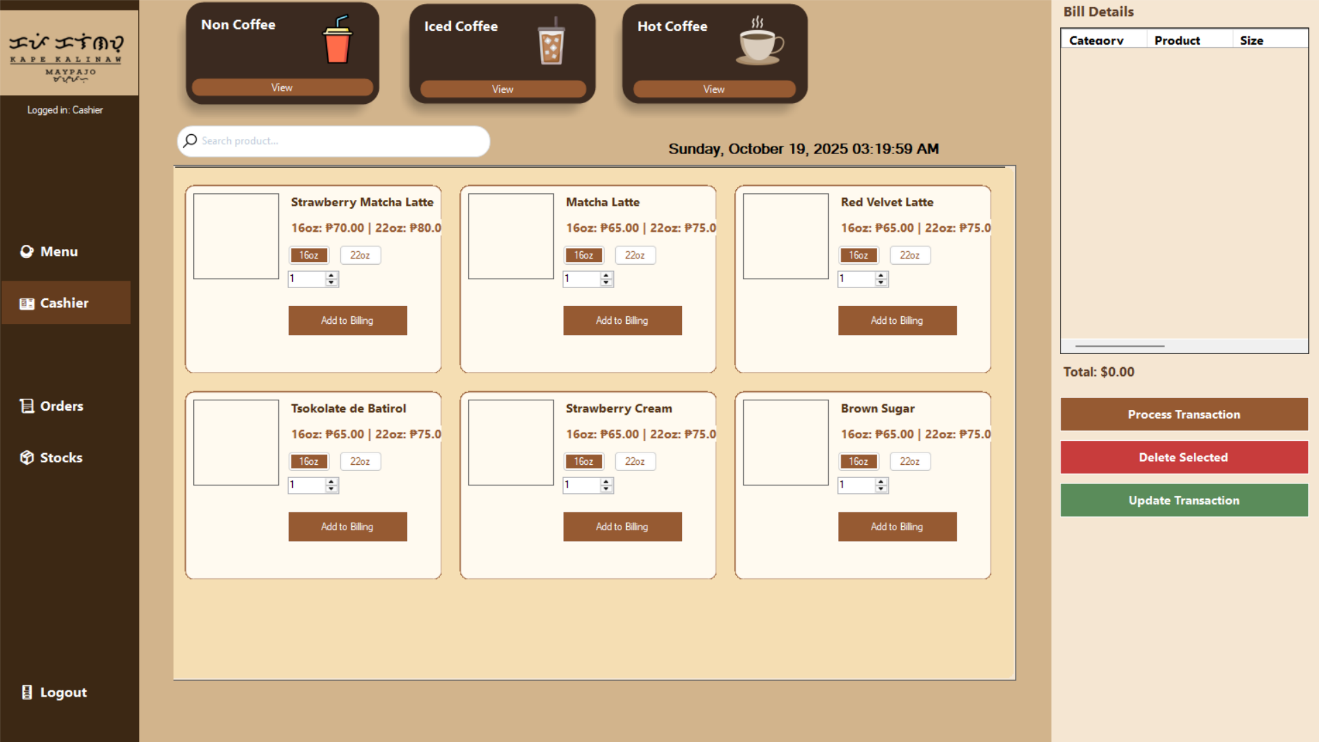
*Figure 3*

The DashboardForm serves as the central control panel for the Admin.It connects to the SQL Server database using the provided connection string, which allows it to load and manage records from the OrderHistory table. The form features a sidebar menu with buttons for navigation, such as the Menu form, the Cashier form, and the Logout option. These buttons are designed with hover effects to improve the user experience. At the center of the dashboard are four summary cards that display important business metrics, including the total number of orders, the total revenue calculated from the sum of GrandTotal, the number of products, and the number of users. These cards use clean icons and colors to provide a quick overview at a glance.

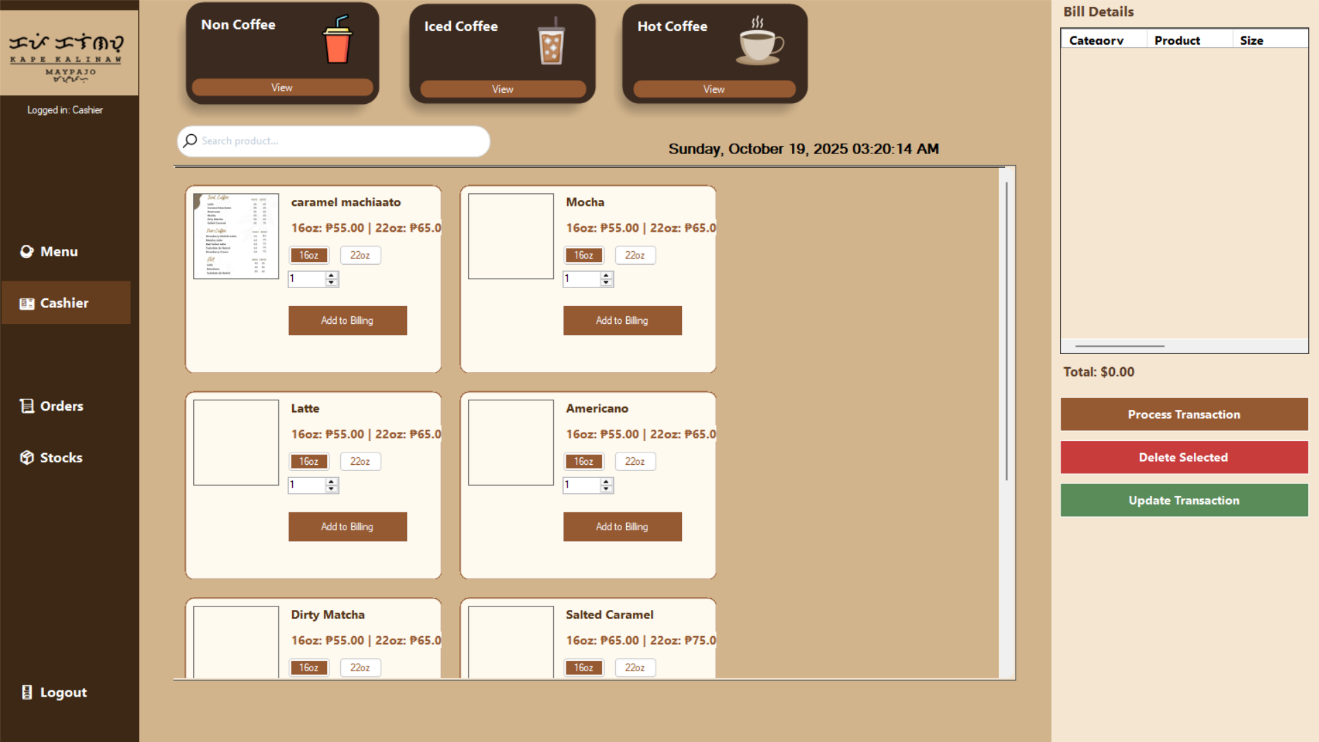
The DataGridView displays the detailed OrderHistory records, showing fields such as OrderID, Category, HotIced, Product, Size, Qty, Price, Total, GrandTotal, Payment, Change, OrderDate, and OrderTime. This table can be searched in real-time using the search box, allowing the admin to quickly locate records by typing keywords. Records can also be updated or deleted directly from the dashboard. The update button takes the currently selected row and applies changes to the database, while the delete button removes the chosen record permanently. Both operations refresh the grid and ensure the dashboard always shows the latest information.

The dashboard also includes a bar chart that visualizes orders trends for the last seven days, helping the admin understand sales performance over time. This chart automatically refreshes every minute to keep the data up to date. Alongside this, a label continuously displays the current date and time for easy reference. Overall, the design of the DashboardForm follows a modern flat style, with a clean layout, dark sidebar, and well organized components. It combines data visualization, record management, and navigation in one interface, making it a complete and functional admin tool.

**Cashier Form**

****

*Figure 4*



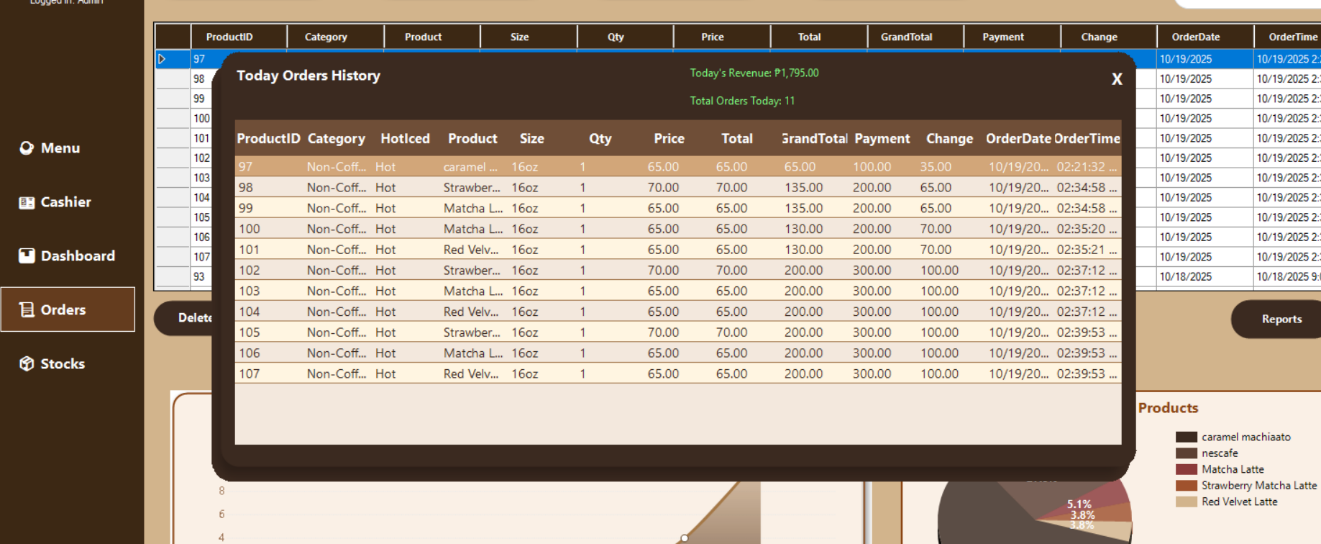
*Figure 4.1*



*Figure 4.2*

In the Cashier Form, the sidebar no longer includes the Dashboard button since it is only intended for cashier. This means the cashier only has access to the features related to processing customer orders and print receipt, without admin functions.

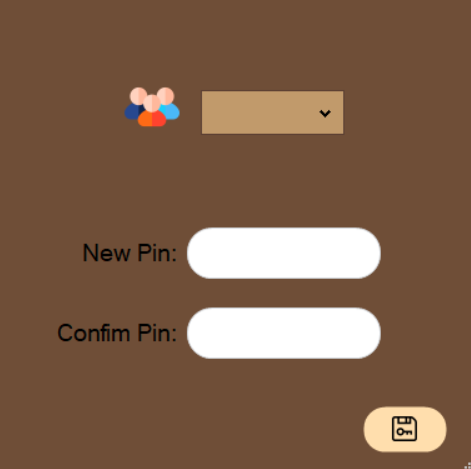
**Today Order History Form**

****

*Figure 5*

The Today’s Orders History shows all the orders made on the current day. It lists important details such as the order number, product, size, quantity, price, total amount, payment, change, and the time the order was placed. This helps the cashier or admin easily check and review today’s sales without going through all past records. It is also useful for keeping track of daily income and confirming customer transactions before closing.

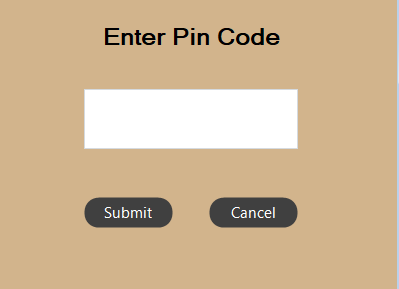
**Change Pin code Form**



*Figure 6*

This form is used to change the PIN code of either the admin or cashier account in the system.

**Pin Code Form**

****

*Figure 7*

In this form, you must enter the PIN code first before you can change the password and there’s an 3 attempts if you failed then you wait 5minutes to try it.

**COMPONENT DESIGN**

The Order Management System is divided into several key components, each with specific functions that work together to ensure smooth operation.

**Key System Components/Modules**

* **Login Module** - Manages authentication for both admin and cashier accounts. It validates usernames and passwords, provides access control, and includes features like password reset and password visibility toggle.
* **Admin Module** - Allows the administrator to manage system data such as menu items, prices, categories, and user accounts. It also provides access to reports and system settings.
* **Cashier Module** - Handles order transactions. The cashier can select menu, products, set quantities, process payments, and record orders in the database.
* **Order History Module** - Stores all completed orders, including details such as order ID, category, product, size, price, payment, and time of transaction. This module ensures records are accurate and retrievable for reporting.
* **Reporting Module** - Generates summaries such as daily sales totals, and revenue reports.

**Interface Specifications**  
Each module communicates with the database through SQL queries and stored procedures. For example, the Login Module uses a function like login username, password, role to verify credentials. The Cashier Module interacts with functions such as Add Order in order Details and Update in productID, quantity.

**Dependency Management and Interactions**

The system follows a layered structure where the User Interface Layer (VB.NET forms) interacts with the Business Logic Layer, which processes rules and validations, and then communicates with the Database Layer (SQL Server). This ensures separation of concerns and easier maintenance. Dependencies are managed so that each module can be updated independently while maintaining smooth communication with the database.

**DATA FLOW DIAGRAM**

Database

Admin

Cashier

The data flow diagram for the Order Management System show how information moves between users, processes, and the database. They provide a visual representation of how data is collected, processed, and stored. In this system, the main data sources are the administrator and cashier. The administrator enters data such as menu items, product categories, prices, and user accounts, while the cashier inputs customer orders and payment details.

The processing logic handles different operations, including validating login credentials, recording sales transactions, updating inventory levels, and generating reports. Each process communicates with the database to either save new records, update existing ones, or retrieve stored information.

The main destinations of the data are the database tables for users, products, sales, and order history. Reports are also generated as an output destination, providing summaries like daily sales totals, revenue reports. These reports help the administrator monitor performance and make business decisions.

**SECURITY DESIGN**

The security of the Order Management System is very important to protect sensitive information such as order details, payments, admin and cashier account. Proper security ensures that only authorized users like the admin and cashier can access the system, while preventing unauthorized access or misuse.

**Overview of Security Requirements and Considerations**

The system must ensure that only users with valid accounts can log in and access their assigned features. Admin accounts have more privileges such as managing menu items and viewing full reports, while cashier accounts are limited to recording orders and processing payments. Passwords and confidential information must be protected from being exposed.

**Authentication and Authorization Mechanisms**

The login form requires each user admin or cashier to enter their username and password before they can use the system. The system verifies the credentials by checking them against the database. Admin users are authorized to access the dashboard, product management, Settings, and reports, while cashiers are only authorized to access the order and sales interface. This separation ensures that no user can access features beyond their role.

**Data Encryption and Protection Measures**  
 All stored passwords in the database are encrypted using hashing techniques so they cannot be read directly. The “Show/Hide Password” feature in the login form only reveals the password to the user during input but does not expose it in storage. Sensitive data such as sales records and payments are only accessible through the system interface and not directly from the database. Backups of the database are also performed regularly to avoid data loss. In addition, the system requires users to enter their PIN code before they can change their password, providing an extra layer of security against unauthorized access.

**PERFORMANCE DESIGN**

The Order Management System is made to work fast and smooth. It should respond quickly when users log in, add orders, or create reports. The main goal is to make sure the system can handle many orders at the same time without slowing down, especially during busy hours.

To keep the system fast, the database is designed in a clear way with no duplicate data. Indexing is used on important fields like product names, order IDs, and dates so that searching and reporting are quicker. The system also updates sales and inventory right away to keep the records correct and up to date.

A performance test will be done to check if the system meets its goals. The test will include simulating users placing orders at once to see if it can handle the load. Report generation will also be tested to make sure it is still fast even when the database grows bigger. If problems like delays or slowdowns are found, they will be fixed to keep the system reliable and smooth.

**ERROR HANDLING AND LOGGING**

The system uses error handling to make sure that it can continue working even when problems happen. Instead of crashing or stopping, the system will show clear and simple error messages that guide the user. For example, if a cashier enters the wrong value or leaves a field empty, the system will display a message such as “This field cannot be empty.” If the database cannot connect, the system will also show a message like “Database connection failed, please try again later.” These messages help users understand what went wrong and how they can fix it.

Error handling is also used for security. When a login fails, the system will not give too much detail, but it will still let the user know that the username or password is incorrect. This prevents people from trying to guess passwords. The system checks every input from the user to make sure it is valid before saving it in the database. This reduces mistakes and keeps the records accurate.

The system also uses simple error codes to make it easier to identify issues. For example, E101 means the username or password is invalid, E202 means there is a database connection error, E303 means some input is missing or invalid, and E404 means the record was not found. These codes give developers and administrators a faster way to check and solve problems.

By using proper error handling and logging, the system becomes more reliable, user-friendly, and secure. Users will not be confused when errors happen, and administrators will have enough information to quickly find and fix problems. This helps keep the system stable and ensures that it continues to support daily operations without interruption.

**THIRD-PARTY INTEGRATION**

The system does not currently use any third-party services or external APIs. All features, such as order management, sales recording, are built directly into the application using VB.NET and Guna UI for the design of the user interface.

Since the system works as a standalone application, there is no need for outside data exchange formats such as JSON or CSV at this time. All operations, including data storage, retrieval, and reporting, are handled internally through the SQL Server database.

This design keeps the system simple, easy to maintain, and free from external dependencies. Future versions may include third-party integrations if needed, such as digital payment gateways or export tools, but the current version works fully on its own.

**DEPLOYMENT PLAN**

The deployment of the system will follow a step-by-step process to make sure the application and database are installed correctly and can be used without problems. The first step is to prepare the computer or laptop where the system will be installed. The computer or laptop must meet the hardware and software requirements. For hardware, the system should run on a Windows computer with at least 4GB of RAM, a dual core processor, and enough storage space for the database, reports, and application files. For software, the computer needs Windows OS, SQL Server for the database, and .NET Framework for running the VB.NET application.

Once the requirements are ready, the next step is installing SQL Server and creating the database. The tables for orders, products, categories, and order history will be set up, and the connection between the application and the database will be configured. After this, the VB.NET application will be installed and linked to the database so users can log in, manage products, and process orders.

Backup copies of the database and the application files will be created so that data is not lost in case of errors or crashes. Version control will be followed, meaning that every update or change in the application will have a version number. This makes it easier to track changes, fix bugs, and roll back to older versions if needed.

After the installation, testing will be done to make sure all features such as login, sales processing, reporting, and inventory updates are working correctly. If issues are found, they will be fixed before the system goes live. When everything is confirmed to be working, the system will be turned over for daily use.

The deployment plan ensures that the system is stable, secure, and easy to maintain even after it has been installed.

**MAINTENANCE AND SUPPORT**

The maintenance and support of the system are important to make sure it continues to work properly after deployment. Regular maintenance will help prevent errors, improve performance, and keep the system secure. The system should be checked from time to time to ensure that the database is not corrupted, the user interface is working smoothly, and no data is missing. Backups of the database will also be scheduled regularly so that data can be restored in case of problems.

Software updates, patches, and bug fixes will be handled through a clear procedure. When errors or bugs are reported, they will first be documented and tested. If the problem is confirmed, a patch or fix will be created and applied to the system. After the update, testing will be done again to make sure the fix works and does not cause new problems. Updates can also include improvements, like adding new features or making the system run faster.

There will be an escalation process to solve issues that cannot be fixed immediately. First, basic issues such as login errors or minor data problems will be handled by the system administrator. If the problem is more serious, like database corruption or application failure, it will be by the developer or for a deeper solution. In very rare cases where the system is down for a long time, an emergency backup system may be used until the issue is resolved.

By following these guidelines and procedures, the system will remain reliable and secure, ensuring that users such as the admin and cashier can continue their work without interruption.

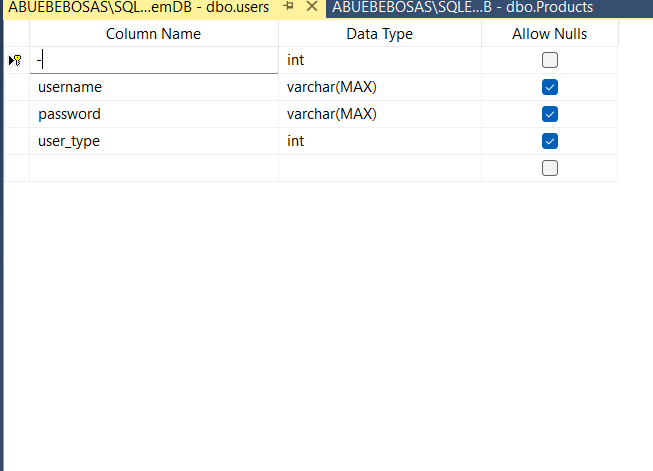
**REVISION HISTORY**

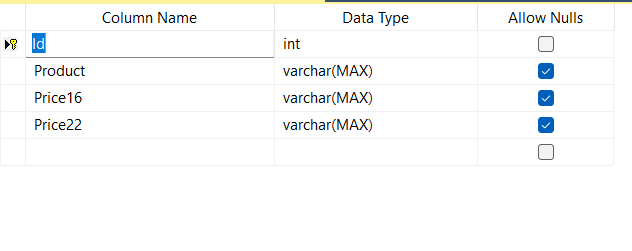
The revision history keeps a record of all the changes made to the document during the project. This includes the date of the change, the version number, the person who made the change, and a short description of what was updated. Keeping a clear log of changes ensures that all team members know the progress of the project and can track how the system design and documentation have developed over time.

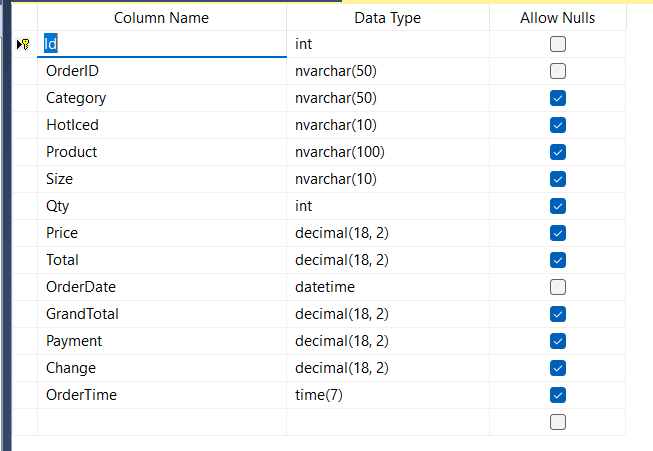
|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description of Change |
| 1.0 | 8/30/2025 | Lorenzana,Joshua | |  | | --- | |  |  |  | | --- | | Initial draft of the document created. | |
| 1.1 | 8/31/2025 | Lorenzana,Joshua | |  | | --- | |  |  |  | | --- | | Updated system architecture details. | |
| 1.2 | 9/1/2025 | Lorenzana,Joshua | |  | | --- | |  |  |  | | --- | | Added error handling and security design sections. | |
| 1.3 | 9/7/2025 | Lorenzana,Joshua | Update the U.I |

**APPENDIX**

This appendix provides additional information that supports the Order Management System (OMS) for Kape Kalinaw. It includes diagrams, screenshots, and technical references that give a clearer view of how the system works.







Output

Database

Application Form

User

**References**

* Microsoft SQL Server Documentation
* VB.NET Windows Forms Documentation
* Guna.UI2 Framework