**Inventory and Sales Management System**

**🔍 Project Overview: Inventory & Sales Tracker System**

This project is a **web-based inventory and sales management system**, built using **ASP.NET MVC (.NET Framework)** with **Entity Framework (Code First)**. It’s designed for small-to-medium businesses to manage their product stock, process sales, and generate basic reports.

**🎯 Core Objectives**

* Track inventory in real-time
* Allow sales processing with automatic stock deduction
* Support multiple user roles (Manager, Salesperson)
* Generate sales reports and invoices
* Maintain data integrity (in 2NF schema)

**👥 User Roles**

1. **Manager** – full access
   * Manages users, categories, products
   * Views reports and low stock alerts
   * Generates invoices and sales summaries
2. **Salesperson** – limited access
   * Adds new sales
   * Can view available stock but not edit products
   * Cannot access admin reports or user management

**🔄 Workflow Overview**

**📦 Inventory Management**

* Manager adds **Categories** and **Products**
* Each product has price, cost, quantity, and low stock threshold

**🛒 Sales Workflow**

* Salesperson logs in, creates a **Sale**
* Chooses one or more products and quantities
* The system:
  + Calculates total
  + Deducts stock
  + Logs who made the sale
* Optionally, generates a printable **invoice**

**📈 Reporting**

* Manager can view:
  + Sales by date
  + Product stock levels
  + Sales by user (optional)
  + Products that are low in stock

**🧾 Data Design Summary**

* 5 main tables: Users, Categories, Products, Sales, SaleItems
* Uses foreign keys to enforce relationships
* Inventory and sales data stay normalized to avoid duplication

**📅 Expected Outcome**

A **functional, secure, role-based** web app where:

* Managers keep track of stock and business performance
* Salespersons handle day-to-day product sales
* The system ensures data accuracy and gives key business insights

**FOLDER STRUCTURE**

**InventorySalesTracker/**

│

**├── App\_Start/**

│ └── RouteConfig.cs (routing config)

**├── Controllers/**

│ ├── ProductController.cs

│ ├── CategoryController.cs

│ ├── SaleController.cs

│ ├── AccountController.cs

│ └── ReportController.cs

**├── Models/**

│ ├── Product.cs

│ ├── Category.cs

│ ├── Sale.cs

│ ├── SaleItem.cs

│ ├── User.cs

│ └── ApplicationDbContext.cs

**├── Views/**

│ ├── Product/ (Index.cshtml, Create.cshtml, etc.)

│ ├── Category/

│ ├── Sale/

│ ├── Account/

│ ├── Report/

│ └── Shared/

│ ├── \_Layout.cshtml

│ └── Error.cshtml

**├── Migrations/**

│ └── [EF migration files auto-generated]

**├── Scripts/**

│ └── jQuery, Bootstrap, SweetAlert (optional)

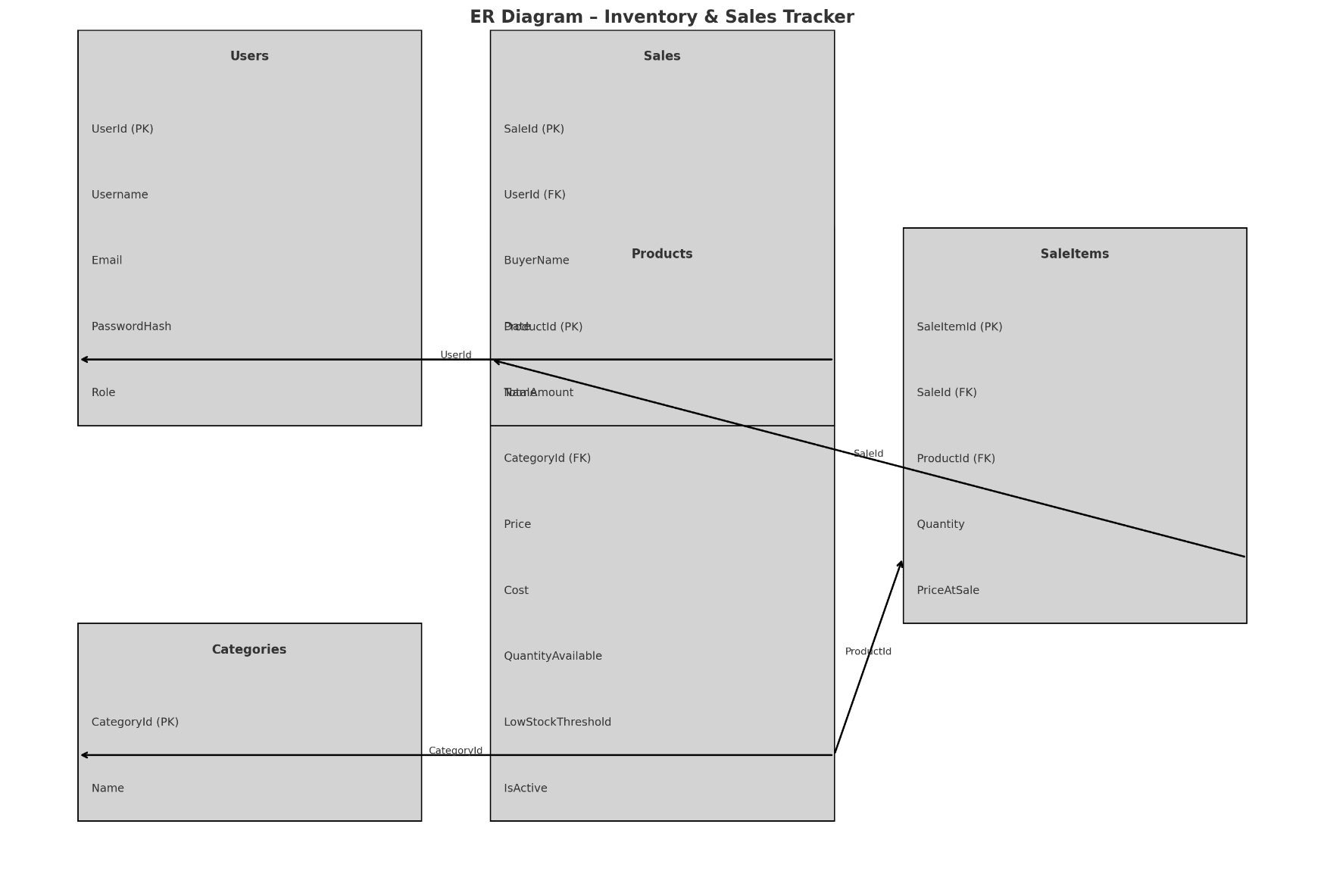
**├── Content/**

│ └── CSS, Bootstrap, Themes

├── Global.asax

├── Web.config

└── packages.config

**ER Diagram**

**Entities & Relationships**

**🧑‍💼 Users**

* **PK:** UserId
* Handles login/auth
* Role can be Manager, Salesperson

**🗃️ Categories**

* **PK:** CategoryId
* One category has many products → **1:N**

**📦 Products**

* **PK:** ProductId
* **FK:** CategoryId → Categories
* Fields: Name, Price, Cost, QuantityAvailable, LowStockThreshold, IsActive

**🧾 Sales**

* **PK:** SaleId
* **FK:** UserId → Users
* Buyer name optional
* Total calculated and stored

**📄 SaleItems (junction table)**

* **PK:** SaleItemId
* **FKs:** SaleId → Sales, ProductId → Products
* Contains: Quantity, PriceAtSale
* Represents **many-to-many** between Sales and Products via a middle table (with extra fields!)

**ASIF: (Project Lead):**

* Setup Project, EF Config, and Identity/Auth
* Handle complex logic: Sales, Invoices, Stock Update, Reporting, Exception Handling
* Layout integration (AdminLTE), Razor layout setup

**GALIB: Product & Category Management**

**Files:**

* Product.cs, Category.cs (models)
* ProductController.cs, CategoryController.cs
* Product and Category views (CRUD – Index, Create, Edit, Delete)

**Tasks:**

* Implement full CRUD using scaffolded views
* Add data validation annotations ([Required], etc.)
* Join Product to Category with FK
* Add dropdown for selecting category in Product Create/Edit view

**Bonus (after finishing everything else):**

* Add Product Search by name in Index

**MEHERAF: Sales CRUD UI + User Management**

**Files:**

* Sale.cs, SaleItem.cs, User.cs (you’ll probably scaffold base models)
* SaleController.cs
* Sale views: Create sale, View all sales

**Tasks:**

* Scaffold sale creation form (you’ll plug in the logic later)
* Show sale list with date, buyer, salesperson
* Create basic user CRUD (admin-only)

**Bonus (after finishing everything else):**

* Add Razor dropdown for selecting product(s) when creating sale (you handle multi-product logic later)

**🔜 Next Steps – Full Checklist**

**🔧 Project Essentials**

1. **Fix layout and routing**
   * Set \_Layout.cshtml properly
   * Set Home/Index as landing page
   * Add navigation to Product, Category, Sale
2. **Implement Authentication**
   * Create a simple login form (skip Identity)
   * Add a User seed or manual record
   * Use Session to store UserId, Role
3. **Restrict Controller Access by Role**
   * Use [Authorize]-style manual checks
   * Block Salespeople from accessing Category/Product Create/Delete

**💼 Product & Category Final Touches**

1. **Polish Category/Product CRUD Views**
   * Add validation messages
   * Add status (Active/Inactive) toggle for products
   * Add product search by name or filter by category (optional)

**💰 Sales System (Core Logic)**

1. **Redesign Sale/Create.cshtml**
   * Remove default scaffolded form
   * Add product dropdown + quantity field (multi-product capable)
   * Dynamically calculate total
2. **Implement Sale Logic in Controller**
   * Create new Sale + linked SaleItem rows
   * Deduct product stock quantity
   * Auto-calculate total price
3. **Add View for Sale Invoice**
   * Simple printable Razor View (Invoice style)
   * Show: Buyer, Date, Products, Quantity, Total

**📊 Reporting & Dashboard**

1. **Create a ReportController**
   * Sales by date range
   * Products low in stock
   * Optional: Sales by user
2. **Create a Dashboard for Manager**
   * Total sales today/this month
   * Products low in stock
   * Quick links

**💥 Final Polish**

1. **Add Exception Handling**
   * Global error page (500, 404)
   * Try-catch in Sale logic
2. **UI Improvements**
   * Add AdminLTE or Bootstrap layout
   * Apply layout to all views
3. **User Management (Optional)**
   * Allow Manager to add/edit users (basic form, no Identity needed)

**🟢 Suggested Order to Work On:**

Next Steps:

1. ✅ **Fix \_Layout.cshtml & basic navigation**
2. 🛡️ **Implement login + session-based auth**
3. 💰 **Work on Sale/Create.cshtml with product selection**
4. 🧮 **Implement stock deduction + sale saving logic**