

# Boxer Product Management System

This document is intended to for anyone who might be required to have a technical understanding of to the implemented solution by outlining high level design and requirements specification.

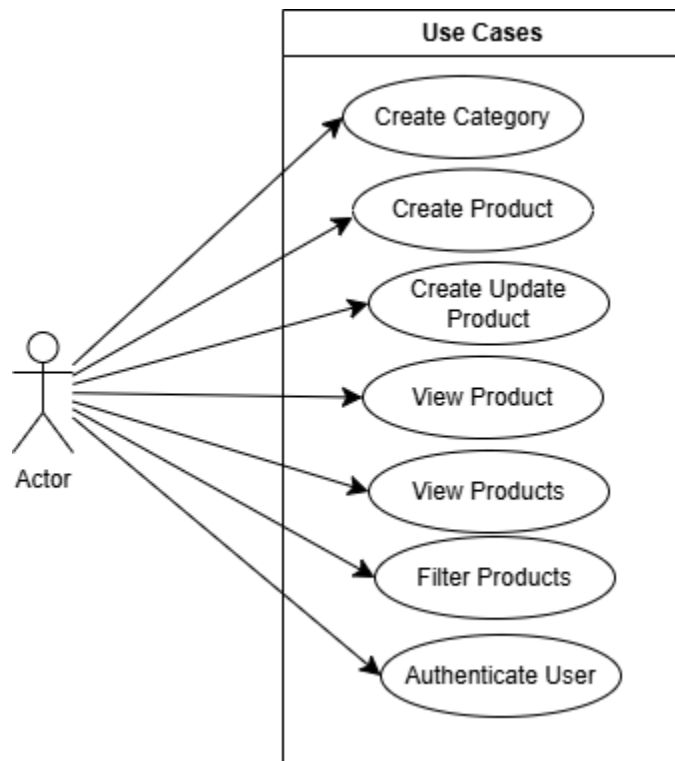
## Summary

The documents outlines the high-level design and specification for the implemented system. The current system is for managing products offered by the business. The system is developed following the Microservice pattern currently with three microservices, Gateway, Security and Product management. The code was developed using c# dotnet core version 9, angular version 25 and ionic 8. The application makes use of two SQL databases and entity core for all interactions. There is a database stored procedure used for filtering the products grid performing SQL custom select query.

## Table of Contents

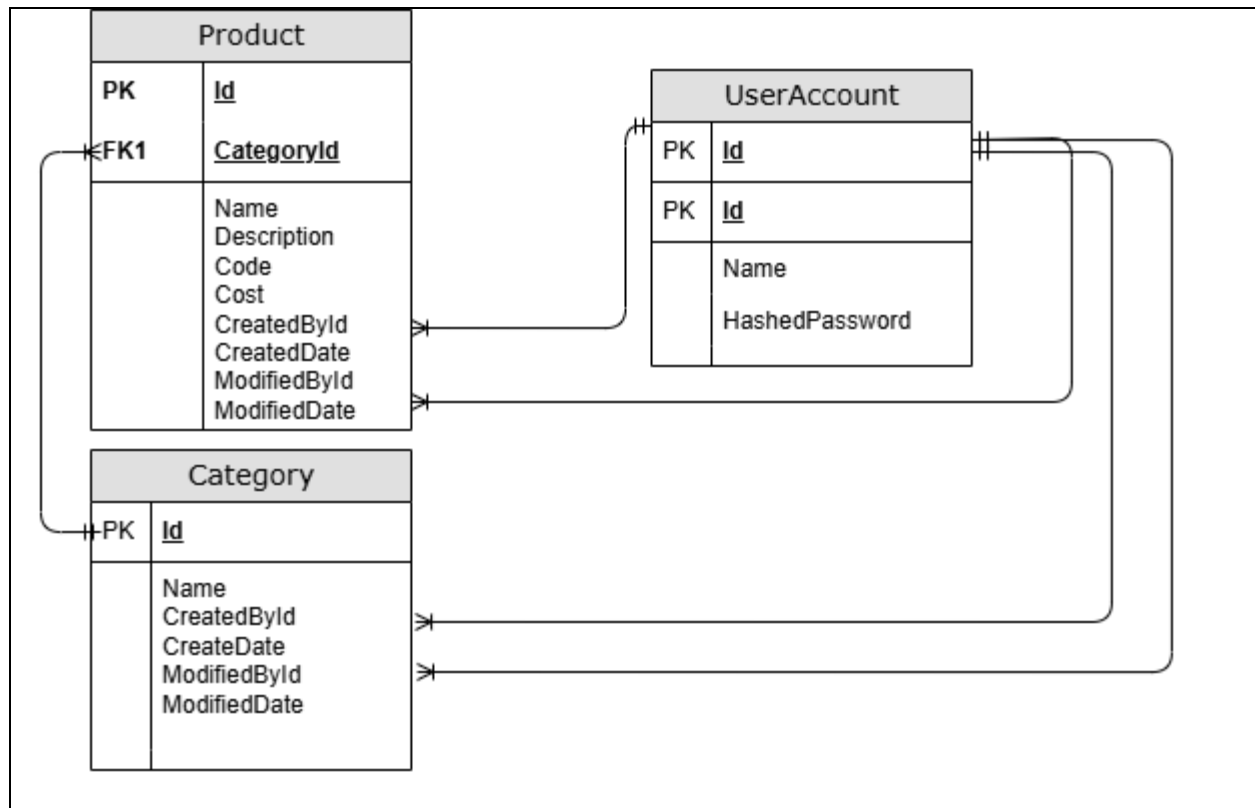
Boxer Product Management System .....	1
Summary .....	1
Use Cases .....	3
Database Model .....	4
Operational Architecture View .....	5
The microservices components .....	6
Data .....	6
Service .....	6
Model .....	6
API .....	6
Setting up development Environment .....	6
NVM .....	7
<b>Angular</b> .....	7
Ionic Framework .....	8
Database migration .....	8
Database backup files .....	9
Login validation .....	9
User credentials .....	9
Scala Documentation .....	10
Docker .....	10
Further Exploration and up-skilling .....	10
References .....	11

## Use Cases



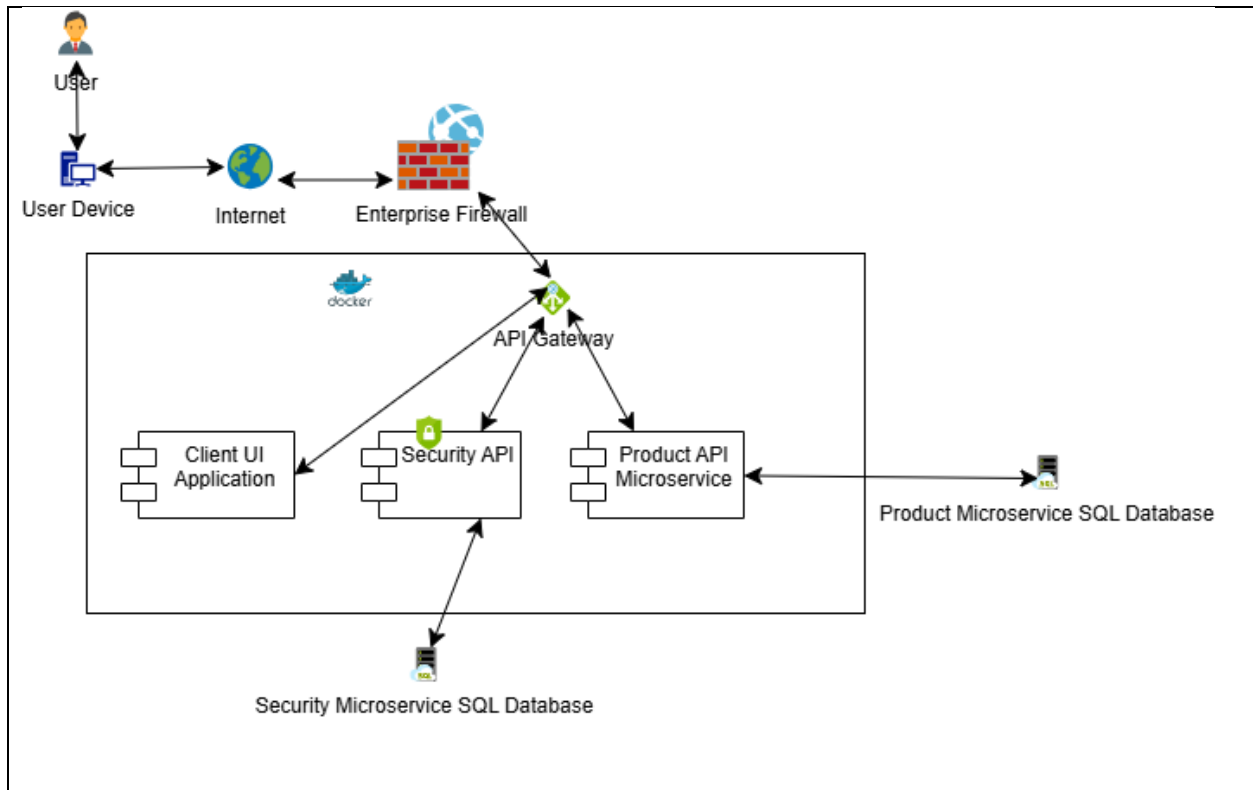
The application allows the user to perform functions shown in the use cases above.

## Database Model



The model above shows the current application tables. Though the model depicts as a single model. The tables are portioned onto different database instances for Security and Product management APIs. The tables provided by dotnet core identity are excluded from model though present on the schema as they are scaffolded by framework.

## Operational Architecture View



The application is currently made up of three microservices, Gateway API, Security API and Product API.

The gateway API applies required Authentication checks for token and restricts downstream traffic where the required valid token is not provided.

The routing to different Microservices is handled in the Gateway.

There are two separate databases for Security and Product APIs.

# The microservices components

The microservices have and not limited to components listed and briefly described below.

## Data

Provides a Database configuration and access

## Service

Interacts with Data and process any required logic for the controller classes.

## Model

Entity definition for data and data/view objects

## API

The integrating component with controller classes. Dependencies are injected via class constructors. Http requests are terminated in the API classes to process client requests.

# Setting up development Environment

The Project was implemented on a Windows 11 Pro operating system, i7 64bit, 16Gb RAM.

Microsoft SQL Server 2022

Visual Studio 2022

Asp.net Core Web API 9, Windows Docker enabled

Uses Entity Framework Core

## NVM

Please use Node version manager(NVM) to install node.js which is required for Angular client application. NVM will enable the User to target multiple node version and update using script commands.

*nvm install node* - will install the latest node.js version.

Note the error below when installing angular with latest version node.js “nvm use node” for latest node version.

```
C:\Source\Boxer\Boxer.Management.Product.Client>ng new boxer.management.product.client
Node.js version v22.2.0 detected.
The Angular CLI requires a minimum Node.js version of v20.19 or v22.12.

Please update your Node.js version or visit https://nodejs.org/ for additional instructions.

C:\Source\Boxer\Boxer.Management.Product.Client>nvm use --lts
Error retrieving "https://nodejs.org/dist/latest---lts/SHASUMS256.txt": HTTP Status 404

C:\Source\Boxer\Boxer.Management.Product.Client>nvm use --lts
Error retrieving "https://nodejs.org/dist/latest---lts/SHASUMS256.txt": HTTP Status 404

C:\Source\Boxer\Boxer.Management.Product.Client>nvm use node
25.2.1
Now using node v25.2.1 (64-bit)

C:\Source\Boxer\Boxer.Management.Product.Client>ng new boxer.management.product.client
Node.js version v25.2.1 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/previous-releases/.
✔ Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ]
✔ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
✔ Do you want to create a 'zoneless' application without zone.js? No
```

## Angular

After installing Node and NVM can run command below to install latest version of angular

`npm install -g @angular/cli`

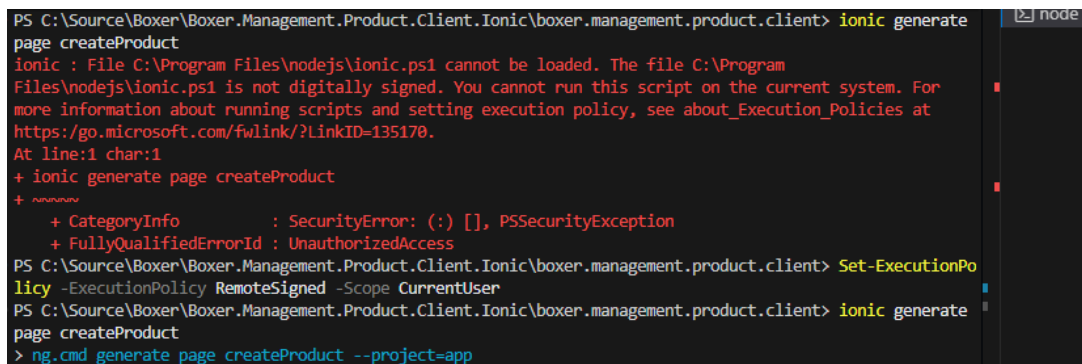
## Ionic Framework

Run command below to install Ionic Framework which works with Angular for client UI development.

```
npm install -g @ionic/cli
```

```
ionic start boxer.management.product.client blank --type angular
```

Please note of insert below with possible exceptions and commands to resolve.



```
PS C:\Source\Boxer\Boxer.Management.Product.Client.Ionic\boxer.management.product.client> ionic generate
page createProduct
ionic : File C:\Program Files\nodejs\ionic.ps1 cannot be loaded. The file C:\Program
Files\nodejs\ionic.ps1 is not digitally signed. You cannot run this script on the current system. For
more information about running scripts and setting execution policy, see about_Execution_Policies at
https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ ionic generate page createProduct
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Source\Boxer\Boxer.Management.Product.Client.Ionic\boxer.management.product.client> Set-ExecutionPo
licy -ExecutionPolicy RemoteSigned -Scope CurrentUser
PS C:\Source\Boxer\Boxer.Management.Product.Client.Ionic\boxer.management.product.client> ionic generate
page createProduct
> ng.cmd generate page createProduct --project=app
```

## Database migration

Please Install from nuget package manager

Microsoft.EntityFrameworkCore,

Microsoft.EntityFrameworkCore.SqlServer,

Microsoft.EntityFrameworkCore.Tools

- Add Database configuration context definition file to root folder inheriting from IdentityDbContext for Default Identity Framework
- Add Database configuration context definition file to root folder inheriting from DbContext
- Add Database connection settings to appsettings file
- Define UserAccount that Inherits from Microsoft.AspNetCore.Identity Identity User



- Run commands to create migrations, update the database and remove used migrations.

`dotnet ef migrations add InitialCreate`

`dotnet ef database update`

`dotnet ef migrations remove`

Examples commands below for database update on the Root API folders.

*`dotnet ef database update --context Boxer.Management.Product.Data.DatabaseContext`*

*`dotnet ef migrations add InitialCreate --context Boxer.Management.Product.Data.DatabaseContext`*

## Database backup files

Two back files will be made available as part of the solution and can be restored for application to run.

## Login validation

The application uses dotnet core signinmanage and usermanager for creating users and authentication.

SignInManager by default validates password minimum length, checks for number, special character and upper case letter.

The UserManager hashes password and validates provided credentials.

### User credentials

Username:user002 or user001

Password: A1!iousafesdd

# Scala Documentation

Please navigate to

<https://localhost:7117/scalar/v1> Product API

[Scalar API Reference](#) Security API

## Docker

Docker desktop for localhost can be downloaded and installed the docker website.

The commands for docker build are as follows,

Security API

```
docker build -t boxer.management:7168/security.authentication:0.0.1 .
```

Product API

```
docker build -t boxer.management: 7168/product:0.0.1 .
```

Gateway API

```
docker build -t boxer.management: 7168/security.gateway:0.0.1 .
```

## Further Exploration and up-skilling

URL links are listed below under references for articles that may be useful in providing a technical foundation of the implemented solution?

## References

<https://codewithmukesh.com/blog/aspnet-core-webapi-crud-with-entity-framework-core-full-course/>

<https://medium.com/@v4sooraj/building-a-minimal-api-in-net-core-9-0-for-product-management-43ee1b984196>

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-10.0>

<https://learn.microsoft.com/en-us/ef/core/miscellaneous/connection-strings?tabs=dotnet-core-cli>

<https://www.telerik.com/blogs/new-net-8-aspnet-core-identity-how-implement>

<https://learn.microsoft.com/en-us/azure/architecture/web-apps/guides/enterprise-app-patterns/modern-web-app/dotnet/guidance>

<https://medium.com/@v4sooraj/building-a-minimal-api-in-net-core-9-0-for-product-management-43ee1b984196>

<https://github.com/zangassis/publish-cms/blob/main/Content.ApiGateway/ocelot.json>

<https://learn.microsoft.com/en-us/dotnet/core/compatibility/aspnet-core/8.0/securitytoken-events>

[Write custom ASP.NET Core middleware | Microsoft Learn](#)

<https://github.com/bezkoder/angular-17-pagination-example>

<https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-10.0&tabs=visual-studio>