



## **GIRLUSH COLLECTIONS DESKTOP APP**

**NAME:KWAGALAKWE CATHERINE**

**REG NO:MAY24/BSE/4308U/F**

**COURSE :OBJECT ORIENTED PROGRAMMING**

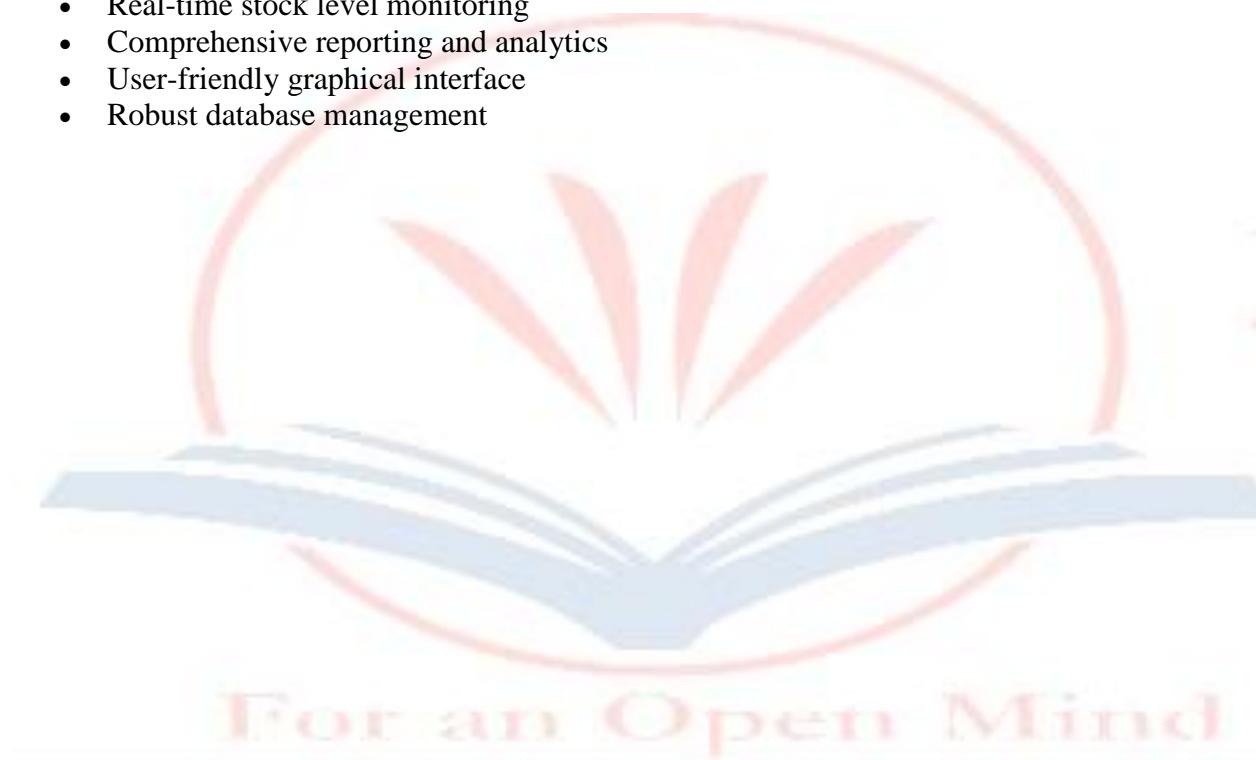
**LECTURER:MR SAMUEL MUGISHA**

## **1. 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



## **1.0 INTRODUCTION**

### **2.1 Background**

In the modern retail landscape, efficient inventory and order management are crucial for business success. Small to medium-sized businesses often struggle with affordable, easy-to-use solutions that meet their specific needs. Girly Collections addresses this gap by providing a tailored solution for fashion accessory retailers.

### **2.2 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

### **2.3 Proposed Solution**

A desktop application that provides:

- Automated inventory tracking
- Seamless order management
- Real-time stock updates
- Comprehensive business analytics
- Dual interfaces for customers and administrators

## **3. SYSTEM OVERVIEW**

### **3.1 System Type**

Desktop Application (Standalone)

### **3.2 Primary Users**

- **Customers:** Browse products, place orders
- **Administrators:** Manage inventory, process orders, view reports

### **3.3 Operating Environment**

- Platform: Windows/Linux/macOS
- Database: SQLite (local)
- GUI Framework: Tkinter (Python)



## 4. PROJECT OBJECTIVES

### 4.1 Primary Objectives

#### 1. Inventory Management

- Add, edit, and delete products
- Track stock levels automatically
- Set low stock alerts
- Categorize products efficiently

#### 2. Order Processing

- Accept customer orders
- Update order status in real-time
- Track order history
- Generate order reports

#### 3. Stock Control

- Monitor inventory levels
- Alert on low stock items
- Support multiple stock operations
- Maintain stock history

#### 4. Business Analytics

- Generate sales reports
- Track customer behavior
- Monitor business performance
- Identify trends and patterns

### 4.2 Secondary Objectives

- Provide intuitive user interface
- Ensure data integrity and security
- Enable easy backup and recovery
- Support future scalability

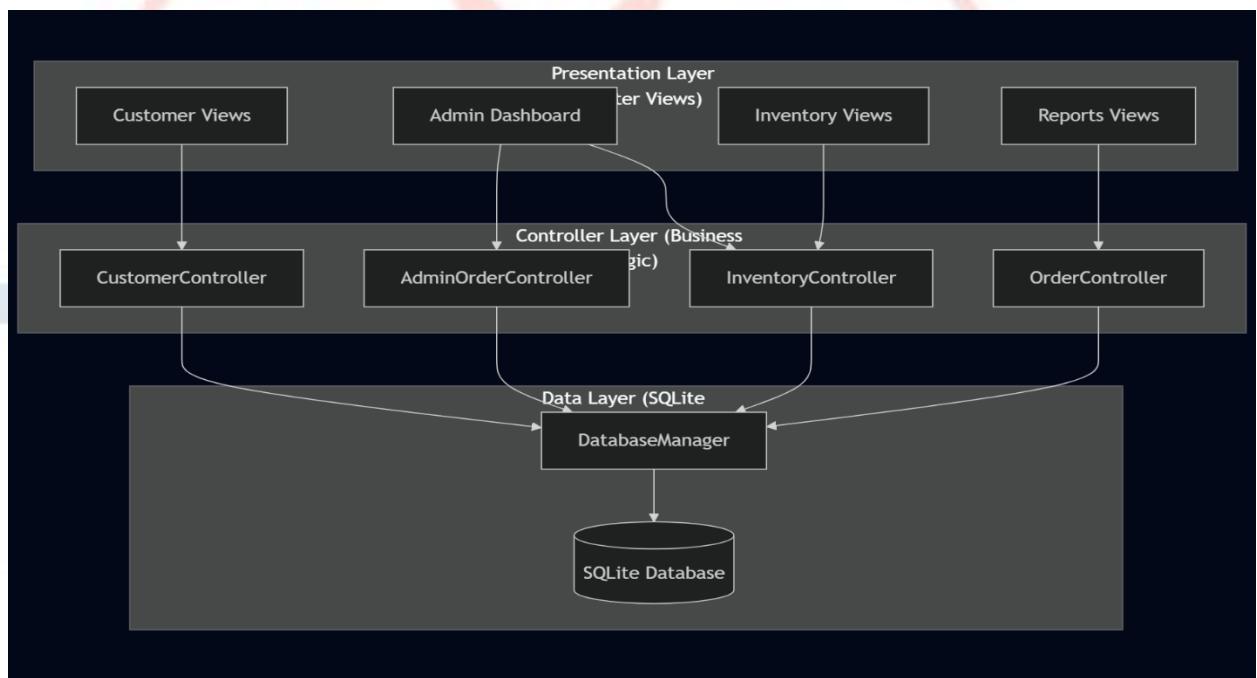
## 5. SYSTEM ARCHITECTURE

### 5.1 Architecture Pattern

#### 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
- **Controller:** Manages business logic and data flow



### 5.2 Component Breakdown

#### Controllers (Business Logic Layer)

- Handle all business operations
- Manage database connections
- Validate data

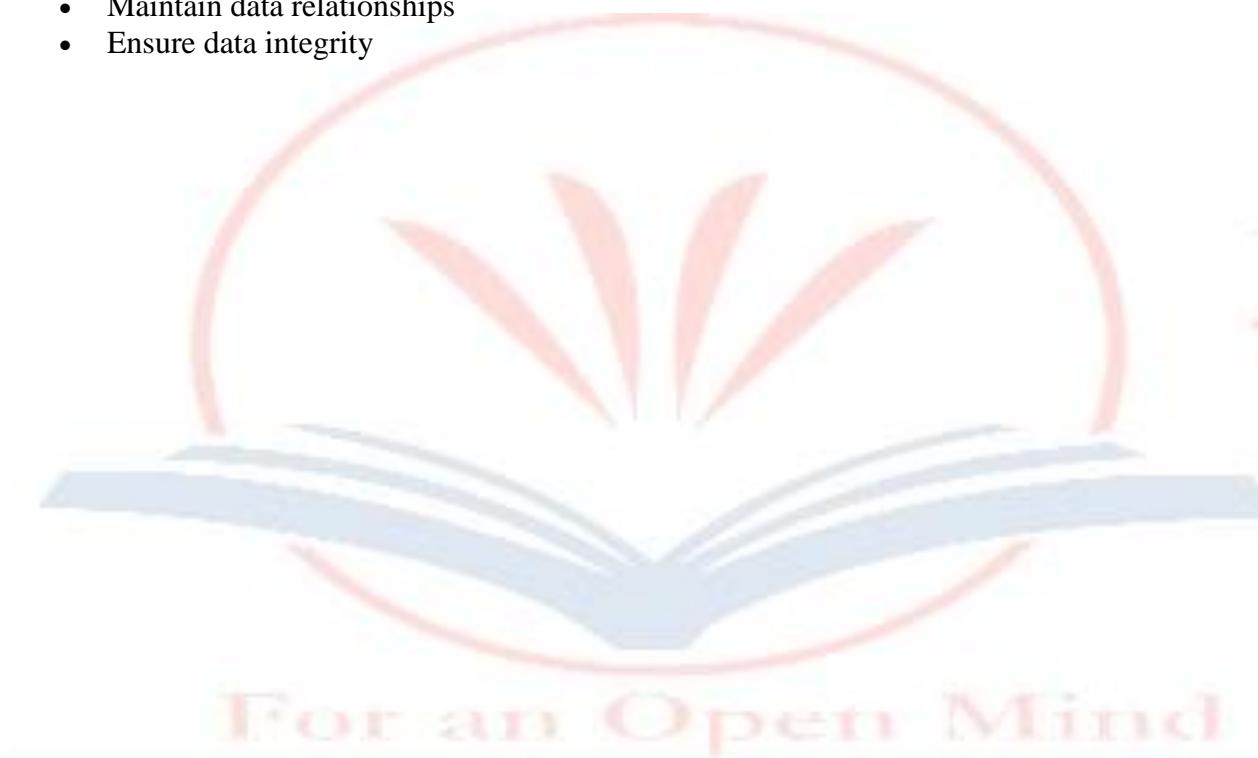
- Execute CRUD operations

### **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



## 6. TECHNOLOGIES USED

### 6.1 Programming Language

#### Python 3.7+

- Reason: Cross-platform, extensive libraries, rapid development

### 6.2 GUI Framework

#### Tkinter

- Built-in Python library
- Cross-platform compatibility
- Lightweight and fast
- Native look and feel

### 6.3 Database

#### SQLite3

- Serverless, zero-configuration
- Lightweight and fast
- Single-file database
- Built-in Python support

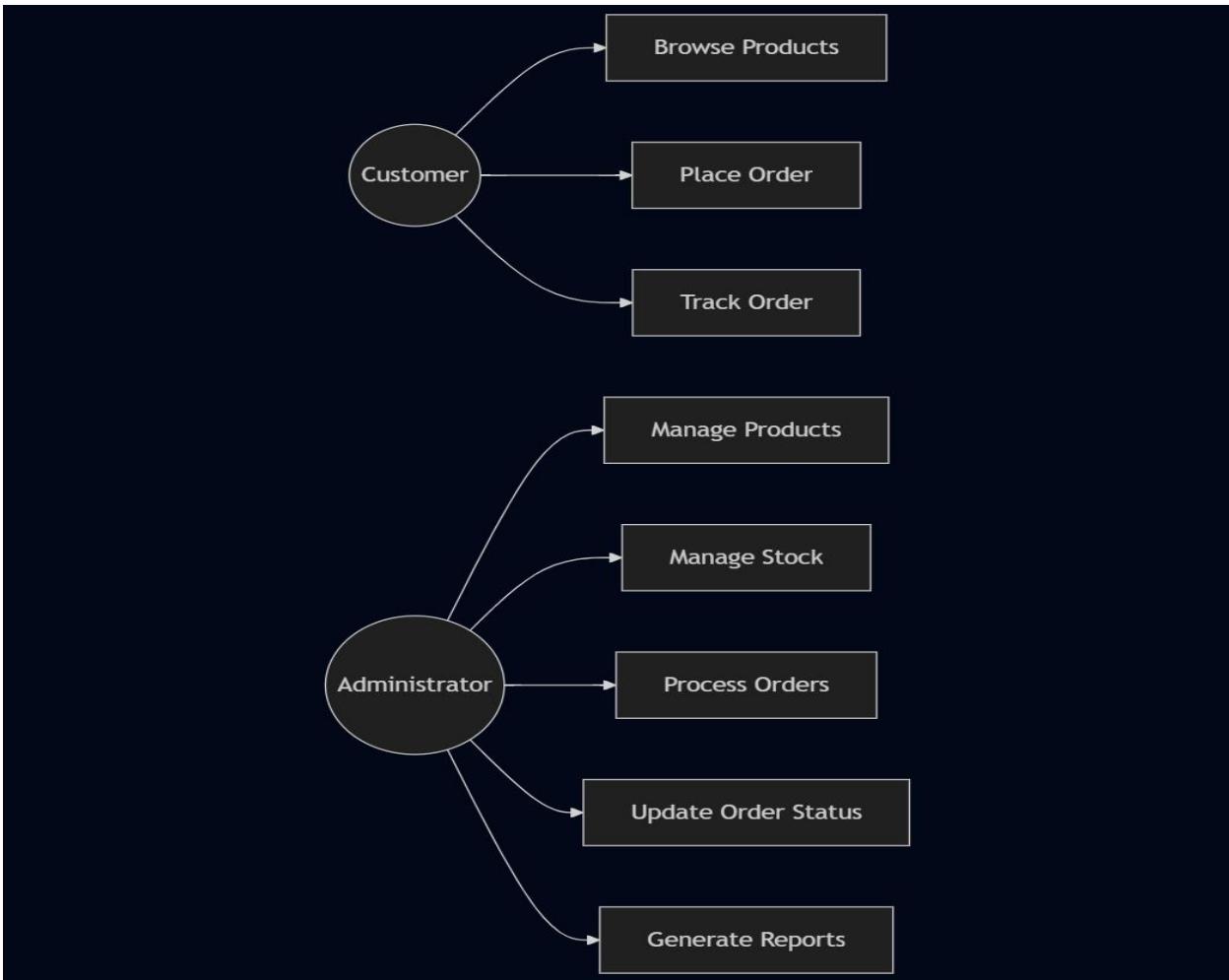
### 6.4 Additional Libraries

- **Pillow (PIL)**: Image processing
- **datetime**: Date/time operations
- **os/sys**: File system operations

### 6.5 Development Tools

- **IDE**: Visual Studio Code
- **Version Control**: Git
- **Database Viewer**: DB Browser for SQLite

## 7. SYSTEM FEATURES



### 7.1 Customer Module

#### *Product Browsing*

- View all available products
- Search products by name/category
- Filter by price range
- View product details and images

#### *Order Placement*

- Add products to cart
- Specify quantities
- Enter customer information
- Submit orders with confirmation

### *Order Tracking*

- View order history
- Check order status
- Track recent orders

## **7.2 Admin Module**

### *7.2.1 Inventory Management*

#### **Features:**

- Add new products with complete details
- Edit existing product information
- Delete products from inventory
- Upload product images
- Manage product categories
- Set SKU codes
- Configure low stock alerts

#### **Operations:**

- Create: Add new inventory items
- Read: View all products with search/filter
- Update: Modify product details
- Delete: Remove products with confirmation

### *7.2.2 Order Management*

#### **Features:**

- View all customer orders
- Filter orders by status
- Update order status workflow:
  - Pending → Processing → Shipped → Delivered
- Search orders by customer/ID
- View detailed order information
- Print order details
- Contact customers

#### **Dashboard Statistics:**

- Total orders
- Pending orders count
- Today's orders
- Total revenue

### 7.2.3 Stock Updates

#### Features:

- Quick stock adjustments
- Multiple operation modes:
  - Add stock (restocking)
  - Remove stock (sales/damage)
  - Set exact amount
- Visual stock indicators:
  - ✓ Good stock (green)
  - ⚡ Warning level (yellow)
  - ⚠ Low stock (red)
  - ✘ Out of stock (red)
- Activity logging
- Quick quantity buttons (5, 10, 20, 50, 100)
- Low stock filtering

#### Stock Overview:

- Total products
- Low stock items count
- Total stock units
- Real-time status updates

### 7.2.4 Reports & Analytics

#### Sales Reports:

- Time period filtering (Today, Week, Month, Year, All)
- Total orders and revenue
- Average order value
- Highest order amount
- Orders by status breakdown
- Sales by category

#### Inventory Reports:

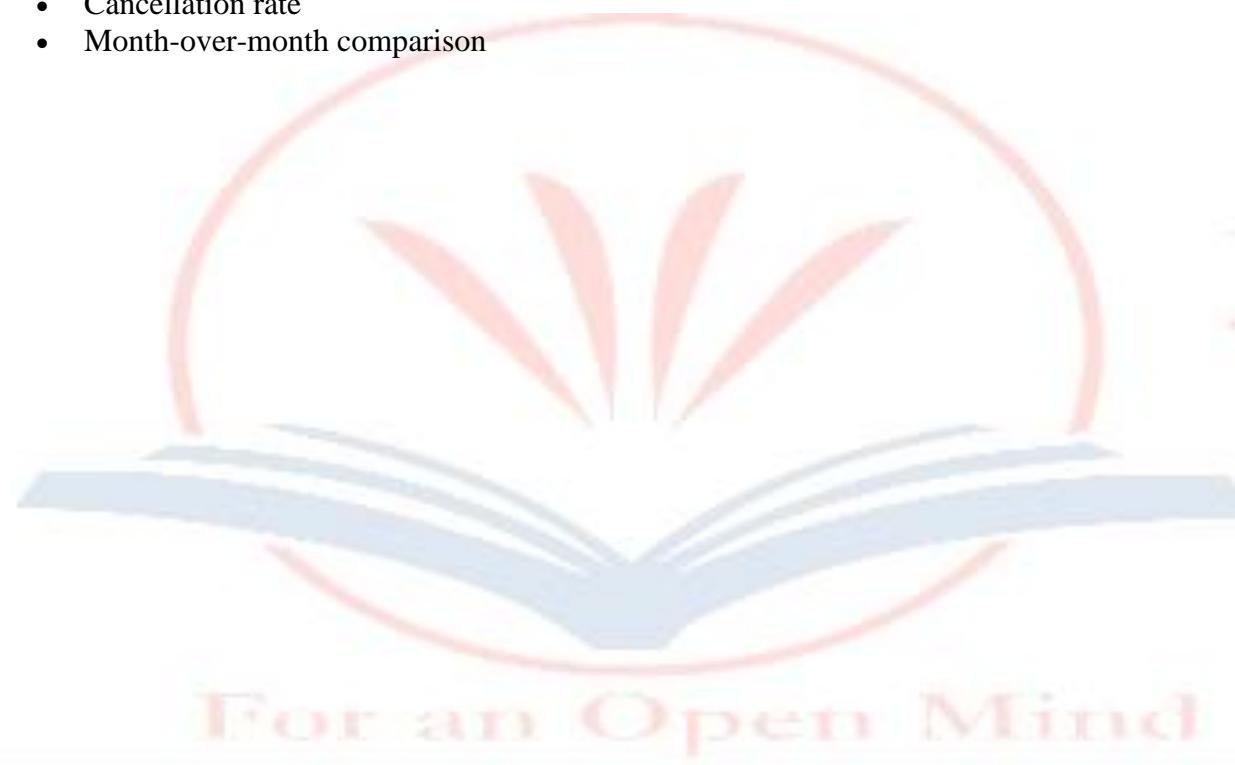
- Total products
- Total stock units
- Inventory value
- Low stock items
- Product-wise stock details
- Stock status indicators

#### Customer Analytics:

- Total customers
- Repeat customers
- Customer retention rate
- Average orders per customer
- Top customers ranking
- Purchase history

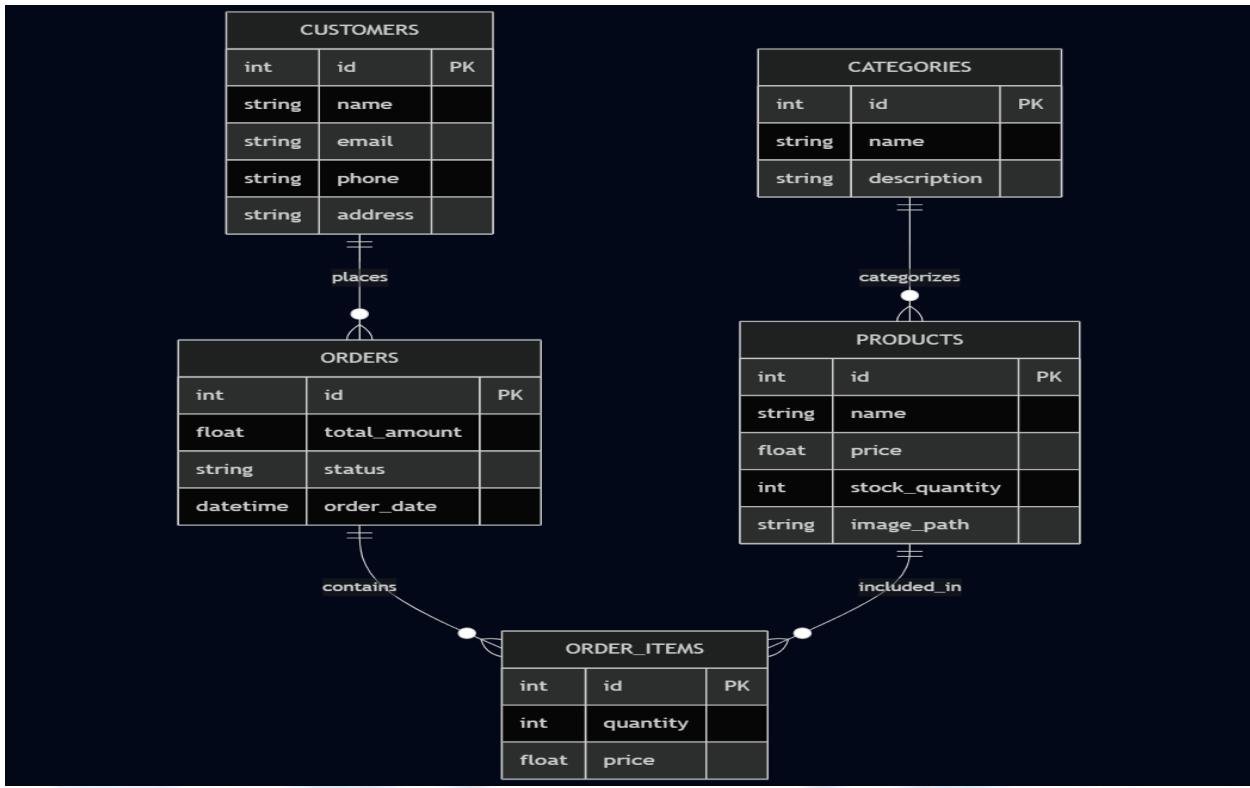
### **Performance Metrics:**

- Monthly performance trends
- Revenue growth tracking
- Order fulfillment rate
- Cancellation rate
- Month-over-month comparison



## 8. DATABASE DESIGN

### 8.1 Entity Relationship Diagram



);

For an Open Mind

## 9. USER INTERFACE DESIGN

### 9.1 Design Principles

1. **Simplicity:** Clean, uncluttered interfaces
2. **Consistency:** Uniform design across all modules
3. **Feedback:** Immediate response to user actions
4. **Accessibility:** Easy-to-read fonts and colors
5. **Efficiency:** Minimal clicks to complete tasks

### 9.2 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)

### 9.3 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

## 10. IMPLEMENTATION DETAILS

### 10.1 Code Organization

```
girlush_collections/
    └── controllers/
        ├── inventory_controller.py
        ├── admin_order_controller.py
        ├── order_controller.py
        └── customer_controller.py
    └── views/
        ├── admin_inventory_view.py
        ├── admin_orders_view.py
        ├── stock_update_view.py
        └── reports_view.py
    └── database/
        └── girlush_collections.db
    └── assets/
        ├── images/
        └── icons/
    └── admin_dashboard.py
└── main.py
```

### 10.2 Key Classes

#### Inventory Controller

- Manages all inventory operations
- Handles CRUD operations for products
- Stock level management
- Category management

#### AdminOrderController

- Manages customer orders
- Order status updates
- Order statistics

- Search and filtering

### **AdminInventoryView**

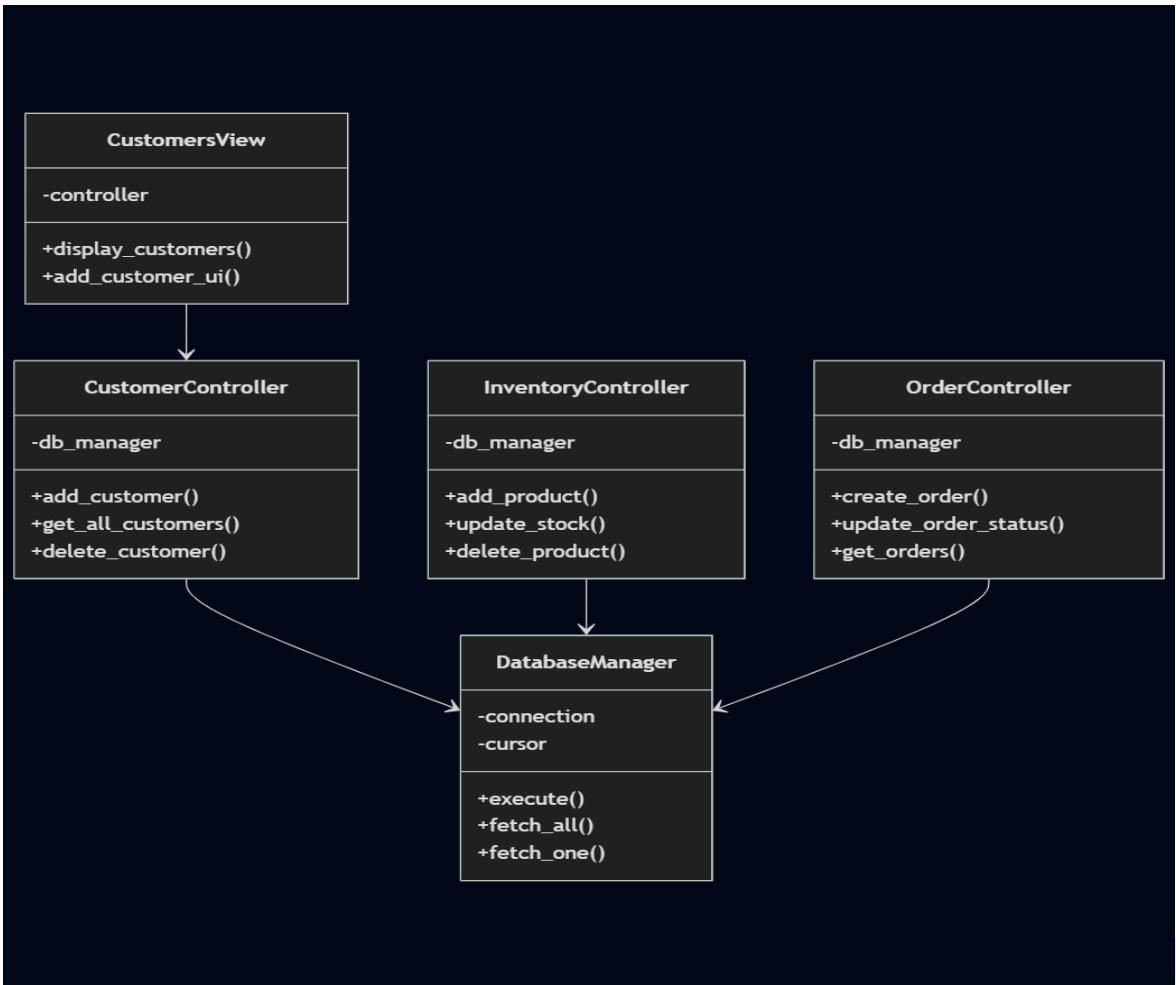
- Product form interface
- Product listing display
- Search and filter UI
- Image upload handling

### **Reports View**

- Multi-tab analytics interface
- Data visualization
- Report generation
- Export functionality

### **10.3 Design Patterns Used**

1. **MVC Pattern:** Separation of concerns
2. **Singleton:** Database connections
3. **Observer:** UI updates on data changes
4. **Factory:** View creation
5. **Strategy:** Different stock operations



## 11. TESTING & VALIDATION

### 11.1 Testing Types Conducted

#### Unit Testing

- Controller method validation
- Database operation testing
- Data validation functions

#### Integration Testing

- Controller-View interaction
- Database connectivity
- Multi-module workflows

#### User Interface Testing

- Navigation testing
- Form validation
- Button functionality
- Error message display

#### System Testing

- End-to-end workflows
- Performance testing
- Data integrity checks

### 11.2 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

**Reports:** ✓ Generate sales reports ✓ Filter by time period ✓ View customer analytics ✓ Calculate metrics correctly

### 11.3 Validation Results

- All critical functions tested: ✓ PASSED
- Data integrity maintained: ✓ PASSED
- User interface responsive: ✓ PASSED
- Error handling effective: ✓ PASSED



## **12. CHALLENGES & SOLUTIONS**

### **12.1 Technical Challenges**

#### **Challenge 1: Database Connection Management**

- Problem: Multiple database connections causing locks
- Solution: Implemented connection pooling and proper closing

#### **Challenge 2: Real-time UI Updates**

- Problem: UI not reflecting database changes immediately
- Solution: Added refresh mechanisms after data modifications

#### **Challenge 3: Large Dataset Performance**

- Problem: Slow loading with many products
- Solution: Implemented pagination and lazy loading

#### **Challenge 4: Image Storage**

- Problem: Large images increasing database size
- Solution: Store image paths instead of binary data

### **12.2 Design Challenges**

#### **Challenge 1: User Experience**

- Problem: Too many clicks to complete tasks
- Solution: Added quick action buttons and shortcuts

#### **Challenge 2: Visual Feedback**

- Problem: Users unsure if actions completed
- Solution: Added confirmation messages and status indicators

#### **Challenge 3: Data Validation**

- Problem: Invalid data entry causing errors
- Solution: Implemented comprehensive validation with clear error messages

## **13. FUTURE ENHANCEMENTS**

### **13.1 Planned Features**

#### **Phase 1: Enhanced Security**

- User authentication system
- Role-based access control
- Password encryption
- Activity audit logs

#### **Phase 2: Advanced Features**

- Barcode scanning
- Receipt printing
- Email notifications
- SMS alerts for low stock

#### **Phase 3: Cloud Integration**

- Cloud database backup
- Multi-device synchronization
- Remote access capability
- Mobile app integration

#### **Phase 4: Business Intelligence**

- Advanced analytics dashboard
- Predictive inventory management
- Customer behavior analysis
- Sales forecasting

### **13.2 Scalability Considerations**

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

## 14. CONCLUSION

### 14.1 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

1. **User-Friendly Interface**: Intuitive design requiring minimal training
2. **Robust Architecture**: Scalable OOP design following best practices
3. **Data Integrity**: Reliable database operations with validation
4. **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.

**Video demo= [girlushdemo.mp4](#)**

**GitHub link= <https://github.com/KwagalaCathy/girlush-collection>**

## **15. REFERENCES**

### **Technical Documentation**

1. Python Official Documentation - <https://docs.python.org/3/>
2. Tkinter Documentation - <https://docs.python.org/3/library/tkinter.html>
3. SQLite Documentation - <https://www.sqlite.org/docs.html>

### **Design Resources**

4. Material Design Guidelines - <https://material.io/design>
5. UI/UX Best Practices - Nielsen Norman Group

### **Development Resources**

6. Real Python Tutorials - <https://realpython.com>
7. Stack Overflow Community
8. Python Package Index (PyPI)

### **Books & Articles**

9. "Design Patterns: Elements of Reusable Object-Oriented Software" - Gang of Four
10. "Clean Code: A Handbook of Agile Software Craftsmanship" - Robert C. Martin

## APPENDICES

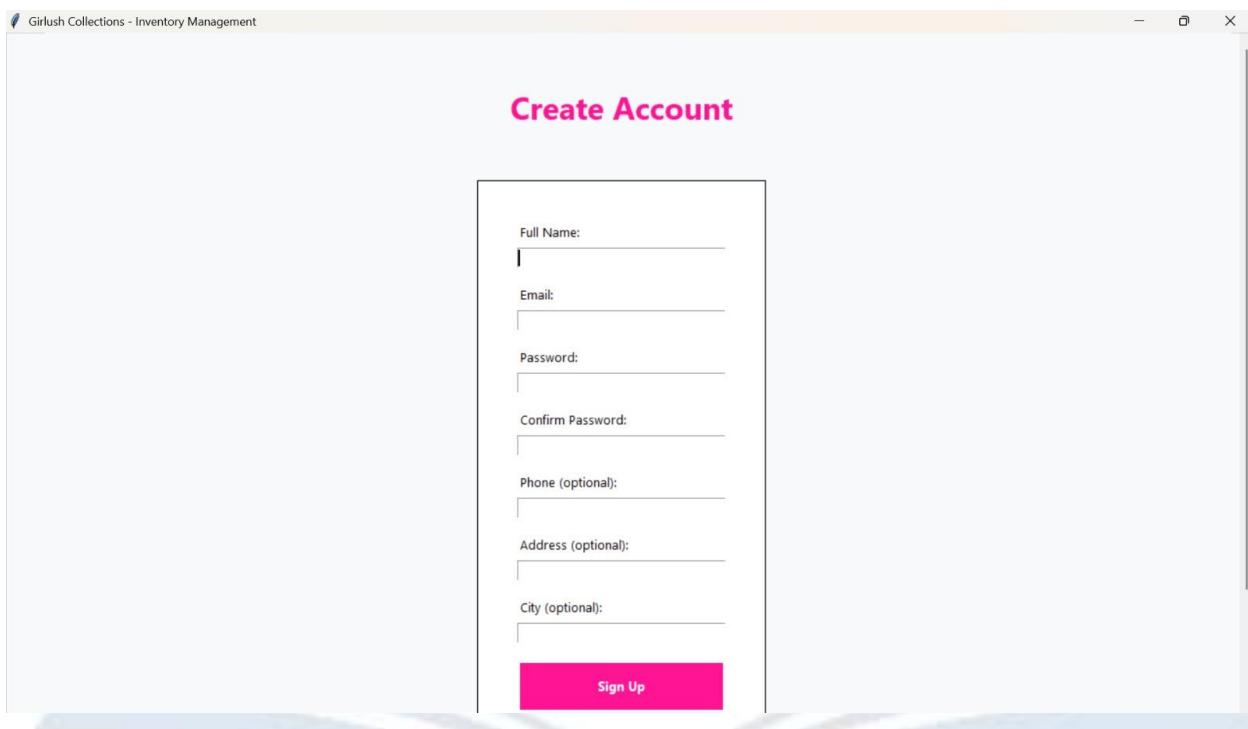
### ADMIN DASHBOARD VIEW

The screenshot shows the Admin Dashboard of the Girlush Collections - Inventory Management system. The interface has a pink header bar with the title "Girlush Collections" and a "Welcome, Admin (Admin)" message. On the left, a dark sidebar lists navigation options: Dashboard, Products, Customers, Orders, Suppliers, Sales Reports, Settings, and Logout. The main content area is titled "Admin Dashboard". It features a "Quick Actions" section with four buttons: "Add Product" (plus sign), "View Orders" (cube), "Sales Report" (bar chart), and "Manage Users" (two people). Below this is an "Overview Statistics" section with four boxes showing: "Total Products" (27), "Total Customers" (3), "Total Orders" (3), and "Total Sales" (UGX 0). At the bottom, there's a "Recent Orders" section listing two pending orders: "Order #3 - mwebaza cynthia" (UGX 45,000) and "Order #2 - buteramarcel" (UGX 90,000).

### LOGIN SCREEN

The screenshot shows the login screen of the Girlush Collections - Inventory Management system. The interface has a light blue header bar with the title "Girlush Collections" and a "Welcome, Admin (Admin)" message. On the left, a vertical sidebar contains icons for Home, Search, User, Settings, and Logout. The main content area is titled "Girlesh Collections" and "Inventory Management System". It features a login form with fields for "Email" and "Password", and a "Login" button. Below the form, a link says "Don't have an account? [Sign Up](#)".

## SIGNUP PAGE



The screenshot shows a web browser window titled "Girlish Collections - Inventory Management". The main content is a "Create Account" form. The form fields are as follows:

- Full Name:
- Email:
- Password:
- Confirm Password:
- Phone (optional):
- Address (optional):
- City (optional):

A pink "Sign Up" button is located at the bottom of the form.

## CUSTOMER DASHBOARD VIEW

For an Open Mind

Girlish Collections - Inventory Management

Welcome, mwebaza cynthia Cart (0)

**Dashboard**

**Quick Actions**

- My Orders
- My Cart
- Profile Settings

**Quick Stats**

- Total Orders: 1
- Cart Items: 0
- Total Spent: UGX 0

**Recent Orders**

Order #3	PENDING	UGX 45,000
----------	---------	------------

## DATABASE

Internal storage DB Browser for SQLite - C:\Users\USER\Desktop\cathie\dist\girlish\_inventory.db

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Create Table Create Index Modify Table Delete Table Print Refresh

Identity Select an identity to connect DBHub.io Local Current Database

Name Type Schema

Tables (10)

- cart
- customers
- inventory\_transactions
- order\_items
- orders
- products
- sales
- sqlite\_sequence
- suppliers
- users

Indices (0)

Views (0)

Triggers (0)

CREATE TABLE cart ( cart\_id )  
CREATE TABLE customers ( cu...  
CREATE TABLE inventory\_transa...  
CREATE TABLE order\_items ( o...  
CREATE TABLE orders ( order\_...  
CREATE TABLE products ( pro...  
CREATE TABLE sales ( sale\_id )  
CREATE TABLE sqlite\_sequenc...  
CREATE TABLE suppliers ( sup...  
CREATE TABLE users ( user\_id )

## CODE SNIPPET

### MAIN.PY



A screenshot of a code editor window titled "cathie". The left sidebar shows a file tree with a folder named "CATHIE" containing subfolders like "assets", "build", "components", "controllers", "database", "dist", "logs", "utils", "views", and files like ".gitignore", "build\_exe.spec", "config.py", "girly\_inventory.db", "main.py", "populate\_database.py", "README.md", "requirements.txt", "test\_database.py", and "UPDATES.md". The main editor area displays the content of "main.py". The code imports various modules such as os, config, database.database\_manager, controllers.auth\_controller, controllers.product\_controller, controllers.customer\_controller, controllers.order\_controller, controllers.cart\_controller, views.login\_view, views.signup\_view, views.admin\_dashboard\_view, and views.customer\_dashboard\_view. It defines a class Application that initializes the window, sets its title and geometry, centers it, initializes controllers, and sets the current user to None.

```
5     import tkinter as tk
6     from tkinter import messagebox
7     import os
8     import config
9     from database.database_manager import DatabaseManager
10    from controllers.auth_controller import AuthController
11    from controllers.product_controller import ProductController
12    from controllers.customer_controller import CustomerController
13    from controllers.order_controller import OrderController
14    from controllers.cart_controller import CartController
15    from views.login_view import LoginView
16    from views.signup_view import SignupView
17    from views.admin_dashboard_view import AdminDashboardView
18    from views.customer_dashboard_view import CustomerDashboardView
19
20    class Application(tk.Tk):
21        def __init__(self):
22            super().__init__()
23
24            # Window configuration
25            self.title(config.WINDOW_TITLE)
26            self.geometry(f"{config.WINDOW_WIDTH}x{config.WINDOW_HEIGHT}")
27            self.minsize(config.MIN_WINDOW_WIDTH, config.MIN_WINDOW_HEIGHT)
28
29            # Center window
30            self.center_window()
31
32            # Initialize database and controllers
33            self.init_controllers()
34
35            # Current user
36            self.current_user = None
```