**Product Management System Documentation**

**Overview**

The Product Management System is a simple application designed for managing products using a modern tech stack:

* Backend: .NET 8 Minimal API with Entity Framework Core and SQL Server.
* Frontend: Angular 18 (standalone components) with PrimeNG and TailwindCSS.

The system allows CRUD operations on products and includes features like search, sorting, and validation.

**Backend Details**

**Technologies Used:**

* Minimal API: Lightweight and modern API design.
* Entity Framework Core: For database operations.
* SQL Server: Database storage.
* FluentValidation: For request validation.

**API Endpoints**

***1. Create Product***

Endpoint: POST /api/Products/CreateProduct

**Request Body:**

* Name: string (required, max 100 characters)
* Description: string (required, max 500 characters)
* Price: decimal (required, must be positive)
* ImageUrl: file (required)

\*\*Response:\*\*



***2. Get All Products***

**Endpoint**: GET /api/Products/GetAllProducts

**Request Body**:

* Name: string (required, max 100 characters)
* Description: string (required, max 500 characters)
* Price: decimal (required, must be positive)
* ImageUrl: file (required)



***3. Get Product By ID***

**Endpoint**: GET /api/Products/GetProductById/{id}

**Request Body**:



***4. Update Product***

**Endpoint**PUT /api/Products/Update/{id}

**Request Body**:

* Same as "Create Product"



***5. Delete Product***

**Endpoint**PUT /api/Products/Delete/{id}



**Frontend Details**

**Technologies Used:**

- Angular 18: With standalone components for modular development.

- PrimeNG: For rich UI components like DataTable.

- TailwindCSS: For styling.

**Features**

1. **Product List Page**:

* Displays products using PrimeNG DataTable
* with sorting and search functionality.

**2. Create Product Page:**

* Form with validation and testing features.

**3. Edit Product Page:**

* Pre-filled form with testing features.

**4. Delete Product:**

* Includes a confirmation dialog.

**Example Code**

* Fetching Products (Service)





**Setup Instructions**

**Backend**

1. Clone the repository.

2. Run `dotnet restore` to install dependencies.

3. Update the connection string in `appsettings.json`.

4. Run migrations using `dotnet ef database update`.

5. Start the API using `dotnet run`.

**Frontend**

1. Navigate to the Angular project directory.

2. Install dependencies using `npm install`.

3. Run the application using `ng serve`.

**Notes**

- Ensure SQL Server is running and accessible.

- Adjust the CORS policy in the backend for frontend communication.

- Added middleware for error handling in the backend.