

Department of Software Engineering

Course Code: SE133

Course Title: Capstone project

Project Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	5
Clarity	1					
Content Quality	2					
Spelling & Grammar	1					
Organization and Formatting	1					
Total obtained mark						

Semester: Fall-2024

Team : 7 (1010, 1213, 1060, 1059)

Batch: 41 Section: C

Course Code: SE133

Course Teacher Name: Abdul Hai Jibon

Project Report: Supershop Management System

Project Title

Supershop Management System with Advanced Features

Introduction

Supershop management is a crucial task that ensures smooth operations, efficient customer service, and effective stock control. This project aims to develop a Super Shop Management System using the C programming language. The software will help manage inventory, sales, and employee roles while introducing advanced features for better performance and usability.

The system is designed to streamline daily operations, reduce manual work, and ensure accurate data handling. It is a practical solution for supershops seeking to modernize their operations and improve efficiency.

Requirement Analysis:

1. Register
2. Login
3. View Product Categories
4. Search Product
5. View Product Details
6. Add New Products
7. See Order List
8. Choose Payment Method

9. View Membership
10. View Membership Details
11. See Employee Management
12. View Employee Performance
13. Get Help and Support
14. Logout

Stakeholders :

The stakeholders involved in this project include:

- Owner
- Manager
- Staff
- Costumer

Tools and Technologies UsedProgramming Language:

- **C Language**
 - Selected for its efficiency, performance, and control over hardware resources.

Compiler:

- GCC (GNU Compiler Collection)

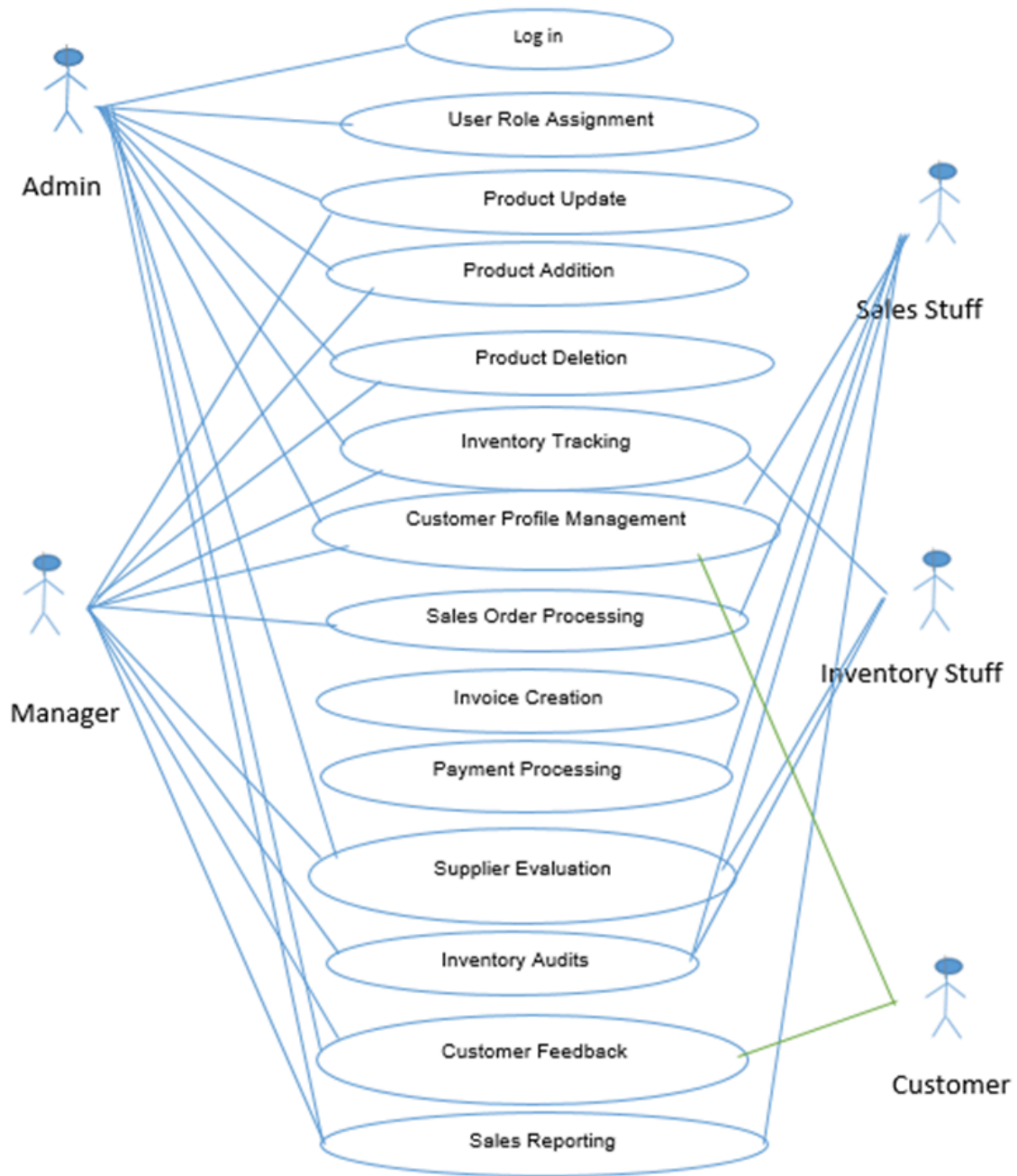
Development Tools:

- **Code Editor:** Visual Studio Code / Code::Blocks

A Software Requirements Specification (SRS) is a detailed document that describes the functionality, features, and constraints of a software system. It serves as a blueprint for developers, stakeholders, and testers to understand the system's expected behavior. The SRS includes both functional requirements (what the system should do) and non-functional requirements (how the system should perform). It ensures clear communication and alignment between all parties involved in the software development process.

Requirement & Description	FR No. & Stakeholder
Register A non-logged customer must create an account to access the system's main features, including buying products. Non-logged customers can view limited features like searching products and viewing product categories.	FR 01 Non-logged Customer
Login Registered users (customers, staff, or managers) can log in to access their specific interfaces and features. Admins and executives can log in using their credentials for management tasks.	FR 02 Logged Customer, Executive, Admin
Forget Password The logged customer, executive, and admin can easily change their password.	FR 03 Logged Customer, Executive, Admin
Search Product Any stakeholder can search for specific products using keywords or filters.	FR 04 Non-logged Customer, Logged Customer,
View Product Categories Customers can browse products categorized into different sections to make navigation easier.	FR 05 Non-logged Customer, Logged Customer
Add New Products Staff or managers can add new products to the inventory by providing all required product information.	FR 06 Manager, Staff
See Order List Customers can view their past and ongoing orders. Staff and managers can access all order details for management purposes.	FR 07 Logged Customer, Manager, Staff
Choose Payment Method Customers can select a payment method (e.g., card, PayPal) during the checkout process.	FR 08 Logged Customer
View Membership Customers can view available membership plans and benefits.	FR 09 Logged Customer
View Membership Details Customers can see detailed information about their current membership plans.	FR 10 Logged Customer
Employee Management Managers can view and manage employee data, including roles, contact details, and performance records.	FR 11 Manager

Use case Diagram for Super shop management

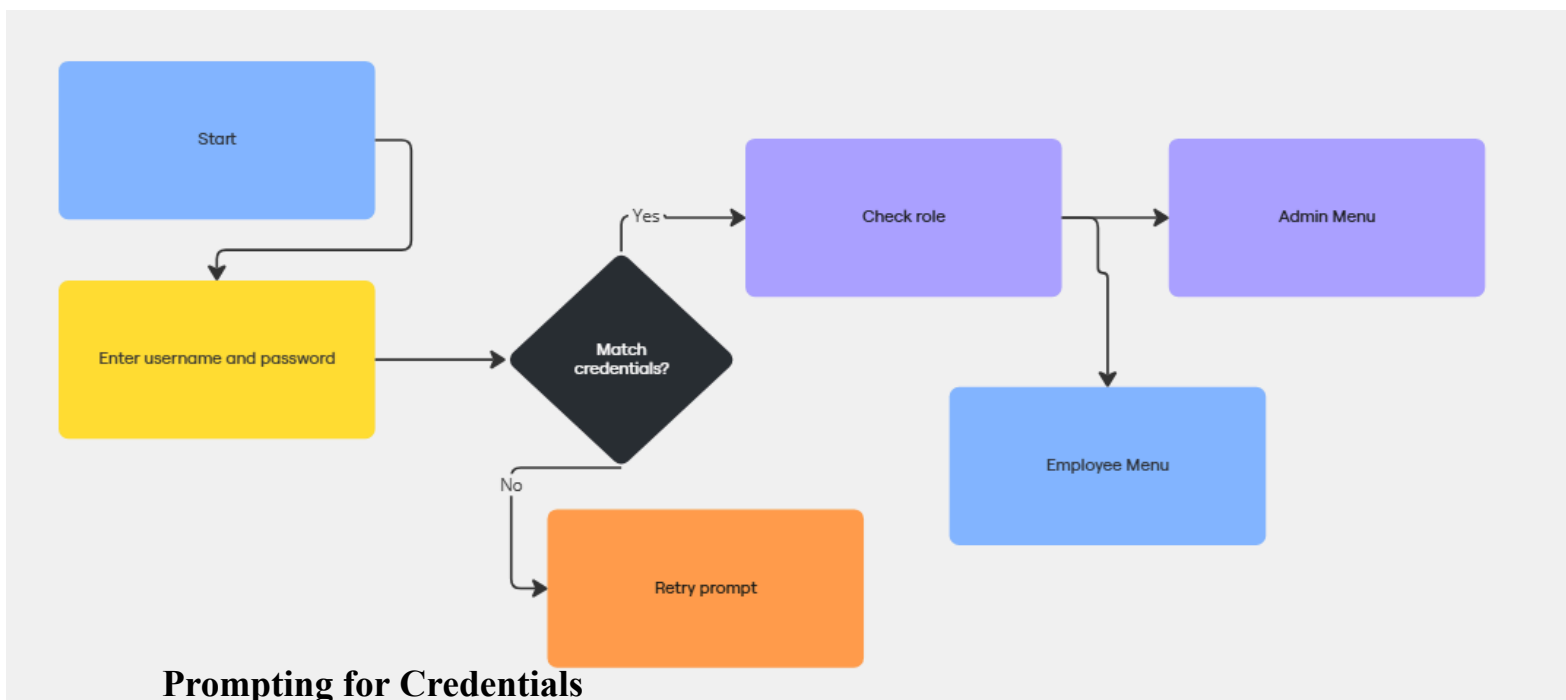


1. Login Authentication – Detailed Explanation

The login system is the foundational component of the project, ensuring secure access and role-based functionality. Below is a comprehensive breakdown of its workings and considerations:

- The login mechanism protects the system by verifying the identity of users.
- It determines access levels, redirecting users to either the Admin Menu or Employee Menu based on their role.
- Ensures unauthorized users cannot interact with sensitive features, such as inventory management or billing operations.

Login Workflow Diagram



Prompting for Credentials

1. The program prompts the user to input a username and password.
2. These inputs are typically captured via `scanf()` in C.

Verification of Credentials

- The system matches the entered credentials against predefined or stored values.
- If credentials are incorrect, an error message is displayed, and the user is prompted to try again.

Access Based on Role

On successful authentication:

Admin Users are redirected to the Admin Menu for high-level management tasks.

Employee Users are redirected to a simplified interface to perform day-to-day operations.

1. Role-Based Access Control

- **Admins:** Full access to system functionalities like inventory management, sales reports, and user management.
- **Employees:** Limited access for conducting billing and customer interactions only.

2. Password Protection

- The system enforces security by requiring a correct password match.
- Consider integrating hashed passwords for enhanced security.

3. Retry Mechanism

- Users are given unlimited retries until successful authentication or exit.
- Enhancements could include a maximum retry limit to prevent brute-force attacks.

4. Modular Code

- The `authenticate()` function separates login logic from the main flow, making it reusable and easier to maintain.
-

1. Storing Credentials Securely

- Currently, user credentials are hardcoded in the program. For scalability:
 - Store credentials in a file or database.
 - Use hashing algorithms like MD5 or SHA for secure password storage.

2. User Feedback

- Provide detailed feedback for failed attempts:
 - “Invalid Username” vs. “Incorrect Password” can help debug issues.

3. Forgot Password Option

- Implement a password reset mechanism where users can recover or reset credentials via email or security questions.

4. Session Management

- Add session tracking to ensure that users stay logged in for a defined period.
 - Allow users to log out manually.
-

1. Brute Force Prevention

- Implement a delay or block the account after multiple failed login attempts.

2. Data Encryption

- Encrypt passwords using secure hashing algorithms before storing them.

3. Secure Input Handling

- Use `fgets()` instead of `scanf()` to prevent buffer overflows.
 - Sanitize inputs to avoid injection attacks.
-

To make the login system visually engaging, include the following images:

1. Login Interface Mock-Up

- Create a simple GUI-like representation of the login screen.
- Example: Labels for username, password fields, and a "Login" button.

2. Flowchart of Authentication

- A flowchart demonstrating the flow from input to role-based redirection.

3. Error Message Example

- Screenshot of the program output when invalid credentials are entered.

4. Access Granted

- Display of the welcome message and redirection to respective menus.

2. Modules and Their Functionality

2.1 Login System

- Prompts the user to enter credentials.
- Authenticates user based on role (**admin** or **employee**).
- Redirects to respective menus after successful login.

Demonstration:

- Enter **admin** credentials (e.g., username: **admin**, password: **admin123**) and show redirection to the Admin Menu.
- Similarly, enter **employee** credentials and demonstrate access to the Employee Menu.

Suggested Image:

Screenshot of the login interface with labeled fields.

- **Placement:** After demonstrating the login system.
-

2.2 Admin Menu

- **Options:**
 1. Inventory Management
 2. View Sales Report
 3. Low Stock Alerts
 4. Logout

Demonstration:

Navigate to **Inventory Management**, view and update product details, and demonstrate how low-stock alerts work.

Suggested Image:

A table of sample products with low stock and a graphical representation (e.g., bar chart).

- **Placement:** After the low-stock alert demonstration.
-

2.3 Employee Menu

- **Options:**
 1. Billing
 2. Logout

Demonstration:

Simulate a customer purchasing products.

- Add multiple items to the bill.
- Show a warning if stock is insufficient.
- Update stock after purchase.

Suggested Image:

Screenshot of the billing system in action.

- **Placement:** After the billing demonstration.
-

2.4 Inventory Management (Admin Only)

- **Features:**
 - **Add Product:** Add new product details to the system.
 - **View Products:** Display all available products with price and stock.
 - **Update Stock:** Modify the stock of a specific product.
 - **Delete Product:** Remove a product from the inventory.

```
Login successful as Admin!
```

```
=== Admin Menu ===
```

```
1. Inventory Management
```

```
2. View Sales Report
```

```
3. Low Stock Alerts
```

```
4. Logout
```

```
Enter your choice: 
```

Demonstration:

Perform all inventory operations and display the effects (e.g., adding a product and verifying it appears in the product list).

Suggested Image:

Diagram or screenshot showcasing the product addition process.

- **Placement:** After demonstrating adding a product.
-

2.5 Billing System (Employee Only)

- Enables employees to generate bills for customers.
- Automatically updates the stock after billing.
- Stores billing data in a file (`bills.txt`).

Demonstration:

- Add a product to the bill.
- Show that the stock decreases in the inventory file after billing.

Suggested Image:

Display a mock bill as a screenshot or a generated text file.

- **Placement:** After generating the bill.
-

2.6 Sales Report and Low Stock Alerts (Admin Only)

- **Sales Report:** Summarizes sales data from the bills file.
- **Low Stock Alerts:** Lists products with stock below a threshold (e.g., 5 units).

Demonstration:

Simulate a few transactions and then generate a sales report. Show the low stock alert system in action.

Suggested Image:

Graph showing sales trends and low-stock product representation.

- **Placement:** After the sales report demonstration.

3. Admin and Employee Menus

The **Admin** and **Employee** menus are integral parts of this project, serving different purposes and providing distinct functionalities based on the user's role. Both menus are designed to enhance the user experience, offering simple but comprehensive interfaces for system management and operations. Below is a detailed explanation of how each menu works and the key operations involved.

Admin Menu: Operations and Functionalities

The **Admin Menu** provides the system administrator with control over the core functionalities of the program. Admins are responsible for maintaining the product inventory, managing sales reports, and monitoring low-stock alerts. The Admin interface should be robust, with easy navigation between options to ensure smooth operations.

Menu Structure

Upon successful login with an admin account, the system displays the Admin Menu. The typical options available include:

- **Option 1: Add Product**
- **Option 2: View Product List**
- **Option 3: Update Product Details**
- **Option 4: Delete Product**
- **Option 5: Sales Report**
- **Option 6: Low-Stock Alert**
- **Option 7: Logout**

These options are chosen based on the specific needs of an admin to manage the store's products and perform key administrative duties.

Key Features of Admin Menu

1. Add Product

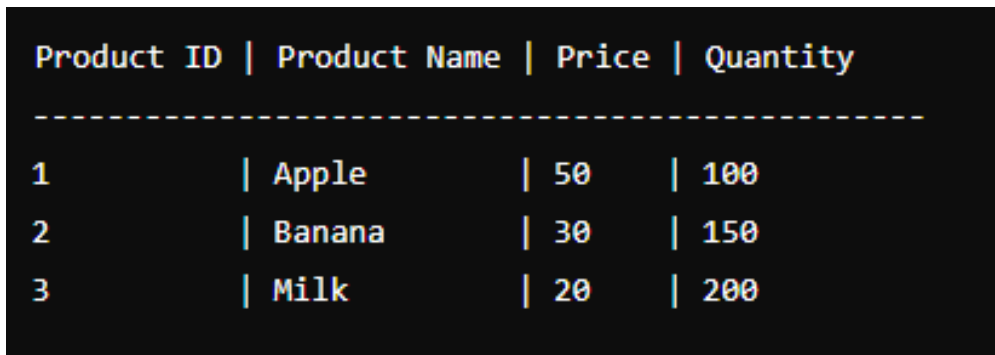
The **Add Product** functionality allows the admin to add new items to the product inventory. When a product is added, the program requires details like the product name, price, quantity, and description. This data is then saved into a file (`products.txt`), which maintains the inventory's state.

Process:

- Admin is prompted to enter product details (name, price, stock quantity).
- Validations ensure the entered data is accurate (e.g., non-negative price or quantity).
- Data is appended to `products.txt` as a new entry. The file is updated automatically, allowing for easy tracking and retrieval of product information.

View Product List

This function displays the entire product list stored in the `products.txt` file. The admin can see the details of all products in the inventory, including their names, prices, and quantities. The list can be shown in a tabular format



Product ID	Product Name	Price	Quantity
1	Apple	50	100
2	Banana	30	150
3	Milk	20	200

Process:

- The program reads `products.txt` and extracts product details.
- It formats and displays the list to the admin in a clean, readable format.

- The list helps admins track product availability, prices, and make inventory decisions.

2. Update Product Details

The **Update Product** option enables the admin to modify the details of any product in the inventory. This could involve

3. changing the product's name, price, or stock quantity. Admins can access this option by entering the product's ID, and the system will display the current product data for modification.

Process:

- Admin selects the product to be updated by entering its ID.
- The current details of the product are displayed.
- Admin enters the new details (e.g., a new price or quantity).
- The system updates the record in the `products.txt` file, preserving the data integrity.

4. Delete Product

If a product is no longer available for sale or needs to be removed from the inventory, the **Delete Product** function allows the admin to delete it from the system. This operation requires the admin to identify the product to be removed, typically by its product ID.

Process:

- Admin selects the product by entering its ID.
- The program deletes the product record from the `products.txt` file.
- The file is rewritten with the remaining products, ensuring the inventory reflects the most up-to-date data.

5. Sales Report

Sales reports allow the admin to generate insights about product sales. These reports can track the quantity of each item sold, the total sales revenue, and trends over time. This feature would help in strategic decision-making for inventory management, identifying best-sellers, and planning promotions.

Process:

- The program generates a report based on the transactions recorded in the `bills.txt` file.

- The report can include total sales, number of items sold, and revenue from individual products.
- A sample sales report might look like:

Output: The system can display the sales data on the screen or save it to a file for later analysis.

6. Low-Stock Alert

This feature helps the admin keep track of products that are running low on stock. The system checks the inventory (the `products.txt` file) and flags products that are below a certain threshold (e.g., 20 items remaining).

Process:

- The system reads the product quantity from `products.txt`.
- If the quantity is below the defined threshold, the program alerts the admin, prompting a restocking action.
- This alert is particularly useful for preventing stockouts and ensuring continuous availability of products.

The **Employee Menu** is designed for employees who interact with the customers directly. It's typically a simplified version of the Admin Menu, focusing on the day-to-day sales operations.

Upon login, an employee is presented with the following options:

- **Option 1: Generate Bill**
- **Option 2: Logout**

Generate Bill The **Generate Bill** function is at the heart of the Employee Menu. When customers make a purchase, the employee enters the product details, such as product ID and quantity, and the system calculates the total cost, including taxes or discounts if applicable.

Process:

- The employee enters the product ID and quantity.
- The program checks the `products.txt` file for the product's price.
- The system calculates the total cost (quantity * price) and prints the bill.

- The transaction is then logged into the `bills.txt` file to track the sale for reporting purposes.

Certainly! Let's elaborate on **Point 4: Inventory Management** in great detail. This part of the project is crucial because it covers the core functionality for maintaining the store's stock, adding new products, updating existing products, and ensuring that inventory records are managed effectively.

```
-----  
Store Name: XYZ Store  
Date: 2024-12-01  
Customer: John Doe  
-----  
Product Name: Apple  
Quantity: 2  
Price: 50  
Total: 100  
-----  
Total Amount: 100  
-----
```

4. Inventory Management System - Detailed Explanation

Objective of Inventory Management:

Inventory management in your system is designed to keep track of the products that a store has, monitor the quantities of each product, and make it easy for administrators to perform operations like adding, updating, deleting, and viewing products in the inventory.

The **Inventory Management** module is central to ensuring that stock levels are accurate and that the right products are available when customers make a purchase. It involves several key functionalities:

1. **Adding New Products to Inventory**
 2. **Viewing Existing Products**
 3. **Updating Product Information**
 4. **Deleting Products**
 5. **Low Stock Alert**
 6. **File Handling** (Storage and retrieval of product data)
-

1. Adding New Products to Inventory

Goal:

The goal is to add new products to the store's inventory, capturing essential information about the product such as its name, category, price, and stock quantity. This operation ensures that when a new item arrives, it is correctly logged into the system.

Key Steps Involved:

- **User Input:** The admin enters the product details like:
 - Product Name
 - Product ID or SKU (Stock Keeping Unit)
 - Category (such as electronics, clothing, food, etc.)
 - Price (cost and selling price)
 - Stock Quantity (how many units are available in the inventory)
- **Validation:** Ensure that all required fields are filled out, and check for logical errors (e.g., stock quantity should not be negative, price should not be zero, etc.).
- **Storage:** The new product is saved to a file (e.g., `products.txt`), with each product entry on a new line. The product's details are typically saved in

a structured format like CSV (comma-separated values) or tab-delimited text.

- **Feedback to Admin:** After adding a new product, provide confirmation to the admin, showing the entered details and ensuring the data is saved correctly.

Code Example for Adding a Product:

```
void addProduct()
{
    Product product;
    FILE *file = fopen(PRODUCT_FILE, "a");

    printf("Enter Product ID: ");
    scanf("%s", product.productID);
    printf("Enter Product Name: ");
    scanf("%s", product.name);
    printf("Enter Product Price: ");
    scanf("%f", &product.price);
    printf("Enter Stock Quantity: ");
    scanf("%d", &product.stock);

    fprintf(file, "%s %s %.2f %d\n", product.productID, product.name, product.price, product.stock);
    fclose(file);
    printf("Product added successfully!\n");
}
```

- **Explanation:**

- The function `addProduct` first defines a `Product` structure to store the new product's details.
- The program prompts the admin to input the product's name, category, price, and stock quantity.
- The product details are written into the `products.txt` file in a structured format, where each field is separated by a comma.
- After the data is written, the file is closed, and a confirmation message is displayed.

2. Viewing Existing Products

Goal:

The goal is to allow the admin to view the entire list of products currently stored in the inventory. This provides visibility on stock levels, pricing, and product details. It also helps the admin to assess whether any items are low in stock or need to be restocked.

Key Steps Involved:

- **Display Information:** When the admin selects this option, the system reads the product data from the file and displays it in a readable format (tabular form or list).
- **File Handling:** The system opens the `products.txt` file, reads the data line by line, and prints it on the screen.
- **Sorting (Optional):** You can add a sorting option to display products based on certain criteria (e.g., by name, price, or stock quantity).

Code Example for Viewing Products:

```
void viewProducts()
{
    FILE *file = fopen(PRODUCT_FILE, "r");
    Product product;

    printf("\n=== Product List ===\n");
    printf("ID\tName\t\tPrice\tStock\n");
    printf("-----\n");
    while (fscanf(file, "%s %s %f %d", product.productID, product.name, &product.price, &product.stock) != EOF)
    {
        printf("%s\t%s\t\t%.2f\t%d\n", product.productID, product.name, product.price, product.stock);
    }
    fclose(file);
}
```

- The function `viewProducts` opens the file `products.txt` in read mode.
 - It reads each product's data and displays it in a tabular format for easy reading.
 - If the file is empty or does not exist, it notifies the admin that no products are available.
-

3. Updating Product Information

Goal:

Sometimes, product details need to be updated due to price changes, quantity adjustments, or re-categorization. This feature allows the admin to modify existing product information.

Key Steps Involved:

- **Product Search:** The admin must be able to search for a product by its name or ID.
- **Edit Information:** After finding the product, the admin is prompted to input the new values for the product fields (such as price or stock quantity).
- **File Update:** The system updates the relevant product data in the file by rewriting the modified line.
- **Confirmation:** Once the update is made, the admin is informed of the successful modification.

Code Example for Updating a Product:

```
void updateStock()  
{  
    char id[10];  
    int newStock;  
    Product product;
```

- **Explanation:**
 - This function allows the admin to search for a product by its name. If the product is found, the admin is prompted to enter the new price and stock quantity.
 - The program updates the corresponding product details in the file.
 - If the product isn't found, an error message is displayed.

```
    printf("Enter Product ID to Update Stock: ");  
    scanf("%s", id);  
    product = findProductbyID(id);  
  
    if (strlen(product.productID) == 0)  
    {  
        printf("Product not found!\n");  
        return;  
    }  
}
```

4. Deleting Products

Goal:

```
printf("Current Stock: %d\n", product.stock);  
printf("Enter New Stock Quantity: ");
```

This feature allows the admin to remove products from the inventory if they are no longer needed or discontinued.

```
scanf("%d", &newStock);  
product.stock = newStock;
```

```
    updateProductFile(product);  
    printf("Stock updated successfully!\n");  
}
```

Key Steps Involved:

- **Product Search:** Admin searches for the product to delete by name or ID.
- **Deletion:** If the product is found, it is removed from the file.
- **File Handling:** Since deleting a specific line in a file isn't a simple task, the system creates a temporary file and rewrites the content without the deleted product.
- **Confirmation:** The admin is notified whether the deletion was successful.

Code Example for Deleting a Product:

```
void deleteProduct()
{
    char id[10];
    FILE *file = fopen(PRODUCT_FILE, "r");
    FILE *temp = fopen("temp.txt", "w");
    Product product;

    printf("Enter Product ID to Delete: ");
    scanf("%s", id);

    while (fscanf(file, "%s %s %f %d", product.productID, product.name, &product.price, &product.stock) != EOF)
    {
        if (strcmp(product.productID, id) != 0)
        {
            fprintf(temp, "%s %s %.2f %d\n", product.productID, product.name, product.price, product.stock);
        }
    }

    fclose(file);
    fclose(temp);
    remove(PRODUCT_FILE);
    rename("temp.txt", PRODUCT_FILE);

    printf("Product deleted successfully!\n");
}
```

5. Billing System: In-Depth Explanation

The **Billing System** in your project is a critical component of the software, designed to handle customer transactions, generate invoices, and update the inventory in real time. It is the backbone for sales operations within the application. Let's break down its functionality in great detail, including how it interacts with the inventory and generates the bill for customers.

The billing system can be divided into several functions, each responsible for a different aspect of the transaction. These functions are integral to the operation and allow employees to perform seamless and efficient billing activities. The core operations of the billing system include:

1. **Product Selection:**

The employee chooses the product(s) the customer is purchasing from the available inventory. This involves displaying the list of products with their details (e.g., name, price, quantity available), allowing the employee to select from them.

2. **Customer Input:**

The system collects the quantity of each product the customer wants to buy. Employees can input this information manually (e.g., through a console interface) or by selecting items from a list. The quantity entered is used to calculate the total price for the transaction.

3. **Calculation of Total Bill:**

For each item selected, the system multiplies the quantity by the unit price of the product to determine the total cost of the item. This is repeated for each product in the customer's shopping cart, and the individual totals are summed up to get the final total price. Discounts and taxes can also be added at this stage, depending on the requirements.

4. **Inventory Update:**

Once the items are selected and the transaction is processed, the system reduces the available stock for each product in the inventory. This real-time update ensures that the inventory remains accurate after each sale. If an item goes out of stock, the system can flag it, so the inventory manager is alerted.

5. **Generating the Invoice:**

The system generates an invoice for the customer, which contains all the necessary details of the transaction, including the item names, quantities, individual prices, total price, taxes, and any applicable discounts. The invoice can be printed, emailed, or saved for future reference. This invoice serves as a proof of purchase and is important for both the customer and the business for record-keeping.

6. **Transaction Recording:**

The transaction is recorded in a file (e.g., `bills.txt`) for future reference and tracking. This file contains the details of every sale, including the time and date, product details, customer details, and total bill amount. This helps with generating reports for sales analysis later on.

```

void generateBill()
{
    FILE *file = fopen(BILL_FILE, "a");
    char productID[10];
    int quantity;
    Product product;
    Bill bill;

    printf("Enter Customer Name: ");
    scanf("%s", bill.customerName);

    while (1)
    {
        printf("Enter Product ID (or 'done' to finish): ");
        scanf("%s", productID);
        if (strcmp(productID, "done") == 0) break;

        product = findProductByID(productID);

        if (strlen(product.productID) == 0)
        {
            printf("Product not found!\n");
            continue;
        }

        printf("Enter Quantity: ");
        scanf("%d", &quantity);

        if (quantity > product.stock)
        {
            printf("Insufficient stock!\n");
            continue;
        }

        bill.totalPrice = product.price * quantity;
        fprintf(file, "%s %s %s %.2f\n", bill.customerName, product.productID, product.name, bill.totalPrice);

        product.stock -= quantity;
        updateProductFile(product);
    }

    fclose(file);
    printf("Bill generated successfully!\n");
}

```

Sample Invoice Output:

=====			
INVOICE			
=====			
Customer Name: John Doe			
Date: 2024-12-06			

Product	Quantity	Unit Price	Total Price

Apple	3	10.00	30.00
Banana	2	5.00	10.00
Milk	1	2.50	2.50

Total Amount:			42.50

Tax:			5.00
Discount:			0.00
Final Total:			47.50
=====			

The **Super Shop Management System** project aims to streamline the operations of a retail business by offering an integrated platform that facilitates efficient inventory management, billing, and sales reporting. It caters to both administrative and employee roles, providing a seamless user experience through clear menu options and functionalities.

Key Achievements:

1. **Efficient Inventory Management:** The system allows administrators to easily add, update, view, and delete product details. The integration of file handling ensures that inventory data is saved securely and updated in real-time.

2. **Seamless Billing System:** Employees can quickly generate customer bills, manage stock levels, and track sales transactions. The automated process reduces human error and enhances customer satisfaction by ensuring fast and accurate billing.
3. **Role-based Access:** The login system authenticates users based on predefined roles (Admin or Employee). Admins can perform a wide range of tasks such as inventory updates and sales reporting, while employees focus primarily on sales and billing, ensuring security and data integrity.
4. **Sales and Reporting:** Admin users can generate sales reports to track the store's performance, monitor stock levels, and identify low-stock products. This helps businesses make informed decisions, plan purchases, and ensure that popular items are always available.
5. **File Handling and Data Persistence:** The program stores critical information such as product details and transaction history in text files (`products.txt` and `bills.txt`). This ensures that all data is preserved and can be accessed when needed, even after the system is restarted.

Challenges Overcome:

- **Data Integrity:** Ensuring that all transactions are accurately reflected in both the billing system and inventory was a key challenge. This was successfully addressed through thorough input validation and consistent file handling.
- **User Interface Design:** While the system operates in a text-based environment, it was designed to be user-friendly with clear prompts and easy navigation for both admins and employees.

Future Improvements:

- **GUI Integration:** Moving from a command-line interface to a graphical user interface (GUI) could improve the user experience, making it more intuitive and visually appealing.
- **Database Integration:** A shift from text files to a relational database like MySQL or SQLite would provide more scalable and robust data management.

- **Additional Features:** Implementing features like product search, detailed reporting, and automatic stock reorder notifications could further enhance the system's functionality.

Conclusion:

The **Super Shop Management System** successfully addresses the core needs of a retail business by improving inventory control, streamlining the billing process, and providing actionable insights through sales reports. It is a valuable tool for enhancing operational efficiency, reducing errors, and improving customer service in a retail environment. With further development and future enhancements, this project has the potential to evolve into a more comprehensive solution for modern retail management.