**Jewelry Shop Management System: Project Report**

**1. Introduction**

The Jewelry Shop Management System is designed to streamline the management of jewelry businesses by providing tools to handle inventory, customers, orders, and sales efficiently. Built with .NET Core and Entity Framework Core (EF Core), the system ensures data consistency and offers robust features for day-to-day operations.

**2. Objective**

To develop a user-friendly management system for jewelry businesses that automates shop operations, manages inventory, tracks orders, and ensures seamless data handling.

**3. Features**

1. **Shop Management**
   * Add, update, and delete shops.
   * Manage shop details such as location and contact information.
2. **Product and Category Management**
   * Organize products into categories.
   * Maintain inventory levels and track stock.
   * Manage product pricing and details.
3. **Customer and User Management**
   * Register customers and maintain their order history.
   * Manage shop staff and admin accounts.
4. **Order Management**
   * Create and process orders.
   * Manage order details and calculate total amounts.
   * Track order status and payment.
5. **Data Consistency**
   * Relationships are enforced using EF Core to ensure integrity across shops, products, orders, and users.

**4. Technology Stack**

1. **Framework:** .NET Core 7.0
2. **ORM:** Entity Framework Core
3. **Database:** SQL Server
4. **Programming Language:** C#
5. **Tools:** Visual Studio, NuGet
6. **Design:** Model-View-Controller (MVC) architecture

**5. Database Schema**

The database includes the following tables:

* **Shops:**
  + ShopId, Name, Location, ContactInfo
* **Categories:**
  + CategoryId, Name, ShopId (FK)
* **Products:**
  + ProductId, Name, Price, Stock, CategoryId (FK), ShopId (FK)
* **Users:**
  + UserId, Name, Email, Password, ShopId (FK)
* **Orders:**
  + OrderId, OrderDate, TotalAmount, UserId (FK), ShopId (FK)
* **OrderDetails:**
  + OrderDetailId, OrderId (FK), ProductId (FK), Quantity, UnitPrice

**6. Relationships and Constraints**

1. **Cascade Delete:**
   * OrderDetails -> Orders.
   * Categories -> Products.
2. **Restrict Delete:**
   * Products -> Shops.
   * Orders -> Shops.
3. **Precision Constraints:**
   * Decimal fields for TotalAmount, UnitPrice, and Price are set to (18,2).

**7. Key Challenges and Solutions**

1. **Challenge:** Multiple cascade paths in OrderDetails.
   * **Solution:** Applied DeleteBehavior.Restrict on ProductId foreign key to prevent cycle issues.
2. **Challenge:** Maintaining data consistency.
   * **Solution:** EF Core configurations for relationships and constraints.
3. **Challenge:** Handling complex relationships.
   * **Solution:** Defined relationships explicitly in the OnModelCreating method.

**8. Implementation Details**

1. **Code Structure:**
   * Organized into Models, Data (DbContext), Controllers, and Views.
2. **Entity Framework Migrations:**
   * Used Add-Migration and Update-Database to manage schema changes.
3. **Validation:**
   * Data annotations and Fluent API for validation rules.

**9. Testing and Debugging**

1. **Unit Testing:**
   * Tested individual features such as adding shops, products, and orders.
2. **Integration Testing:**
   * Validated relationships and data consistency across entities.
3. **Error Handling:**
   * Addressed cascade path issues by fine-tuning foreign key constraints.

**10. Conclusion**

The Jewelry Shop Management System successfully provides an efficient solution for managing jewelry shop operations. With its modular structure, scalability, and robust data handling, it is tailored to meet the dynamic needs of jewelry businesses.

**11. Future Enhancements**

1. Add real-time inventory tracking.
2. Implement user authentication and role-based access control.
3. Introduce analytics dashboards for sales and inventory.
4. Enable multi-shop support with centralized management.
5. Integrate online order processing and payment gateways.