Project Proposal: PC Parts E-Commerce System

---

1. Project Overview

This project aims to build an e-commerce platform for PC components, enabling customers to view, search, and order various products online, while allowing store admins to manage inventory, orders, and product attributes across multiple branches. The system will be designed using **.NET Core** with **PostgreSQL** as the database, following MVC architecture for clean separation of concerns.

2. Core Features

User Roles & Permissions

1. **Admin**:
    - Full access to the system, including user management, global inventory, and sales reports.
    - Can edit any user information and view all orders and product data.
2. **Branch Admin (BAdmin)**:
    - Can access their specific branch's inventory and manage staff members (view and edit staff info).
    - Can request products from other branches and view orders for their branch.
    - Manages warranty claims for their branch.
3. **Salesman**:
    - Has access only to their branch's sales data and inventory.
    - Sales performance is tracked for generating bonus reports (based on sales or references).
4. **Customer**:
    - Can view, search products by categories/brands/attributes, and place orders.
    - Can track the status of warranty claims and see purchase history.
    - Can use discount coupons and pay via mobile banking (e.g., bKash, Visa).

Product Management:

- Products have detailed attributes (e.g., memory size, CPU cores, refresh rates).
- Categories, brands, and vendors are managed to organize products.
- Support for adding product attributes like CPU core count, GPU memory size, and monitor screen size.
- Product data includes warranty periods, prices, descriptions, etc.

Order Management:

- **Order Status**: Includes status updates for processing, shipping, delivered, etc., which can be manually updated by Branch Admins.
- **Payment Status**: Includes tracking for payments made via various methods (mobile banking, Visa).
- **Order Notes**: Notes for each order, visible to Admin, BAdmin, and Customer.

Warranty Management:

- Warranty claims are handled by Admins and BAdmins, and the status is tracked.

- A ticket is generated for each warranty claim, and progress can be followed by the customer, Admin, and BAdmin.

Discount System:

- In-store discounts are applied manually by the sales staff at checkout.
- Online discount coupons are applied during the checkout process.

Customer Review/Rating:

- Customers can leave reviews and ratings for products they have purchased.
- Reviews include a text description and a numerical rating.
- These reviews are visible to other customers and help guide purchasing decisions.

Online Payment:

- **Payment Info**: Payment details (amount, method, status) are stored in the database for each order.
- **Payment Status**: The status of payment (paid, pending, failed) is updated for each order.
- While no third-party payment integration is required, the system will track and manage payments for online orders.

3. Database Design

The database is designed to manage key entities related to products, orders, and user roles. The final schema includes relationships between `Products`, `Categories`, `Brands`, `Vendors`, `Orders`, `OrderDetails`, `ProductAttributes`, `Reviews`, and `Payments`.

- **Key Tables**:
  - `Users`, `Roles`, `Products`, `Categories`, `Brands`, `Vendors`, `Orders`, `OrderDetails`, `ProductAttributes`, `Reviews`, `Payments`
  - Foreign key relationships ensure referential integrity between tables (e.g., `ProductId` in `ProductAttributes` is linked to `Products`).
- **Product Attributes**:
  - The `ProductAttributes` table uses a **composite primary key** (`ProductId`, `AttributeName`) to ensure each product's attribute is unique. Attributes like CPU cores, memory sizes, screen size, etc., are stored in this table.
  - Attributes are linked to the `Products` table, providing flexibility to manage product specifications.
- **Customer Reviews/Rating**:
  - The `Reviews` table stores the product reviews and ratings, with each review linked to a specific product and customer.
  - Reviews include a rating (1-5 stars) and an optional review text.
- **Payments**:
  - The `Payments` table stores information related to online payments, such as payment method, status, and transaction details. This allows the system to track payment statuses and ensure smooth processing of online orders.

4. Key Business Logic

1. **Customer Orders**:
   - Customers can place orders, and the status of the order (e.g., confirmed, processing, shipped) is tracked.

    - Payment statuses are also managed (paid, pending, failed).
2. **Warranty Claims**:
    - When a product is returned for warranty service, a ticket is created and can be tracked by the customer, BAdmin, and Admin.
3. **Sales Reporting**:
    - Salesmen's performance is tracked, and reports are generated to determine the top performers (based on sales or references).
4. **Review/Rating**:
    - Reviews and ratings are tracked for products, and these are available for customers to view on the product detail page.
    - Review data is stored along with the user ID and product ID for reference.
5. **Payment Tracking**:
    - For online orders, payment details (amount, payment method, and status) are stored in the database for each order.
    - Payment status updates are tracked, ensuring accurate records for each transaction.

5. Architecture & Design Patterns

- **Model-View-Controller (MVC)**:
    - The system is designed with MVC architecture to maintain clear separation of concerns, improving maintainability and scalability.
    - Backend logic is handled by controllers, with data retrieved from repositories (following the Repository Pattern).
- **Repository Pattern**:
    - The system utilizes the repository pattern to abstract data access, keeping business logic and database queries separated for clean, testable code.
    - This ensures that queries are not embedded directly in the controllers, following best practices for code cleanliness and maintainability.
- **SOLID Principles**:
    - The code will follow SOLID principles for better object-oriented design. This will allow easy extensibility, and maintainability, and will make the codebase more robust.

6. Database Management

- **Triggers**:
    - **Automatic Timestamp Management**: Triggers will automatically set the `CreatedOn`, `UpdatedOn`, `CreatedBy`, and `UpdatedBy` fields whenever a record is inserted or updated.
    - **Soft Delete**: Soft delete functionality is handled manually in the application, with `IsDeleted` flags used to mark records as deleted instead of physically removing them.
- **Data Integrity**:
    - Foreign keys are used throughout the database schema to enforce referential integrity between related tables (e.g., `Products` to `Categories`, `Brands`, `Vendors`).

7. Frontend & User Interface

- **UI Design**:
    - A clean and modern UI is developed using **Bootstrap** and **jQuery** for dynamic, client-side actions like modals, live updates, and form validation.
    - The application will be mobile-responsive, ensuring a good user experience across devices (smartphones, tablets, desktops).
- **Frontend Interaction**:

    - Users interact with the system via views built with **ASP.NET MVC Views**. These views will leverage **Bootstrap** for layout and **jQuery** for frontend functionality.

8. Security & Authentication

- **ASP.NET Core Identity**:
    - User authentication and authorization are handled using **ASP.NET Core Identity**, providing built-in support for secure user management (e.g., registration, login, password recovery).
    - Users will have roles (e.g., Admin, BAdmin, Salesman, Customer) that define their permissions and access control.
- **Session Management**:
    - PostgreSQL sessions will be managed per user connection, ensuring session consistency and performance.

9. Future Considerations

1. **Email Notifications**: Email functionality can be added for order confirmations, shipment tracking, and warranty claims.
2. **Analytics & Reporting**: Advanced sales and inventory analytics can be integrated into the system for better decision-making.

10. Conclusion

This e-commerce platform for PC components provides a comprehensive solution for managing products, orders, reviews, payments, and warranty claims, while maintaining clear role-based access and business logic. The clean architecture with MVC and the repository pattern ensures the system is maintainable, scalable, and follows best practices.