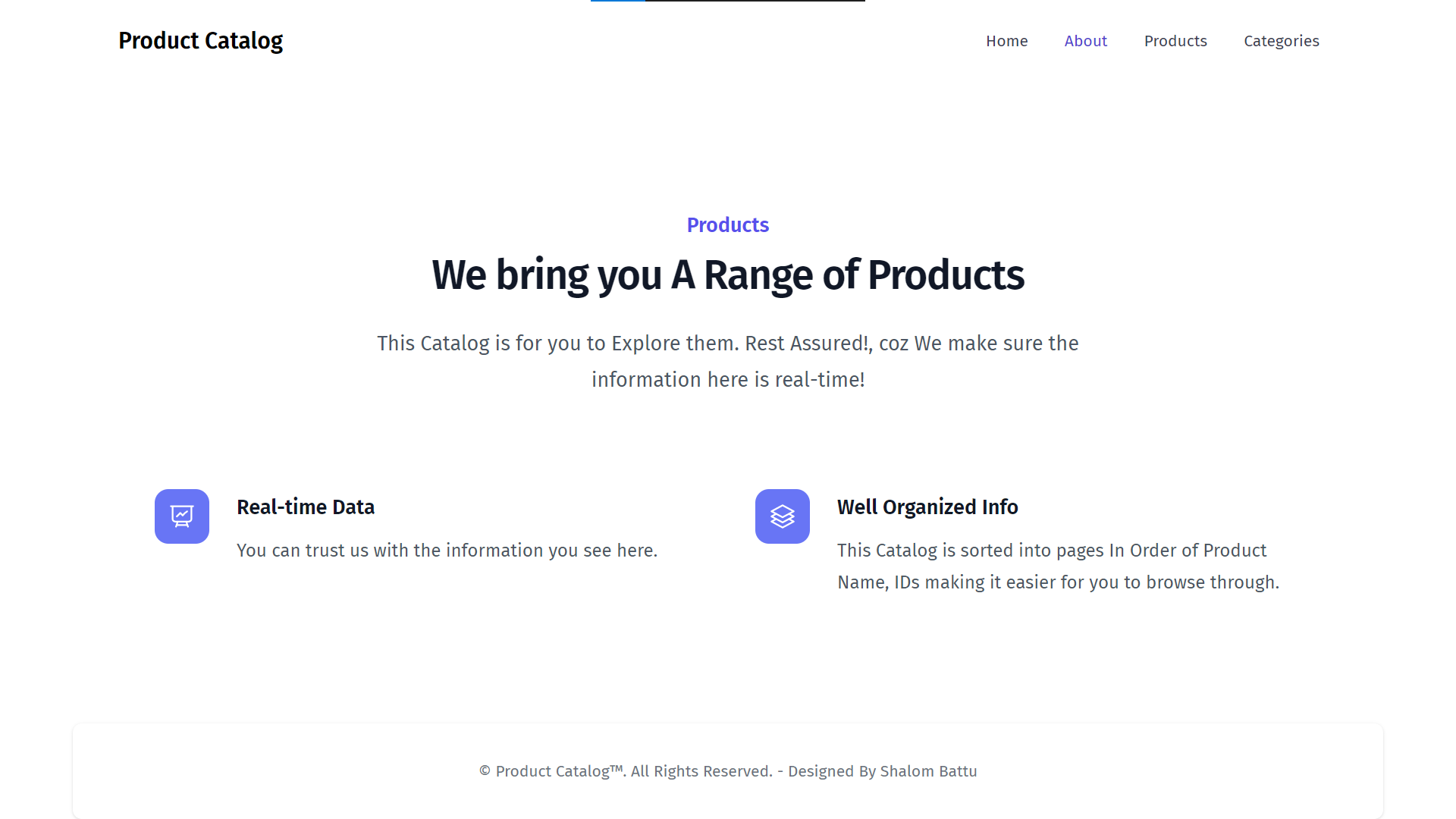
**Product Catalog Application**

(Technical Documentation)



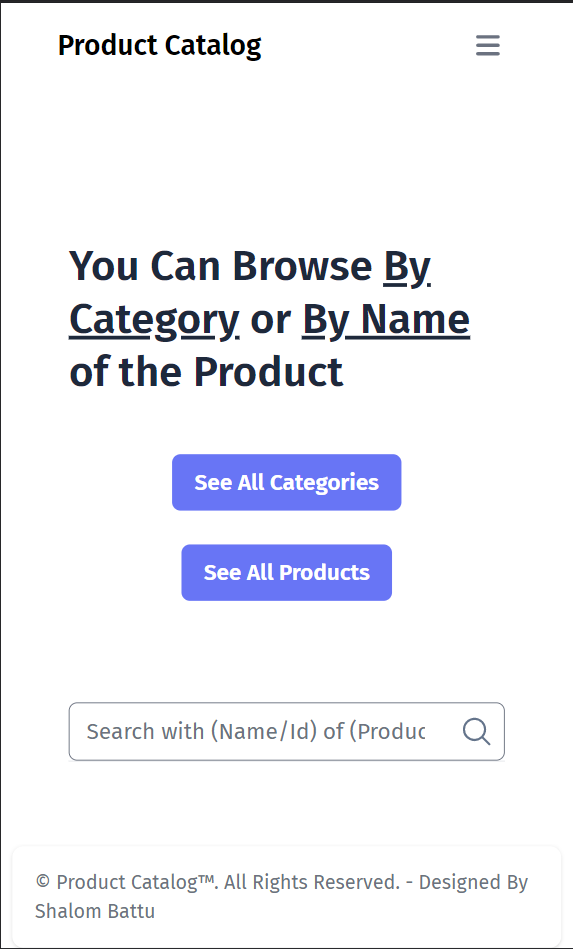


Table of Contents

[Document Control 3](#_Toc127201805)

[Project Overview 3](#_Toc127201806)

[Technologies used 3](#_Toc127201807)

[Data Flow 4](#_Toc127201808)

[Functions 5](#_Toc127201809)

[All the Different Views 6](#_Toc127201810)

# **Document Control**

|  |  |
| --- | --- |
| **Project Name** | Product Catalog Application |
| **Mentor** | Satish Kumar Vadlavalli |
| **Delivery Manager** | Deepthi R. |
| **Business Analyst** | Aruna R. |
| **GIT Repo** | [Link to the Repository](https://github.com/BanriSh777/practice-task1-ngdnc) |

# **Project Overview**

This is A Catalog Application provides one to Explore through the products of different categories

Add and have a glance at the catalog of your products in two organized formats:

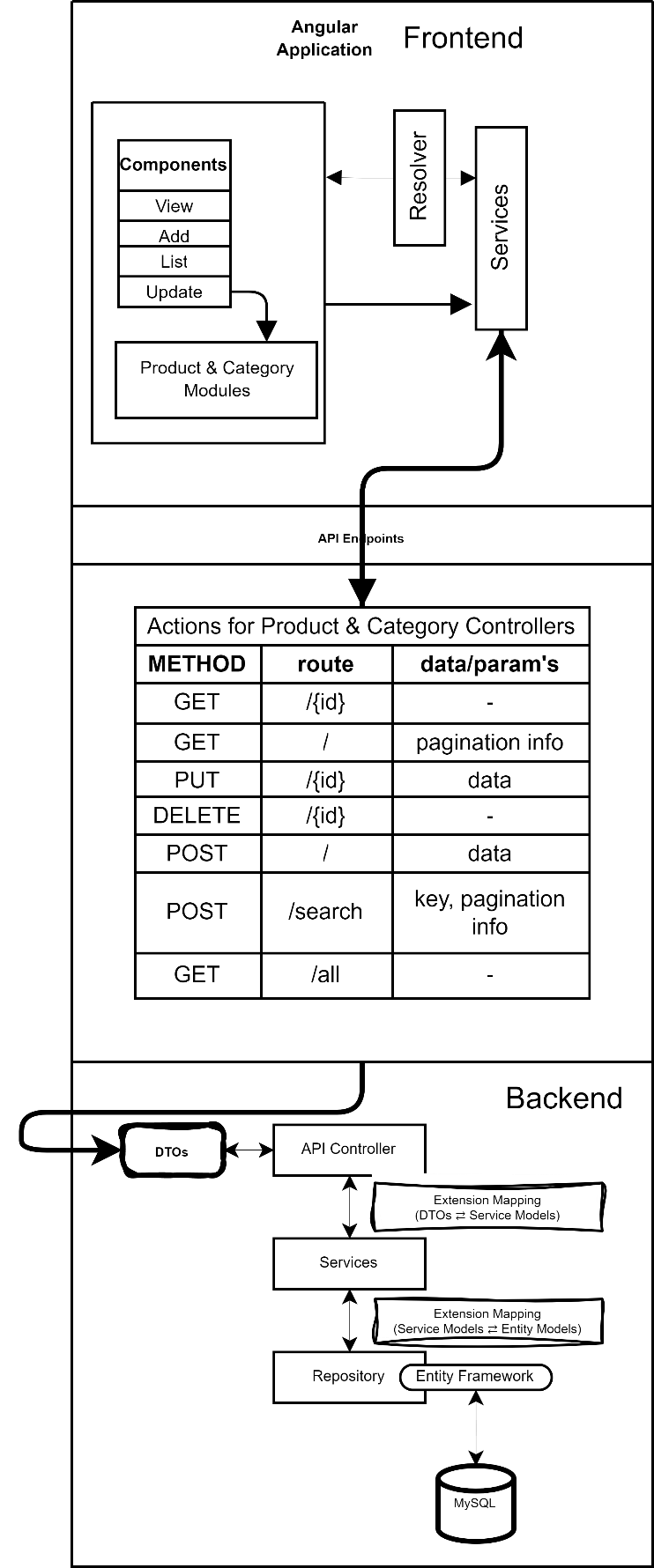
* Alphabetically sorted Lists of Categories and Products
* View the products Category wise.
* Search with the names and ids of products/categories

All Basic **CRUD Operations** and Search with **pagination** provided

# **Technologies used**

1. **Front-end Technologies**
2. Angular
   1. *Resolvers*
      * + Used resolver route Guard to make sure data is pre-fetched before navigating to the respective route.
   2. *Services*
      * + Injectable Services in resolvers are used to fetch data from API
   3. Lazy loading
      * + To improve the load time of the application lazy loading is used, enabling bundles of the app at routes to only be loaded when needed
3. Tailwind CSS
   * + - A utility-first CSS framework packed with classes – used this for a Responsive Design
4. RxJS – Observables of RxJS library are used to work with the stream of asynchronous data
5. **Back-end Technologies**
6. .NET Core Web API
   1. *Dependency Injection* – DI is used to loosely couple the code so that managing future changes would be easier.
   2. *Entity Framework –* Communication b/w relationaland Objects/Models by Mapping automatically – ORM
   3. *Monolithic 3-layered Architecture* – Maintainable pattern of different layers is used as project structure, this self-contained nature of the app makes it easier to debug, manage and maintain the code.
   4. *Mappers* – Conversion b/w Data Models exposed to API user, the DTOs and Service layer Models & Entity Models
7. MySQL Database

# **Data Flow**

****

# **Functions**

**Listing** all the Entities in ***Pages, Sort, Search, Create, Update, Delete.***

|  |  |
| --- | --- |
| **Sort Function** | Search Function |
| Pagination | |
| Create  . Add Product    . Add Category | |
| Update | |

# **All the Different Views**

1. **Create**, **Update**, **List** Views for both Products as well as Categories
2. **Entity View**: *View Product, View Category*

