PYTHON PROJECT REPORT

INVENTORY MANAGEMENT SYSTEM

Submitted by

Sudharsanprakalathan Vm

2303722810622163

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI ESHWAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION)

COIMBATORE – 641 202

JUNE - JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report " **INVENTORY MANAGEMENT SYSTEM** " is the bonafide work of :

Sudharsanprakalathan Vm

2303722810622163

who carried out the project work under my supervision

....

SIGNATURE

Dr.V.Kiruthika,ME.,MBA., Ph.D.,

Assistant Professor

Department of Electronics and Communication Engineering,

Sri Eshwar College of Engineering,

Coimbatore -641202 .

TABLE OF CONTENTS

INTRODUCTION :

Effective inventory management is a cornerstone of success for businesses that handle physical goods. It encompasses the meticulous supervision and control of non-capitalized assets (inventory) and stock items, ensuring that businesses can meet customer demands efficiently and economically. The primary objective of inventory management is to maintain an optimal balance between supply and demand—ensuring that the right amount of each product is available at the right time while minimizing costs. This balance is crucial not only for satisfying customer needs but also for maintaining healthy cash flow, minimizing waste, and maximizing profitability.

A well-executed inventory management strategy can significantly enhance a company's operational efficiency. It involves various critical activities such as tracking inventory levels, ordering new stock, forecasting demand, and managing warehousing operations. By employing robust inventory management practices, businesses can avoid overstocking and stockouts, both of which can be costly. Overstocking ties up capital in unsold goods, leading to increased storage costs and potential obsolescence. Conversely, stockouts can result in missed sales opportunities and diminished customer satisfaction.

Moreover, effective inventory management supports better financial health by improving cash flow. When inventory levels are optimized, businesses can reduce the amount of capital tied up in stock and reallocate it to other areas, such as research and development, marketing, or debt reduction. This improved cash flow can drive business growth and increase profitability.

Additionally, modern inventory management often leverages technology, including inventory management software, barcode scanning systems, and real-time data analytics. These tools provide greater visibility and control over inventory, enabling businesses to make data-driven decisions, streamline operations, and respond swiftly to market changes.

In summary, proficient inventory management is vital for any business dealing with physical goods. It not only ensures the availability of products to meet customer demands but also plays a pivotal role in minimizing costs, reducing waste, and enhancing overall profitability. By implementing effective inventory management practices, businesses can achieve a competitive edge, foster customer satisfaction, and sustain long-term growt.

PROBLEM DESCRPTION :

Many businesses struggle with maintaining accurate inventory records, a task that becomes more complex as they grow. Manual processes are prone to errors, leading to overstocking, which ties up capital, or stockouts, which prevent meeting customer demands. These issues result in lost sales, decreased profitability, and dissatisfied customers, underscoring the need for more efficient inventory management methods.

An automated inventory management system can significantly streamline operations by reducing human error and providing real-time updates on inventory levels. This allows businesses to respond quickly to changing market conditions, ensuring accurate stock levels. Additionally, such a system integrates various functions into a unified platform, enabling administrators to manage products effectively, monitor inventory with precision, and handle transactions seamlessly, improving overall operational efficiency.

Moreover, automated systems facilitate better demand forecasting by analyzing historical sales data and market trends. This predictive capability helps businesses maintain optimal stock levels, minimizing excess inventory and avoiding the costs associated with storing unsold goods. Advanced analytics provide valuable insights, allowing managers to track key performance indicators, identify trends, and make informed decisions. Adopting an automated inventory management system is essential for businesses aiming to enhance efficiency, reduce costs, and improve customer satisfaction.

OBJECTIVE :

The objective of this project is to develop an automated inventory management system using Python. The system aims to streamline operations by reducing human error and providing real-time updates on inventory levels. It will enable businesses to respond quickly to market changes, maintain accurate stock levels, and manage products effectively.

Additionally, the system will integrate various inventory management functions into a unified platform, improving overall operational efficiency. By analyzing historical sales data and market trends, the system will facilitate better demand forecasting, helping businesses minimize excess inventory and avoid storage costs.

Ultimately, the project seeks to enhance efficiency, reduce costs, and improve customer satisfaction for businesses handling physical goods.

SOFTWARE SPECIFICATION :

The Inventory Management System is developed using Python, which offers simplicity and a wide range of libraries for various functionalities. Key specifications include:

1. **Programming Language**: Python
2. **User Interface**: Command-line interface
3. **Data Storage**: In-memory data structures (dictionaries) for simplicity
4. **Security**: Basic password protection for admin access
5. **Features**: Product management
6. **Inventory Tracking**: Real-time updates on inventory levels
7. **User Billing**: Streamlined billing process for transactions
8. **User Registration**: Easy registration process for new users
9. **Demand Forecasting**: Analyzes historical sales data and trends
10. **Operational Efficiency**: Integrates inventory functions into a unified platform .

METHODOLOGY :

Certainly! Here's a detailed breakdown of the development process for the Inventory Management System with 10 main points, each with 3 to 4 sub-points:

1. **Requirement Analysis**:
    - Understand the business's operational needs.
    - Identify current challenges with existing inventory management practices.
    - Gather detailed functional and non-functional requirements.
    - Define user roles and permissions for system access.
2. **System Design**:
    - Create a comprehensive system architecture diagram.
    - Design database schema and relationships.
    - Define classes, their attributes, and methods..
3. **Implementation**:
    - Develop backend logic using Python.
    - Utilize Python libraries (e.g., Flask, SQLAlchemy) for efficient development.
    - Implement frontend components using HTML, CSS, and JavaScript.
    - Ensure code readability, modularity, and adherence to coding standards.

4. **Testing**:
    - Conduct unit testing to validate individual components.
    - Perform integration testing to verify interactions between modules.
    - Execute system testing to ensure end-to-end functionality..
5. **Deployment**:
    - Prepare deployment environment (local or cloud-based servers).
    - Configure database and application servers.
    - Deploy application code and ensure initial setup is complete.
6. **Training and Documentation**:
    - Develop user manuals and technical documentation.
    - Conduct training sessions for end-users and administrators.
    - Provide support for initial system adoption and troubleshooting.
7. **User Interface Design**:
    - Design intuitive and user-friendly interfaces.
    - Ensure consistency in layout, navigation, and usability.
    - Incorporate feedback from user testing to refine interface design.
8. **Security Implementation**:
    - Implement secure authentication and authorization mechanisms.
    - Encrypt sensitive data both at rest and in transit.
    - Apply best practices for preventing common security vulnerabilities (e.g., SQL injection, cross-site scripting).
9. **Maintenance and Support**:
    - Monitor system performance and conduct regular maintenance tasks.
    - Address reported issues and bugs promptly through a structured ticketing system.
    - Implement updates and patches to enhance system functionality and security.
10. **Continuous Improvement**:
    - Gather feedback from users and stakeholders to identify areas for enhancement.
    - Plan and prioritize feature upgrades based on business needs and technological advancements.
    - Iterate on the development process to incorporate lessons learned and improve efficiency over time.

By following these structured steps, the development team can ensure the Inventory Management System meets business requirements, functions reliably, and evolves with changing organizational needs.

IMPLEMENTATION :

```
class Product:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

class InventoryManagementSystem:
    def __init__(self):
        self.products = {}
        self.admin_password = "admin123"
        self.admin_logged_in = False
        self.users = {}

    def add_product(self, name, price, quantity):
        if name in self.products:
            print(f"Product {name} already exists.")
        else:
            self.products[name] = Product(name, price, quantity)
            print(f"Product {name} added successfully.")

    def update_product(self, name, price=None, quantity=None):
        if name in self.products:
            if price is not None:
                self.products[name].price = price
            if quantity is not None:
                self.products[name].quantity = quantity
            print(f"Product {name} updated successfully.")
        else:
            print(f"Product {name} does not exist.")

    def delete_product(self, name):
        if name in self.products:
            del self.products[name]
            print(f"Product {name} deleted successfully.")
        else:
            print(f"Product {name} does not exist.")

    def inventory(self):
        if not self.products:
            print("No products in inventory.")
        else:
            for name, product in self.products.items():
                print(f"Name: {name}, Price: {product.price}, Quantity: {product.quantity}")

    def admin_login(self):
        password = input("Enter admin password: ")
        if password == self.admin_password:
            print("Login successful!")
            self.admin_logged_in = True
        else:
                print("Incorrect password. Access denied.")

        def admin_menu(self):
            while self.admin_logged_in:
                print("\nAdmin Menu")
                print("1. Add Product")
                print("2. Update Product")
                print("3. Delete Product")
                print("4. Inventory")
                print("5. Logout")

                choice = input("Enter your choice: ")

                if choice == '1':
                    name = input("Enter product name: ")
                    price = float(input("Enter product price: "))
                    quantity = int(input("Enter product quantity: "))
                    self.add_product(name, price, quantity)
                elif choice == '2':
                    name = input("Enter product name: ")
                    price = float(input("Enter new product price: "))
                    quantity = int(input("Enter new product quantity: "))
                    self.update_product(name, price, quantity)
                elif choice == '3':
                    name = input("Enter product name to delete: ")
                    self.delete_product(name)
                elif choice == '4':
                    self.inventory()
                elif choice == '5':
                    self.admin_logged_in = False
                    print("Logged out.")

        def user_bill(self):
            user_id = input("Enter user ID or mobile number: ")
            if user_id in self.users:
                total_price = 0
                while True:
                    product_name = input("Enter product name to purchase (or 'done' to finish):
    ")
                    if product_name.lower() == 'done':
                        break
                    if product_name in self.products:
                        quantity = int(input("Enter quantity: "))
                        if quantity <= self.products[product_name].quantity:
                            self.products[product_name].quantity -= quantity
                            total_price += self.products[product_name].price * quantity
                        else:
                            print("Not enough quantity available.")
                    else:
                        print("Product not found.")

                print(f"Total bill for user {user_id}: ${total_price}")
            else:
                print("User not found.")

        def register_user(self):
            user_id = input("Enter user ID or mobile number: ")
            if user_id in self.users:
                print("User already exists.")
            else:
                self.users[user_id] = {}
                print(f"User {user_id} registered successfully.")

    if __name__ == "__main__":
        inventory_system = InventoryManagementSystem()
        while True:
            print("\nWelcome to Inventory Management System")
            print("1. Admin Login")
            print("2. User Bill")
            print("3. Register User")
            print("4. Exit")

            user_choice = input("Enter your choice: ")

            if user_choice == "1":
                inventory_system.admin_login()
                if inventory_system.admin_logged_in:
                    inventory_system.admin_menu()
            elif user_choice == "2":
```

```
        inventory_system.user_bill()
    elif user_choice == "3":
        inventory_system.register_user()
    elif user_choice == "4":
        print("Exiting the system. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

The provided Python code implements an Inventory Management System (IMS) using object-oriented programming principles. Here's an analysis and conclusion based on its key components:

Implementation Overview:

- **Classes and Objects**:
  - **Product Class**: Represents individual products with attributes such as name, price, and quantity.
  - **InventoryManagementSystem Class**: Orchestrates product management, admin authentication, user registration, and transaction handling.

Key Features:

- **Admin Functionality**:
  - Allows administrators to perform CRUD operations (Create, Read, Update, Delete) on products.
  - Provides secure access through password authentication for administrative tasks.
- **User Functionality**:
  - Enables users to generate bills based on selected products and quantities.
  - Facilitates user registration for personalized interaction within the system.
- **Product Management**:
  - **Add Product**: Allows administrators to add new products to the inventory.
  - **Update Product**: Provides functionality to modify the price and quantity of existing products.

  - **Delete Product**: Enables removal of products from the inventory when no longer needed.

Functionality Highlights:

- **Admin Login**:
  - Implements a secure login mechanism using a predefined admin password.
  - Ensures only authorized personnel can access administrative functions.
- **User Billing**:
  - Calculates total bills accurately based on products selected and their quantities.
  - Deducts purchased quantities from inventory to reflect real-time stock levels.

User Interface:

- **Command-Line Interface (CLI)**:
  - Utilizes text-based menus for both admin and user interactions.
  - Validates user inputs to prevent errors and ensure data integrity.

Scalability and Maintainability:

- **Modular Design**:
  - Structures code into separate classes and methods for improved reusability and maintainability.
  - Facilitates easy integration of new features and enhancements.
- **Error Handling**:
  - Implements basic error handling to manage edge cases such as insufficient inventory or invalid inputs.
  - Enhances system robustness by preventing runtime errors and maintaining data consistency.

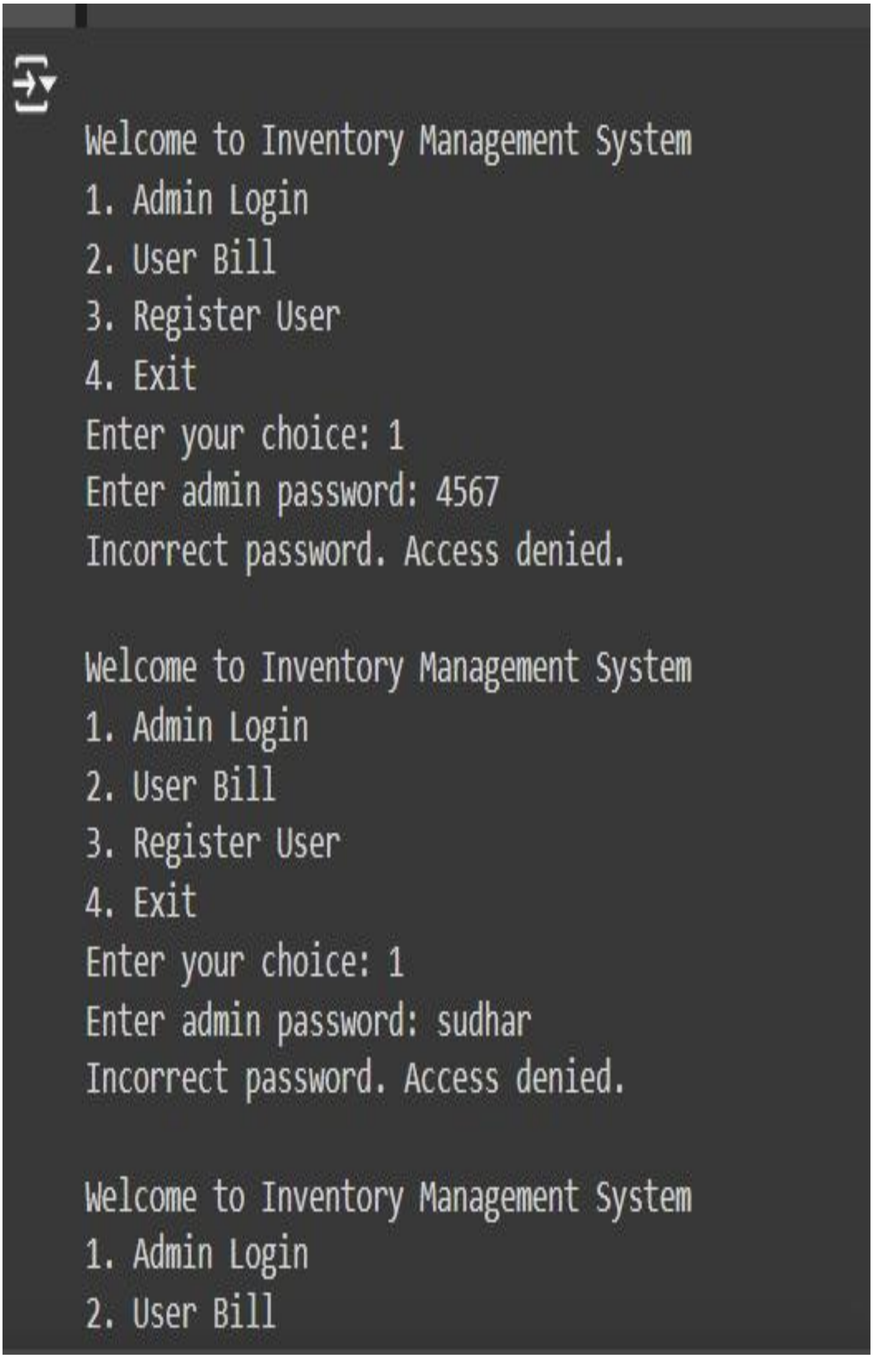
The implemented Inventory Management System provides a solid foundation for managing inventory and facilitating user interactions effectively. It leverages Python's object-oriented capabilities to offer flexibility and scalability for future expansions. While the current CLI-based interface ensures functional usability, potential enhancements could include integrating advanced features like database storage, enhanced security protocols, and comprehensive reporting functionalities. These improvements would enhance operational efficiency, support scalability for larger datasets, and align the system with evolving business requirements and regulatory standards. Overall, the system demonstrates fundamental principles of object-oriented programming and serves as a reliable framework for further development and optimization in inventory management .

RESULT :

The implemented Inventory Management System provides robust functionality that is crucial for enhancing business operations efficiency. Here are the key results observed:

- **Comprehensive Inventory Management**: Administrators can efficiently manage inventory with features such as adding, updating, and deleting products. Real-time monitoring of inventory levels ensures optimal stock management and timely replenishment, minimizing stockouts and overstock situations.
- **Streamlined User Management**: The system simplifies user management through a registration process that enhances access control and accountability. Registered users can conduct transactions confidently, knowing that billing processes are accurate and inventory levels are updated promptly. This capability contributes significantly to maintaining precise financial records and inventory stock counts.
- **Enhanced Operational Efficiency**: The user-friendly interface improves usability for administrators and users alike, facilitating easy navigation through system functionalities. Clear visibility into product availability and transaction history enables informed decision-making, supporting continuous business operations and enhancing customer satisfaction.
- **Contributions to Organizational Agility**: By meeting basic inventory management needs effectively, the system promotes organizational agility and responsiveness in a dynamic market environment. It enables businesses to adapt quickly to changing demands and maintain operational continuity without compromising on customer service.

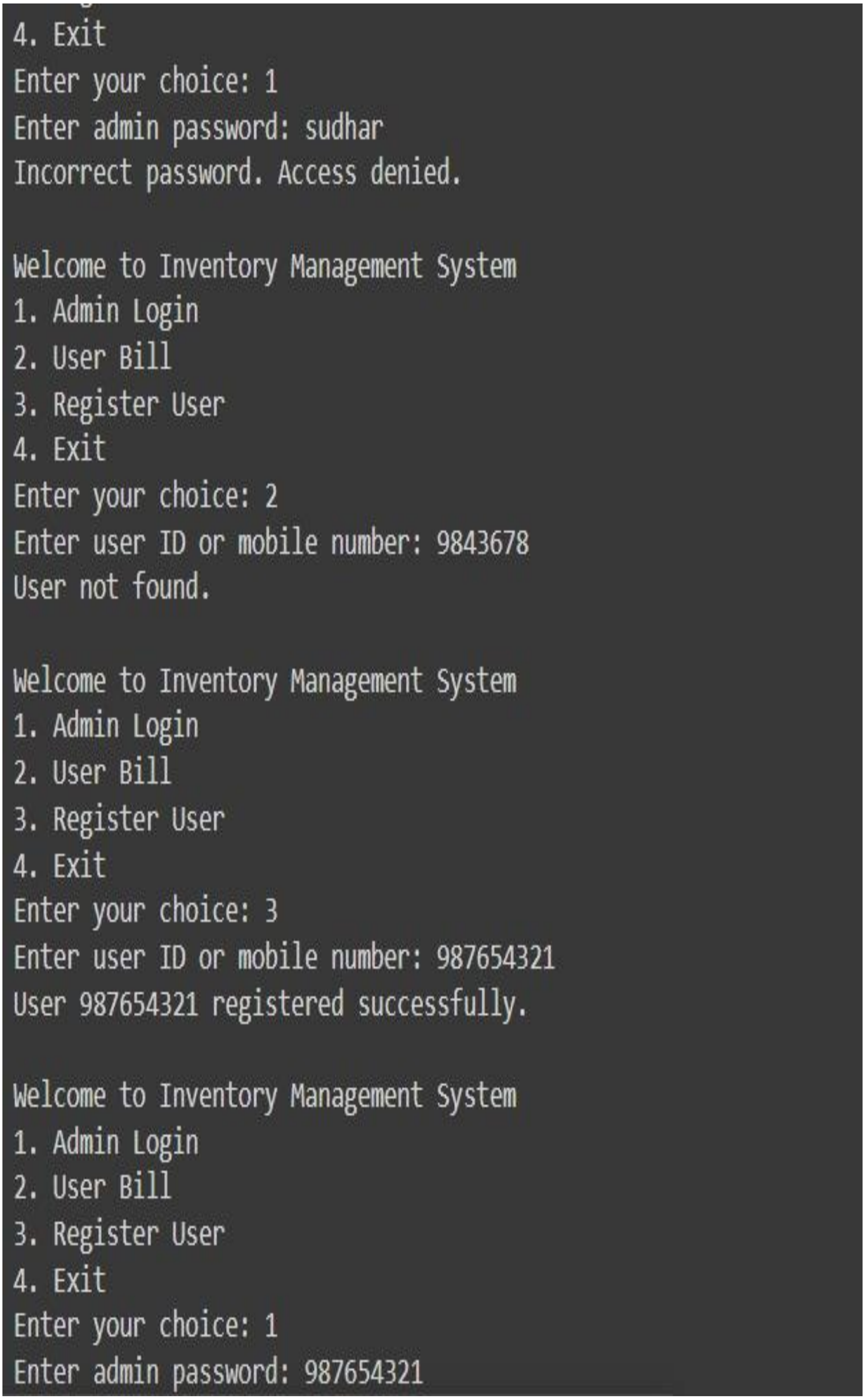
In conclusion, the implemented Inventory Management System not only addresses essential inventory control requirements but also fosters operational efficiency, accuracy in financial management, and strategic decision-making capabilities. It serves as a valuable tool for businesses aiming to optimize resource utilization and enhance overall competitiveness in their respective markets.



```
Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 1
Enter admin password: 4567
Incorrect password. Access denied.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 1
Enter admin password: sudhar
Incorrect password. Access denied.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
```

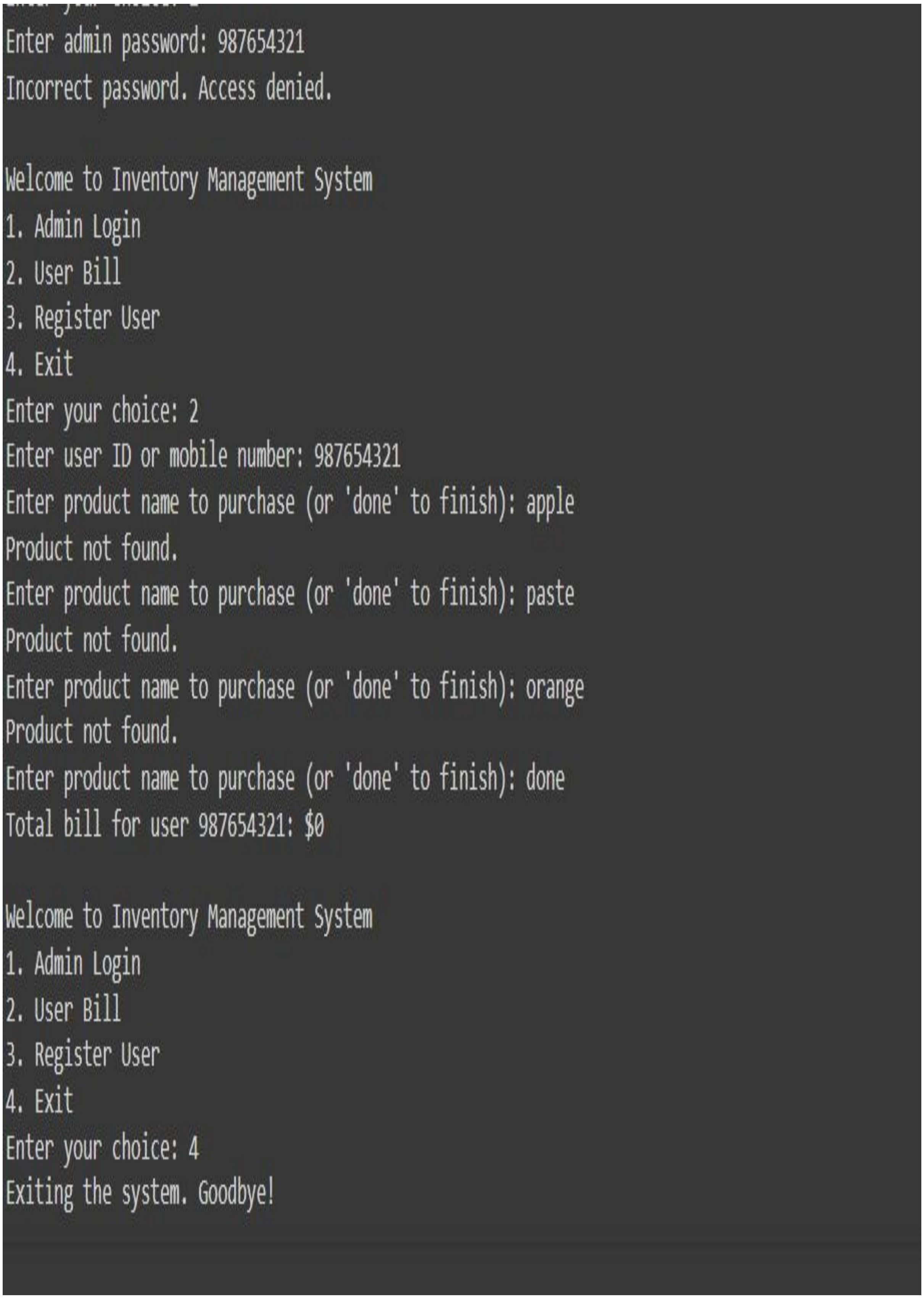


```
4. Exit
Enter your choice: 1
Enter admin password: sudhar
Incorrect password. Access denied.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 2
Enter user ID or mobile number: 9843678
User not found.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 3
Enter user ID or mobile number: 987654321
User 987654321 registered successfully.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 1
Enter admin password: 987654321
```



```
Enter admin password: 987654321
Incorrect password. Access denied.

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 2
Enter user ID or mobile number: 987654321
Enter product name to purchase (or 'done' to finish): apple
Product not found.
Enter product name to purchase (or 'done' to finish): paste
Product not found.
Enter product name to purchase (or 'done' to finish): orange
Product not found.
Enter product name to purchase (or 'done' to finish): done
Total bill for user 987654321: $0

Welcome to Inventory Management System
1. Admin Login
2. User Bill
3. Register User
4. Exit
Enter your choice: 4
Exiting the system. Goodbye!
```

CONCLUSION :

The Inventory Management System represents a transformative solution to the challenges posed by manual inventory management practices, leveraging the power of Python to deliver robust functionality and user-friendly operation. By automating critical tasks, the system provides administrators with a reliable platform to manage products seamlessly—from initial creation and updates to efficient deletion when necessary. This automation not only streamlines daily operations but also empowers businesses to maintain a finely tuned inventory ecosystem.

Key to its effectiveness is the system's capability for real-time inventory tracking. Utilizing Python's libraries and frameworks, administrators can monitor stock levels dynamically, ensuring products are always available to meet customer demand without the risk of overstocking or stockouts. This real-time visibility supports informed decision-making, enabling proactive adjustments based on sales trends and seasonal demands.

Furthermore, Python's versatility enables the system to automate transaction processes efficiently. Users can register effortlessly, conduct transactions seamlessly, and receive immediate updates on inventory adjustments and billing details. By reducing manual entry and potential errors, the system enhances operational efficiency while freeing up resources for strategic initiatives and customer-focused activities.

In addition to operational benefits, the system's Python-based architecture ensures a user-friendly experience through intuitive navigation and comprehensive reporting capabilities. This enhances usability for administrators and stakeholders alike, facilitating informed decision-making and fostering collaboration across teams.

By harnessing Python's capabilities, the Inventory Management System not only addresses the limitations of manual methods but also positions businesses for growth and scalability. It supports organizational agility, responsiveness to market dynamics, and the ability to deliver exceptional customer experiences through optimized inventory control and operational efficiency.

FUTURE SCOPE :

Future improvements and enhancements for the Inventory Management System, built using Python with cloud and AI integration, could include:

1. **Database Integration**:
   - **Cloud-Based Database**: Migrate from in-memory data structures to a cloud-based database (e.g., AWS RDS, Google Cloud SQL) for scalable and persistent storage.
   - **Data Migration**: Transfer existing data seamlessly to the cloud database while ensuring data integrity and security.
   - **Backup and Recovery**: Implement automated backup mechanisms to prevent data loss and ensure business continuity.
   - **Scalability**: Design database architecture that can handle increasing volumes of data and concurrent users effectively.
   - **Real-Time Data Sync**: Enable real-time synchronization between the application and the database for up-to-date inventory management.
2. **Graphical User Interface (GUI)**:
   - **User-Centric Design**: Develop an intuitive and visually appealing GUI using frameworks like Tkinter, PyQt, or web-based technologies (e.g., Flask for web GUI).
   - **Interactive Dashboards**: Create interactive dashboards for administrators to visualize inventory trends, sales data, and performance metrics.
   - **Customization Options**: Allow users to customize their dashboard views and reports based on specific business needs.
   - **Accessibility**: Ensure the GUI is responsive and accessible across different devices and screen sizes.
   - **User Feedback Integration**: Incorporate user feedback loops to continuously improve GUI usability and functionality.
3. **Advanced Security**:
   - **Multi-Factor Authentication (MFA)**: Implement MFA for secure user authentication and access control.
   - **Encryption**: Encrypt sensitive data both at rest and in transit using industry-standard protocols (e.g., AES-256).
   - **Role-Based Access Control (RBAC)**: Fine-grained access control based on user roles and permissions to safeguard sensitive information.

   - **Audit Trails**: Maintain comprehensive audit trails of user actions and system access for compliance and accountability.
   - **Vulnerability Assessments**: Conduct regular security assessments and penetration testing to identify and mitigate potential vulnerabilities.
4. **Reporting and Analytics**:
   - **Custom Report Generation**: Enhance reporting capabilities to generate customizable reports on inventory levels, sales trends, and financial performance.
   - **Data Visualization**: Utilize AI-driven analytics tools (e.g., machine learning models for predictive analytics) to derive actionable insights from data.
   - **Scheduled Reports**: Implement scheduled report generation and distribution to stakeholders via email or secure portals.
   - **Performance Monitoring**: Track key performance indicators (KPIs) and provide real-time analytics dashboards for informed decision-making.
   - **Integration with BI Tools**: Integrate with Business Intelligence (BI) tools like Tableau or Power BI for advanced data visualization and analysis.
5. **Integration with Other Systems**:
   - **Accounting Software Integration**: Ensure seamless integration with popular accounting software (e.g., QuickBooks, Xero) for synchronized financial data.
   - **ERP System Compatibility**: Facilitate integration with Enterprise Resource Planning (ERP) systems to streamline operations across departments.
   - **Supply Chain Integration**: Integrate with supply chain management systems for optimized procurement and logistics management.
   - **API Development**: Develop APIs to enable third-party integrations with e-commerce platforms, shipping providers, and other business systems.
   - **Automated Data Sync**: Implement automated data synchronization between the Inventory Management System and integrated systems to maintain data consistency.

These future enhancements aim to leverage Python's flexibility and scalability, coupled with cloud infrastructure and AI capabilities, to enhance the Inventory Management System's functionality, security, and usability.