Sure! Here's an idea for an **ASP.NET MVC** project with **SQL Server** that could help you showcase your skills and land a job: **Online Inventory Management System**.

**Project Overview: Online Inventory Management System**

This project will involve building a web-based application where users (typically warehouse managers or business owners) can manage product inventories, track stock levels, add or remove products, and generate reports for analysis. You'll need both the backend (C# and SQL Server) and frontend (MVC framework) to work together seamlessly.

**Key Features:**

1. **User Authentication & Authorization**:
   * Implement user registration, login, and role-based authorization (Admin, Manager, Viewer).
   * Admins can manage products, users, and reports.
   * Managers can add/remove products but cannot access user management.
   * Viewers can only view inventory and reports.
2. **Product Management**:
   * Add, edit, or delete products in the inventory.
   * Track product details: name, category, price, quantity, supplier, etc.
   * Display product inventory with filtering and searching capabilities.
3. **Stock Management**:
   * Track product stock levels.
   * Update stock when receiving new products or when items are sold.
   * Alert the user when stock reaches a threshold level (low stock warning).
4. **Supplier Management**:
   * Add and manage suppliers of products.
   * Each product can be linked to a supplier.
5. **Reports & Analytics**:
   * Generate reports to see the sales, stock levels, and product trends.
   * Include features like monthly sales report, inventory report, and supplier performance.
6. **CRUD Operations**:
   * Create, read, update, and delete products, suppliers, users, and stock transactions.

**Tech Stack:**

* **Backend**:
  + C# (ASP.NET MVC)
  + SQL Server for data storage
  + Entity Framework (for ORM)
* **Frontend**:
  + HTML, CSS, JavaScript
  + Bootstrap for responsive design

**Database Design:**

You'll need a few tables in your SQL Server database:

1. **Users** (UserID, Username, PasswordHash, Role)
2. **Products** (ProductID, Name, Description, Price, StockLevel, Category, SupplierID)
3. **Suppliers** (SupplierID, Name, ContactInfo, Address)
4. **StockTransactions** (TransactionID, ProductID, Date, Quantity, TransactionType) — Track stock adjustments.
5. **Sales** (SaleID, ProductID, Date, QuantitySold, TotalAmount)

**Step-by-Step Implementation:**

1. **Create the Database**:
   * Start by designing the SQL Server database schema based on the requirements (tables for Users, Products, Suppliers, etc.).
   * Use SQL Server Management Studio (SSMS) to create the tables and relationships.
2. **ASP.NET MVC Setup**:
   * Create a new ASP.NET MVC project in Visual Studio.
   * Set up authentication and user roles (using ASP.NET Identity for user management).
   * Implement the controller actions to interact with the database via Entity Framework (using LINQ to query and update the data).
3. **Product Management UI**:
   * Design the views using Razor Pages to show product data, with options to add, edit, and delete products.
   * Implement forms for adding/editing product details and validating inputs (e.g., product name, price, stock quantity).
4. **Stock Management**:
   * Implement stock tracking using transactions: each time stock is adjusted (added or sold), create an entry in the StockTransactions table.
   * Include a function to alert users if stock is low (based on a set threshold).
5. **Reports and Analytics**:
   * Implement report generation features (e.g., total sales, current inventory, low-stock products).
   * Display these reports in a readable and interactive format (tables, charts).
6. **Frontend & UI Design**:
   * Use Bootstrap or Material Design for the user interface to make it responsive and visually appealing.
   * Focus on a clean and user-friendly UI with easy navigation.
7. **Deploy the Application**:
   * Deploy the app on **IIS** or **Azure** to showcase it to potential employers.
   * Provide access to the backend for employers to test various functionalities (e.g., managing inventory, generating reports).

**Bonus Features (to make the project stand out):**

* **Email Notifications**: Send an email when stock goes low, a product is added, or a sale happens.
* **Search and Filtering**: Allow users to search for products and filter by category, price range, or stock level.
* **Audit Trail**: Log and track changes made to products and stock, showing who made each change and when.
* **API Integration**: Implement an API to allow external systems (e.g., an e-commerce platform) to retrieve inventory data.

**Why This Project Will Help:**

* **Full-Stack Development**: You’ll showcase your ability to build both the frontend and backend of a web application using **ASP.NET MVC**.
* **SQL Server**: Demonstrates your ability to design and manage relational databases.
* **Security Features**: Implementing authentication and role-based authorization shows that you understand basic security concepts.
* **Real-World Use Case**: Inventory management is a common application in businesses, and this system can be easily adapted to many industries, making it a valuable project.
* **Professional Presentation**: Having a project that looks professional and is easy to use will help you stand out to employers.

**Deployment and GitHub:**

1. **Deploy to Azure**: This will make your application accessible online and show you know how to deploy applications.
2. **GitHub Repository**: Share the project on GitHub with clear documentation (README file) explaining the setup, features, and any additional configuration needed.

This project provides a solid foundation to demonstrate your ASP.NET MVC and SQL Server skills to potential employers. Would you like a sample structure or any specific code snippets to help you get started?