STATIONARY SHOP INVENTORY MANAGEMENT SYSTEM

PROJECT DESCRIPTION

1. AIM OF THE PROJECT:

Inventory Management System aims to provide a user-friendly and robust solution for managing inventory and sales operations within a stationery shop.

1. **Effortless Inventory Management:** Empower shopkeepers to precisely track stock levels, ensuring items are consistently available for customers without unnecessary stockouts or overstocking.

2. **Seamless Sales Tracking:** Facilitate the recording of sales transactions to maintain a clear record of sold items and monitor sales performance over time.

3. **Simplified Operations:** Offer an intuitive interface where shopkeepers can effortlessly perform tasks such as adding new items, restocking existing items, and viewing current stock levels in real-time.

4. **Data-Driven Decision Making:** Generate valuable insights into inventory trends and sales patterns, allowing shopkeepers to make informed decisions regarding stock replenishment and business strategies.

5. **Enhanced Customer Service:** Guarantee items are readily available for customers, minimizing wait times and improving overall customer satisfaction.

2. PROBLEM STATEMENT:

The current inventory and sales management practices in many stationery shops are inefficient and prone to errors.

1. **Stock Discrepancies:** Manual inventory tracking often results in inaccuracies, leading to stockouts or overstocking of items.
2. **Sales Tracking Hurdles:** Difficulty in keeping track of sales transactions and monitoring which items are selling well or poorly.
3. **Time-Consuming Operations:** Cumbersome processes for adding new items to inventory, restocking items, and manually updating stock levels.
4. **Impact on Customer Service:** Inconsistent availability of items may lead to dissatisfied customers and lost sales opportunities.
5. **Limited Data Insights:** Inability to analyze historical sales data to forecast demand, optimize inventory levels, and make informed business decisions.
6. **Error-Prone Manual Processes:** Dependence on manual calculations and recordkeeping increases the likelihood of errors in stock management and sales reporting.

Problem Statement:

"Our current inventory and sales management at [Shop Name] are manual and error-prone. We face challenges in accurately tracking inventory levels, recording sales transactions, and maintaining consistent stock availability. This negatively impacts our ability to provide excellent customer service and optimize operational efficiency. We require a robust system that automates these processes, improves inventory accuracy, provides real-time insights into sales performance, and enhances our overall business operations."

3. ERROR HANDLING AND EXCEPTION HANDLING:

Streamlined Stationery incorporates robust error handling and exception handling mechanisms to ensure reliable operation by accommodating various user interactions and gracefully managing unexpected scenarios.

Flexible User Inputs: The system accepts menu-driven choices, full item names, and predefined short forms for item names, enhancing ease of use.

Quantity Input Validation: Prompts for quantities during operations like selling, restocking, or adding items, ensuring accurate data entry.

Effective Error Handling Mechanisms: Efficiently addresses potential issues such as:

- **Invalid Menu Choices:** Guides users to correct input mistakes for selecting valid menu options.
- **Invalid Item Names:** Alerts users if an item name or abbreviation doesn't match existing inventory, prompting correction.
- **Insufficient Stock:** Notifies users of inadequate stock levels when attempting to sell or restock items, suggesting appropriate actions.
- **Quantity Validation:** Ensures that entered quantities are positive integers, preventing errors due to incorrect data types.

Exception Handling:

- **ValueError Handling:** Utilizes try-except blocks to catch errors arising from non-integer inputs for quantities, providing clear instructions for valid input.
- **KeyError Handling:** Manages exceptions when accessing dictionaries for item names, ensuring operations proceed smoothly.
- **General Exception Handling:** Safeguards against unforeseen errors with a broad except block, offering informative messages and guiding users through resolution steps.

4. Code Implementation:

Certainly! Below is a simplified implementation of the Stationary Shop Inventory Management System focusing on input versatility, error handling, and exception handling. This implementation covers essential functionalities such as displaying stock, selling items, displaying sales, adding new items, restocking items, and handling various types of user inputs and errors.

CODE:

```
stock = {
    'pen': 150,
    'pencil': 175,
    'notebook': 100,
    'geometry box': 120,
    'scale': 140
}

sales = {
    'pen': 0,
    'pencil': 0,
    'notebook': 0,
    'geometry box': 0,
    'scale': 0
}
```

```
short_forms = {
    'pe': 'pen',
    'p': 'pencil',
    'n': 'notebook',
    'g': 'geometry box',
    's': 'scale'
}

def display_stock():
    print("\nCurrent Stock:")
    for item, quantity in stock.items():
        print(f"{item}: {quantity}")
    print()

def sell_item(item, quantity):
    try:
        if item in stock:
            if quantity > 0:
                if stock[item] >= quantity:
                    stock[item] -= quantity
                    sales[item] += quantity
                    print(f"\n{quantity} {item}(s) sold.")
                    print(f"Remaining stock of {item}: {stock[item]}")
```

```
            else:
                print(f"\nNot enough {item} in stock. Only {stock[item]} available.")
        else:
            print("\nQuantity should be greater than 0.")
    elif item in short_forms:
        full_item_name = short_forms[item]
        if quantity > 0:
            if stock[full_item_name] >= quantity:
                stock[full_item_name] -= quantity
                sales[full_item_name] += quantity
                print(f"\n{quantity} {full_item_name}(s) sold.")
                print(f"Remaining stock of {full_item_name}: {stock[full_item_name]}")
            else:
                print(f"\nNot enough {full_item_name} in stock. Only {stock[full_item_name]} available.")
        else:
            print("\nQuantity should be greater than 0.")
    else:
        print("\nInvalid item.")
  except ValueError:
    print("\nPlease enter a valid quantity (must be a positive integer).")
  except KeyError as e:
    print(f"\nError: {str(e)}. Please enter a valid item.")
```

```
    except Exception as e:
        print(f"\nAn unexpected error occurred: {str(e)}. Please try again later.")


def display_sales():
    print("\nTotal Sales:")
    for item, quantity_sold in sales.items():
        print(f"{item}: {quantity_sold}")
    print()


def add_item(item, quantity):
    try:
        if item not in stock:
            if quantity > 0:
                stock[item] = quantity
                sales[item] = 0
                print(f"\nAdded {item} with quantity {quantity}.")
            else:
                print("\nQuantity should be greater than 0.")
        else:
            print("\nItem already exists. Use restock to add more quantity.")
    except ValueError:
        print("\nPlease enter a valid quantity (must be a positive integer).")
    except Exception as e:
```

```
        print(f"\nAn unexpected error occurred: {str(e)}. Please try again later.")


def restock_item(item, quantity):
    try:
        if item in stock:
            if quantity > 0:
                stock[item] += quantity
                print(f"\nRestocked {item} with quantity {quantity}.")
                print(f"New stock of {item}: {stock[item]}")
            else:
                print("\nQuantity should be greater than 0.")
        else:
            print("\nItem does not exist. Use add to add a new item.")
    except ValueError:
        print("\nPlease enter a valid quantity (must be a positive integer).")
    except Exception as e:
        print(f"\nAn unexpected error occurred: {str(e)}. Please try again later.")


while True:
    print("Stationary Shop Inventory Management System\n")
    print("1. Display Stock")
    print("2. Sell Item")
    print("3. Display Sales")
```

```
print("4. Add New Item")
print("5. Restock Item")
print("6. Exit")
choice = input("Enter your choice (1-6): ")

if choice == '1':
    display_stock()
elif choice == '2':
    item = input("Enter item name or short form (pe, p, n, g, s): ").lower()
    try:
        quantity = int(input("Enter quantity to sell: "))
        sell_item(item, quantity)
    except ValueError:
        print("\nPlease enter a valid quantity (must be a positive integer).")
elif choice == '3':
    display_sales()
elif choice == '4':
    item = input("Enter new item name: ").lower()
    try:
        quantity = int(input("Enter quantity to add: "))
        add_item(item, quantity)
    except ValueError:
        print("\nPlease enter a valid quantity (must be a positive integer).")
```

```
elif choice == '5':
    item = input("Enter item name to restock: ").lower()
    try:
        quantity = int(input("Enter quantity to restock: "))
        restock_item(item, quantity)
    except ValueError:
        print("\nPlease enter a valid quantity (must be a positive integer).")
elif choice == '6':
    print("Shop Closed.")
    break
else:
    print("\nInvalid choice. Please enter a number from 1 to 6.")
```

5. Measurable Results and Outcomes

Implementing the Streamlined Stationery system delivers significant and measurable results, positively impacting a stationery shop's key performance indicators:

- **Reduced Stock Discrepancies:** Real-time inventory tracking minimizes stockouts and overstocking, leading to significantly improved inventory accuracy.
- **Enhanced Sales Visibility:** Streamlined sales recording allows for clear identification of top-selling items, enabling better product selection and potentially resulting in a stronger sales performance.
- **Improved Operational Efficiency:** Automated tasks and a user-friendly interface free up employee time, leading to increased employee productivity in managing inventory.
- **Boosted Customer Satisfaction:** Consistent item availability and faster checkout times can lead to a noticeable improvement in customer satisfaction.
- **Data-Driven Decision Making:** Sales analytics empower shopkeepers to optimize stock levels, identify sales trends, and make informed purchasing decisions, potentially resulting in substantial cost savings and improved profitability.

6. Conclusion

Stationery shops, in today's dynamic market, often struggle with managing inventory and gaining valuable customer insights. Streamlined Stationery transcends these challenges by empowering your shop with the tools to achieve significant improvements in efficiency, like faster checkouts and reduced stockouts, leading to happier customers and a boost in profitability. By leveraging accurate inventory data and actionable sales insights, Streamlined Stationery unlocks the potential for a competitive edge, optimized operations, and sustainable growth.