GIRLUSH COLLECTIONS

Desktop inventory and order management system

Presented by:

KWAGALAKWE CATHERINE –MAY24/BSE/4308U/F

DECEMBER

Presentation outline

- Executive summary
- Problem statement
- Project objectives
- System overview
- System architecture
- System features
- technologies
- User interface
- Testing and results
- Challenges and solutions
- Future enhancements
- conclusion

Executive summary

Girlush Collections is a comprehensive desktop-based inventory and order management system designed specifically for retail businesses dealing with fashion accessories, particularly bags. The application provides a dual-interface solution: a customer-facing module for browsing products and placing orders, and an administrative panel for managing inventory, processing orders, and generating business analytics.

- **Key Achievements:**
- Fully functional inventory management system
- Customer order processing and tracking
- Real-time stock level monitoring
- Comprehensive reporting and analytics
- User-friendly graphical interface
- Robust database management

Problem statement

- **Problem Statement**
- Traditional manual inventory management systems suffer from:
- Time-consuming data entry and retrieval
- Human error in stock tracking
- Difficulty in generating business insights
- Lack of real-time order status updates
- Poor customer order tracking

Problem objectives

Primary Objectives	**2 Secondary Objectives**
Inventory Management Add, edit, and delete products Track stock levels automatically Set low stock alerts Categorize products efficiently Order Processing Accept customer orders Update order status in real-time Track order history Generate order reports Stock Control Monitor inventory levels Alert on low stock items Support multiple stock operations Maintain stock history Business Analytics Generate sales reports Track customer behavior	Provide intuitive user interface Ensure data integrity and security Enable easy backup and recovery Support future scalability

System overview

- **Application Type:** Desktop Application (Standalone)
- **Architecture:** Model-View-Controller (MVC)
- **Database:** SQLite (Embedded)
- **Framework:** Python with Tkinter
- **Dual Interface:** Customer Module + Admin Module

System architecture

Architecture Pattern	**Component Breakdown**
Model-View-Controller (MVC) with Object-Oriented Programming The system follows the Model–View–Controller (MVC) architecture combined with Object-Oriented Programming principles. This approach separates user interface design, business logic, and data management, making the system easier to maintain and extend. Model: Represents application data and database structure View: Handles user interaction and graphical interface	Controllers (Business Logic Layer) Handle all business operations Manage database connections Validate data Execute CRUD operations Views (Presentation Layer) Display data to users Capture user input Provide interactive interface Update based on data changes Models (Data Layer) Store application data Maintain data relationships Ensure data integrity

System features

- **Customer Module:**
- Product Browsing
- Order Placement
- Order Tracking
- Product Search

Admin Module:

- Inventory Management
- Order Processing
- Stock Updates
- Reports & Analytics

Technologies used

Programming Language
Python 3.7+
Reason: Cross-platform, extensive libraries, rapid development
GUI Framework
Tkinter
Built-in Python library
Cross-platform compatibility
Lightweight and fast
Native look and feel

Database
SQLite3
Serverless, zero-configuration
Lightweight and fast
Single-file database
Built-in Python support
Additional Libraries
Pillow (PIL): Image processing
datetime: Date/time operations
os/sys: File system operations
Development Tools
IDE: Visual Studio Code
Version Control: Git
Database Viewer: DB Browser for SQLite

User Interface Design

Design Principles
Simplicity: Clean, uncluttered interfaces
Consistency: Uniform design across all modules
Feedback: Immediate response to user actions
Accessibility: Easy-to-read fonts and colors
Efficiency: Minimal clicks to complete tasks
Color Scheme
Primary Blue: #3498db (Information, primary actions)
Success Green: #27ae60 (Confirmations, success states)
Warning Orange: #f39c12 (Warnings, pending states)
Danger Red: #e74c3c (Errors, critical alerts)
Purple: #9b59b6 (Analytics, reports)
Dark Gray: #2c3e50 (Headers, navigation)
Light Gray: #ecf0f1 (Backgrounds)

Interface Components
Navigation:
Sidebar navigation for admin dashboard
Tab-based navigation for reports
Breadcrumb navigation where applicable
Data Display:
Tree view tables for data listing
Cards for statistics display
Forms for data entry
Modal dialogs for confirmations
Interactive Elements:
Buttons with hover effects
Search bars with real-time filtering
Dropdown menus for selections
Radio buttons for options

Testing and validation

Testing Types Conducted

- Unit Testing
- Integration Testing
- User Interface Testing
- System Testing

Test Scenarios

- **Inventory Management:** ✓ Add product with valid data ✓ Add product with missing fields ✓ Edit product details ✓ Delete product with confirmation ✓ Search products ✓ Filter by category ✓ Upload product image
- **Order Management:** ✓ Create new order ✓ Update order status ✓ View order details ✓ Search orders ✓ Filter by status ✓ Generate order statistics
- **Stock Updates:** ✓ Add stock to product ✓ Remove stock from product ✓ Set exact stock amount ✓ Low stock alerts ✓ Prevent negative stock

Challenges and solutions

challenges

- **Technical Challenges**
- Problem: Multiple database connections causing locks
- Problem: UI not reflecting database changes immediately
- **2 Design Challenges**
- Problem: Too many clicks to complete tasks
- Problem: Users unsure if actions completed

solutions

- Implemented connection pooling and proper closing
- Added refresh mechanisms after data modifications
- Added quick action buttons and shortcuts
- Added confirmation messages and status indicators

Future enhancements

Planned Features

- Phase 1: Enhanced Security
- Phase 2: Advanced Features
- Phase 3: Cloud Integration
- Phase 4: Business Intelligence

Scalability Considerations

- Migrate to PostgreSQL for larger datasets
- Implement caching mechanisms
- Add API for third-party integrations
- Multi-store support

Conclusion

- **Project Success**
- The Girlush Collections Inventory & Order Management System successfully meets all initial objectives:
- ✓ **Comprehensive Inventory Management** - Full CRUD operations with advanced features ✓ **Efficient Order Processing** - Streamlined workflow from order to delivery ✓ **Real-time Stock Control** - Accurate tracking with automated alerts ✓ **Powerful Analytics** - Business insights through detailed reports
- **14.2 Key Achievements**
- **User-Friendly Interface**: Intuitive design requiring minimal training
- **Robust Architecture**: Scalable OOP design following best practices
- **Data Integrity**: Reliable database operations with validation
- **Business Value**: Immediate operational efficiency improvements
- **14.3 Learning Outcomes**
- Advanced Python programming
- GUI development with Tkinter
- Database design and management
- Software architecture patterns
- Project management
- User experience design
- **14.4 Final Thoughts**
- The system provides a solid foundation for retail business management with room for future growth. The modular architecture allows easy addition of new features, and the clean code structure ensures maintainability.