Project 1: TechXpress E-commerce Platform with ASP.NET Core

Objective:

Build a scalable and maintainable e-commerce web application for selling electronics, leveraging design patterns like NTier Architecture, Repository Pattern, and Unit of Work for efficient development and separation of concerns.

Description:

TechXpress is a full-featured e-commerce platform that allows users to browse electronics (laptops, mobiles, cameras), add them to a shopping cart, and complete purchases using integrated payment gateways. The platform includes an admin panel for managing products, categories, and orders. It will follow the NTier architecture to separate concerns (UI, Business Logic, and Data Access), improving maintainability and scalability.

Design Patterns to Use:

- **NTier Architecture:** Separation of the application into Presentation (UI), Business Logic (Service Layer), and Data Access (Repository/Entity Framework) layers.
- **Repository Pattern:** A layer that abstracts database interactions, providing a clean API for querying and persisting data.
- **Unit of Work Pattern:** Ensures that multiple related changes are grouped together in a single transaction, reducing the risk of data inconsistencies.
- **Dependency Injection:** Inject services (repositories, business logic) into controllers to promote testability and loose coupling.

Technologies to Use:

ASP.NET Core MVC, Entity Framework Core, ASP.NET Identity (for authentication), Stripe (for payment integration), JQuery, DataTables, Toaster JS, Microsoft Azure (for deployment).

Week 1: Initial Setup and Product Listings

- **NTier Architecture Setup:**
    - **Presentation Layer (TechXpress.Web):** The user-facing ASP.NET MVC application where views, controllers, and client-side logic (JQuery, DataTables) are implemented.
    - **Business Logic Layer (TechXpress.Services):** Handles the business rules, operations like adding/removing products, calculating totals in the cart, and managing orders.
    - **Data Access Layer (TechXpress.Data):** This layer uses the Repository and Unit of Work patterns to interact with the database, ensuring clean separation from business logic.
- **Database design:**

Use **Entity Framework Core** to define the schema for tables like `Products`, `Categories`, `Orders`, `OrderDetails`, and `Users`. The `DbContext` will serve as the Unit of Work, and repositories will abstract CRUD operations.

- **Repository Pattern Implementation:**

Create generic repositories for common data operations (CRUD) and specific repositories for `ProductRepository` and `CategoryRepository`.

- **User authentication setup:**

Set up ASP.NET Identity for user registration, login, and role management (customer and admin).

Deliverables:

- Full NTier Architecture setup with proper separation of Presentation, Business Logic, and Data Access.
- Basic product listing page integrated with search and filters.
- Database schema designed using Entity Framework Core, with repositories and the Unit of Work pattern implemented.
- User authentication system integrated using ASP.NET Identity.

Week 2: Shopping Cart, Role-Based Access Control, and Admin Panel

- **Shopping cart functionality:**

Implement the shopping cart using **Session State** to store products added by users, allowing customers to review items before checkout. Utilize the **Business Logic Layer** for calculating cart totals and handling promotions or discounts.

- **Role-based access control (RBAC):**

Define roles (`Customer`, `Admin`) using **ASP.NET Identity**. Restrict access to the Admin Panel and product management features based on roles.

- **Admin dashboard:**

Create an admin interface using DataTables for managing products, categories, and orders. Use the **Repository Pattern** to abstract data access and ensure all product updates are handled through the **Unit of Work** pattern.

Deliverables:

- Fully functional shopping cart with session management.
- Role-based access control with permissions for admin and customers.
- Admin dashboard for managing product listings and orders, powered by repositories and unit of work.
- Thorough testing of shopping cart and role management.

Week 3: Order Placement, Stripe Payment Integration, and User Profiles

- **Order placement:**

Implement a checkout flow where customers can place orders. Use the **Unit of Work** to ensure the order transaction (saving the order and reducing stock) is atomic.

- **Stripe Payment Integration:**

Integrate Stripe for handling payments. Use Stripe's API to securely process payments during checkout and store transaction details for reference.

- **User Profiles and Order History:**

Create customer profile pages where users can view personal details, track their order history, and manage account settings. Ensure that profile updates and order views use the **Repository Pattern** for clean data access.

Deliverables:

- Order placement with integrated Stripe payments.
- Customer profile page with order history and account management features.
- Tested and functional order placement with Stripe payment gateway.

Week 4: Final Testing, UI Enhancements, and Deployment

- **UI enhancements:**

Refine the overall look and feel of the site using **Bootstrap** and **custom CSS** to create a modern, responsive design. Add **JQuery** for dynamic elements (e.g., live search, interactive product listings).

- **DataTables Integration for Admin Panel:**

Use **DataTables** for product management in the admin dashboard, allowing features like sorting, filtering, and pagination to improve usability.

- **Final testing and deployment:**

Perform comprehensive testing on the entire system, covering functionality like product browsing, cart management, order placement, and admin management. Deploy the application to **Microsoft Azure** or **Hostinger** for live use.

Deliverables:

- Fully responsive, polished UI for both front-end and admin panel.
- DataTables integrated into the admin dashboard for efficient product management.
- Live deployment of the TechXpress platform on Microsoft Azure/Hostinger.
- Completed project documentation, including architecture explanation (NTier, Repository, Unit of Work), setup, and user manual.