**Table of Contents**

1. **Introduction** …….…………………………………………………………………………………………………….02

* Purpose

1. **Learning** **Objectives** ….………………………………………………………………………….………….…….02
2. **Background** …………………………………………………………………………………………………………….02

* Microservices Architecture
* ASP.NET Core WebAPI
* Ocelot Gateway
* Angular Framework
* JWT Authentication
* Co-pilot Introduction

1. **Creating** **Microservices** …………………………….…………………………………………………………….02

* Designing Microservices Structure
* Implementing Business Logic
* Exposing Endpoints in WebAPIs
* Testing Microservices Independently

1. **Integrating Angular Frontend** ………………………………………………………………………………….03

* Creating Angular Components and Modules
* Consuming Microservices via Ocelot
* Displaying Data in UI

1. **JWT Authentication and Co-pilot**.……………….………………………………………………………….03

* Introduction to JWT and Authentication
* Adding Authentication Middleware in WebAPIs
* Secure Communication between Angular and WebAPIs
* Co-pilot Assistance

1. **Testing and Debugging** …………………………………………………………………………………………….04

* End-to-End Testing Angular UI
* End-to-End Testing Microservices

1. **Conclusion** ………………………………………………………….……………………………………….………….04

* Summary of Key Learnings

1. **References and Resources** ……………………….……………………………………………….…………….05

* Articles, Tutorials
* Online Documentation

1. **Introduction**

* **Purpose**

The POC is about ASP.NET Core v7 WebAPI Microservices with Angular v14 and JWT Authentication. The purpose of this POC was to have a maximum use of Co-pilot tool. This tool decreases the developing time by giving suggestion based on the code logic and generates code based on the given prompt.

1. **Learning Objectives**

By the end of this POC, I am able to learn and understand following things:

* Understand the fundamentals of microservices architecture.
* How to create ASP.NET Core WebAPI projects for microservices.
* Configure and use Ocelot as an API Gateway.
* Develop an Angular frontend that consumes microservices.
* Implement JWT authentication for secure communication.
* Utilize Co-pilot to enhance coding productivity.

1. **Background**
2. Microservices Architecture
3. ASP.NET Core WebAPI v7
4. Ocelot Gateway
5. Angular 14 Framework
6. JWT Authentication
7. C-pilot AI tool
8. **Creating Microservices**
9. **Designing Microservices Structure**

I created Microservice architecture for 3 WebAPIs namely Product, Order and Customer. And accessed them all with the use of Ocelot WebAPI Gateway.

1. **Implementing Business Logic**

I used 3 different design patterns for the services in my Microservices app.:

1. CustomerAPI: Generic Repository Pattern
2. OrderAPI: EF Core Code First Approach Pattern
3. ProductAPI: CQRS and MediatR Pattern
4. **Exposing Endpoints in WebAPIs**

I used Ocelot API Gateway to make connection between every service. And also learned that working of Upstream, Downstream, Ports and routes in Ocelot Gateway.

1. **Testing Microservices Independently**

Run and tested each service individually and as a whole. Tested the server-side code with Swagger and Postman and client-side tested the whole app.

1. **Integrating Angular Frontend**
2. **Creating Angular Components and Modules**

Created 5 components namely Login, Homepage, Product, Order and Customer. Also created modals for Product, Order and Customer.

1. **Consuming Microservices via Ocelot**

Made 3 services namely Product, Order and Customer to access data from 3 WebAPIs and can perform CRUD Operations seamlessly.

1. **Displaying Data in UI**

The data from WebAPI was bind with the html through typeScript file of every Services. Displayed the data in a tabular format.

1. **JWT Authentication and Co-pilot**
2. **Adding Authentication Middleware in WebAPIs**

I created a class library named JwtAuthenticationManager and gave its reference to every WebAPI. In this class library I added models of UserLogin, Response and Request and also added JwtHandler. This library will generate the Jwt Token.

1. **Secure Communication between Angular and WebAPIs**

I created a WebAPI namely JwtAuthenticationAPI which will have reference of the JwtAuthenticationManager class library and will help to pass generated token to Angular.

And in Angular, I accessed those tokens through guard and interceptor and stored it in local storage. Each API will be able to access the token as I have stored it in header.

1. **Co-pilot Assistance**

Co-pilot was able to give accurate code snippets and code suggestions based on my coding logic and prompts. Sometimes the suggestions were not up to the mark so at that time I hard coded the logic and brought the output.

1. **Testing and Debugging**
2. **End-to-End Testing Angular UI**

Tested CRUD Operations of each Service with Angular. The JWT Auth implantation is done from client-side as well to access the app for particular users only.

1. **End-to-End Testing Microservices**

Tested CRUD Operations of each Service with Swagger and Postman. The JWT Auth implementation is done from server-side as well to access the app for particular users only. Also implemented Role-based Login to get the Role specific output for users.

1. **Conclusion**

**Summary of Key Learnings**

In this learning journey through POC, I’ve gained understanding of building a microservices architecture with ASP.NET Core WebAPI, Angular, and JWT authentication using Co-pilot assistance. Here are some of my takeaways:

1. Microservices Knowledge: I've gained understanding of the architecture, including their benefits, challenges, and design principles and how does it differ from monolithic architecture.
2. WebAPI Knowledge: I've learned how to create ASP.NET Core WebAPIs using 3 different design patterns, and how the implementation of business logic of all those 3 WebAPIs is done within microservices.
3. Gateway Knowledge: With Ocelot, I've came to know the art of building an API Gateway that routes and balances requests, enhancing the scalability and appearance of your microservices ecosystem. The importance of Upstream and Downstream and the port and routes is very essential to make a microservice.
4. Frontend Knowledge: I've successfully integrated Angular into my architecture, crafting a user-friendly interface that communicates with your microservices via the API Gateway. And also implemented JWT Auth for the Authentication of User. With which I cam to know the deep meaning between Authorization and Authentication.
5. Secure Communication: By implementing JWT authentication, I've fortified communication between my Angular and microservices, ensuring data integrity and user privacy. I have implemented JWT Auth from both the side client and server so we can test them individually and as a whole.
6. Co-pilot Collaboration: I've witnessed the power of Co-pilot to streamline my development process, generating code snippets and solutions that accelerate my workflow. I used Co-pilot in my Visual Studio and in my Visual Studio Code. This helped to get faster suggestions for my logic of particular code snippets and also to generate the logic based on my given prompts.
7. **References**

I took reference from various google websites, articles and YouTube videos. I am mentioning some of them here:

1. **Backend**
2. ProductAPI: CQRS and MediatR design pattern:

<https://www.c-sharpcorner.com/article/cqrs-and-mediatr-pattern-implementation-using-net-core-6-web-api/>

1. OrderAPI: Code First EF Core design pattern:

<https://www.c-sharpcorner.com/article/create-asp-net-core-web-api-with-entity-framework-code-first-approach/>

1. CustomerAPI: Generic Repository design pattern:

<https://medium.com/@niteshsinghal85/reducing-code-with-generic-repository-pattern-in-asp-net-core-api-ba611f7c4ab2>

1. **JWT Authentication:** <https://www.youtube.com/watch?v=P2osfctiHAc>
2. **Frontend:** <https://blog.devops.dev/microservices-implementation-with-ocelot-gateway-using-net-core-6-api-and-angular-14-64b2ce3248ee>
3. **Some other useful links:**
4. **Ocelot gateway understanding:**

<https://myview.rahulnivi.net/api-gateway-ocelot/>

<https://buildmedia.readthedocs.org/media/pdf/ocelot/latest/ocelot.pdf>

1. **Full stack App using ASP.NET, Angular and JWT:**

<https://www.youtube.com/watch?v=vvPenp70NgA>