FOP

(Lab)

**Project Proposal Final**

**[Stationary Management System]**

**Last Date of Submission: --------------------------**

|  |  |  |
| --- | --- | --- |
| **Submitted to** | |  |
| **S.No.** | **Name** | **Registration Number** |
| **01** | Member 1 |  |
| **02** | Member 2 |  |
| **03** | Member 3 |  |
| **04** | Member 4 |  |
| **05** | Member 5 |  |

Contents

[Team Declaration 3](#_Toc186182939)

[Title: Stationery Management System 4](#_Toc186182940)

[Objective: 4](#_Toc186182941)

[Project Overview: 4](#_Toc186182942)

[Program Login Following are the Login credential for the program 4](#_Toc186182943)

[Project Details: 5](#_Toc186182944)

[Program Flow: 5](#_Toc186182945)

[**Key Functionalities:** 6](#_Toc186182946)

[**Discount Logic:** 6](#_Toc186182947)

[**Error Handling:** 6](#_Toc186182948)

[Program Structure: 7](#_Toc186182949)

[Output Screen 9](#_Toc186182950)

[Conclusion: 10](#_Toc186182951)

[References: 10](#_Toc186182952)

# **Team Declaration**

We, the undersigned, hereby declare that the project titled **"Stationery Management System"**, developed as part of the requirements for the **Fundamentals of Programming (FOP)** course in the **Bachelor of Science in Artificial Intelligence (BSAI)** program, is the result of our collective effort and contribution. This system, implemented using the **C++ programming language**, is designed to facilitate the management and purchase of stationery items, with distinct functionalities for both staff and customers.

The core features of the system include:

* **Staff functionality**: Staff members can log in to view and update stock levels and prices.
* **Customer functionality**: Customers can browse available items, make purchases, and receive discounts based on the total amount spent.

This project applies fundamental programming principles, including user input/output handling, conditional structures, and looping mechanisms. The system also incorporates features for generating invoices and applying discounts, ensuring a smooth user experience.

We affirm that the work presented in this project is entirely original and the result of our own effort. We have not copied or plagiarized any part of this project from external sources. All external resources utilized during the development of the project have been properly acknowledged and referenced.

**Team Members: Signature**  
  
  
Member 1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
Member 2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
Member 3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
Member 4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Date:** -----------------

# Title: Stationery Management System

### Objective:

This C++ project simulates a basic stationery management system where a user (staff or customer) can manage and purchase items. It allows staff to view and update stock levels, and customers can view stock, buy items, and receive promotional discounts.

### Project Overview:

The system is structured to have two primary user roles:

* **Staff**: Can log using passwords and username (Pre-Defined in the program) in and manage the stock by updating prices and quantities. Which further supports five different users (respectively Group members)
* **Customer**: Can browse products, make purchases, and receive discounts based on the total amount spent and in the end receive a transcript.

### Program Login Following are the Login credential for the program

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project By : Alpha Squad | | | | |
| Sr.no | **Name** | **Role - Position** | **Username**  **(Case independent)** | **Password** |
| 1 | Member 1 | Manager (GM) | Alpha1 | 1122Z |
| 2 | Member 2 | Shop - Owner | Alpha2 | 1122K |
| 3 | Member 3 | Operational Head | Alpha3 | 1122A |
| 4 | Member 4 | Sales Director | Alpha4 | 1122F |
| 5 | Member 5 | Tech Consultant | Alpha5 | 1122AB |

### 

# Project Details:

#### **Programming Language:** **C++**

#### **Libraries Used:**

1. **<iostream>**: For handling input and output.
2. **<string>**: For string manipulation (e.g., item names).
3. **<sstream>**: To handle item information for generating invoices.
4. **<algorithm>**: For case-insensitive comparisons and transformations.
5. **<iomanip>**: For formatted output, particularly in generating invoices
6. **<fstream>**: For file handling to save stock updates and log customer purchases.
7. **<conio.h>**: Provides utility functions such as **getch()**.

##### **Product Information (Predefined but Editable):**

1. **products[ ]:**Array of Product structures to store names, prices, and quantities of items.
2. **productCount:** Number of predefined products (currently 5).

##### **User Inputs:**

1. **customerName**: Stores the name of the customer for generating invoices.
2. **itemsBought, quantitiesBought, pricesBought**: Strings to store purchase details
3. **choice1:** Used for navigating within the stock update menu.
4. **itemName:** Stores the name of the item for buying or updating.
5. **buyQuantity:** Stores the quantity of items the customer wants to buy.
6. **username** and **password:** For staff login.
7. **userType:** Identifies the user type (staff or customer).
8. **totalAmount:** Tracks the total amount spent by the customer for calculating discounts.

#### **New Functionalities:**

1. **Structure Definitions:**
   * Product: To hold product details like name, price, and quantity.
   * Purchase: To record customer purchase details for invoices and logs.
2. **File Handling:**
   * **saveStockToFile() and logCustomerPurchase()** function to save details.
3. **Staff Login Enhancements:**
   * Role-based login with predefined usernames and passwords for different staff members (e.g., Manager, Owner, Operational Head).
4. **Improved Stock Management:**
   * Separate submenus for updating quantities and prices.
   * Case-insensitive handling of product names.
5. **Invoice Generation:**
   * String concatenation to build an invoice with item names, quantities, and prices.
   * Discount calculation and total bill display.
6. **Password Masking:**
   * Use of a custom function or external library to mask password input, making it appear as **\*\*\*\*\*** during entry

# Program Flow:

1. **Main Interface**: A simple menu-driven system where users select options to either view stock, update stock, or buy items.
2. **Looping Menus**: The menus will keep displaying until the user chooses to exit the system.

### **Key Functionalities:**

* **Staff Operations**:
  + **Login**: Staff can log in with a predefined username and password.
  + **View Stock**: Displays current stock levels for pens, notebooks, and pencils.
  + **Update Stock**: Staff can update the quantity or price of the items.
* **Customer Operations**:
  + **View Stock**: Customers can view the available stock.
  + **Buy Item**: Customers can select items to buy, with a discount applied based on the total cost of their purchases.
* **Invoice Generation**:   
  After a customer finishes shopping, the system generates an invoice showing:
  + Purchased items.
  + Quantities and prices of those items.
  + A discount (if applicable).
  + Total payable amount.

### **Discount Logic:**

* A **5% discount** is applied for purchases over Rs. 1000.
* A **10% discount** is applied for purchases over Rs. 2500.

### **Error Handling:**

* **Invalid Item Selection**: The program checks if the user inputs valid item names or numbers.
* **Insufficient Stock**: If the user tries to buy more than what is available in stock, an error is displayed.
* **Invalid Menu Selection**: Only valid choices (1, 2, 3, etc.) are accepted in the menu system.

# Program Structure:

The algorithm uses nested loops to manage user navigation through menus, ensuring modularity and efficient flow control.

* **Separation of Concerns:** Separate logic for staff and customers.
* **User-Friendly Design:** Clear prompts for input.
* **Data Persistence:** Updated stock is saved to a file for future sessions.
* **Scalability:** Easy to add new functionality (e.g., staff roles or customer discounts).

**Algorithm:**

**1. Start**

1. Display the welcome screen with system details.

**2. Main Menu**

1. Prompt the user to choose their role:
   * 1 for Staff.
   * 2 for Customer.
   * 3 to End Program.
2. If the user chooses **1** (Staff):
   * Go to **Staff Login**.
3. If the user chooses **2** (Customer):
   * Go to **Customer Menu**.
4. If the user chooses **3**:
   * Exit the program.

**3. Staff Login**

1. Prompt for username and password.
2. Convert the username to lowercase for uniformity.
3. Validate credentials:
   * If valid, display a welcome message and proceed to the **Staff Menu**.
   * If invalid, display an error message and prompt again.

**4. Staff Menu**

1. Display options:
   * 1 to View Stock.
   * 2 to Update Stock.
   * 3 to Exit.
2. If the user chooses **1**:
   * Display the current stock (names, prices, and quantities).
3. If the user chooses **2** (Update Stock):
   * Prompt to choose between:
     + Update Quantity.
     + Update Price.
     + Exit
   * For updating quantity:
     1. Display the list of items.
     2. Prompt the user to select an item.
     3. Ask for the new quantity and update the stock.
   * For updating price:
     1. Display the list of items.
     2. Prompt the user to select an item.
     3. Ask for the new price and update the stock.
4. If the user chooses **3**:
   * Return to the main menu.

**5. Customer Menu**

1. Prompt for the customer's name.
2. Display the available stock (names, prices, and quantities).
3. Allow the customer to:
   * Select items to buy.
   * Enter the quantity for each item.
   * Update the total amount.
4. Calculate the total bill with any applicable discounts.
5. Display the invoice:
   * Items bought.
   * Quantities.
   * Prices.
   * Total amount.
   * Discount (if any).
6. Log the purchase details using the **logCustomerPurchase** function.
7. Update stock quantities based on items purchased.

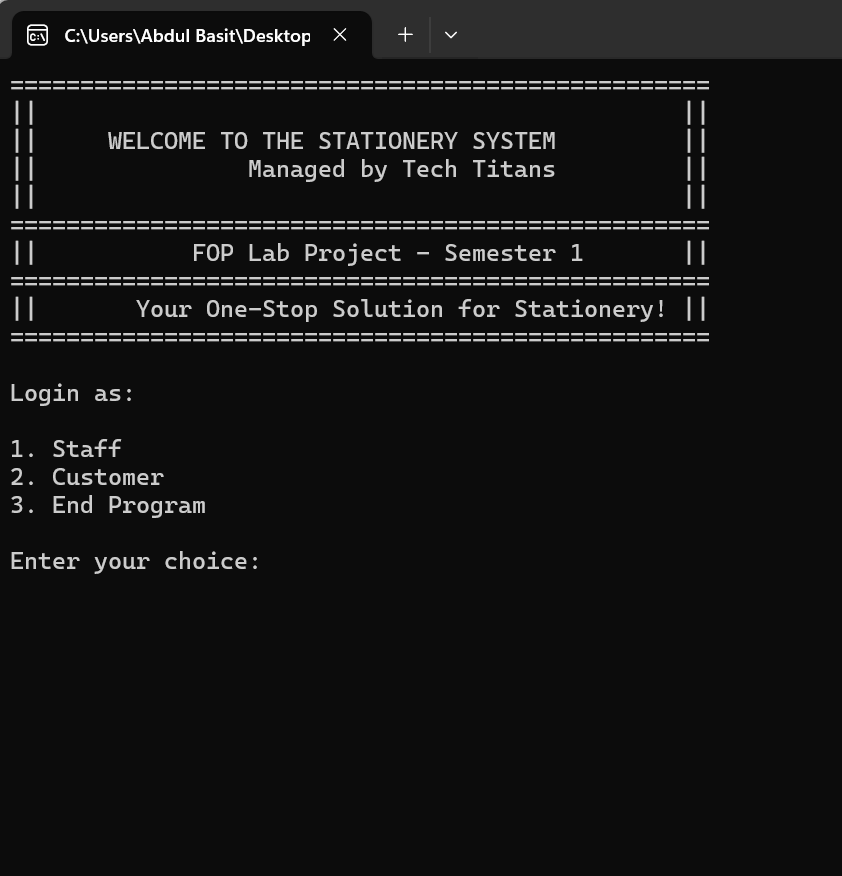
**6. Save and Log Operations**

1. Save the updated stock details to a file using the ***saveStockToFile*** function.
2. Log customer purchases for future reference using the ***logCustomerPurchase*** function.

**7. End Program**

1. Display a thank-you message.
2. Terminate the program.

# Output Screen:

****

# **Conclusion**:

The **Stationery Management System** prototype effectively demonstrates basic functionalities for both **staff** and **customers**. Staff can manage stock by updating prices and quantities, while customers can browse products, make purchases, and receive discounts based on their total spending.

The system uses fundamental programming concepts such as:

* **Input/Output** for user interaction.
* **Conditionals** for decision-making, like applying discounts.
* **Loops and Nested Structures** to handle user choices and stock updates.

Though simple, this project provides a solid foundation for further development, showcasing essential programming skills and offering a useful tool for managing inventory and sales

# ****References:****

1. **W3Schools** – C++ Tutorial. https://www.w3schools.com/
2. **YouTube** – Mosh Hamdani C++ Tutorials.
3. **GeeksforGeeks** – C++ Programming Basics.
4. **Stack Overflow** – C++ Troubleshooting and Solutions.
5. **C++ Documentation** – C++ Reference.