# MicroService Assignment

**Product Management:** Represent the main project namespace or assembly.

**DbContext** : Manages the database context and includes a DbSet<User> for interacting with the User entities and DbSet<Product> for interacting with the Product entities.

**Interfaces and Repositories:**

* **IUserRepository:** Defines methods for CRUD operations on Users entities.
* **UserRepository:** Implements IUserRepository and provides concrete implementations for the data access operations
* **IProductRepository:** Defines methods for CRUD operations on Users entities.
* **ProductRepository:** Implements IProductRepository and provides concrete implementations for the data access operations

**Services :**

* **User Service**: Implements business logic related to user entities. Depends on IUserRepository for data access operations
* **Product Service**: Implements business logic related to User entities. Depends on IProductRepository for data access operations
* **Inventory Service**: Handles inventory management including adding/removing products.
* **Cart Service**: Manages shopping cart operations.
* **Order Service**: Handles order processing and checkout functionality.

**Identity Server:**

Identity Server is used for handling authentication and authorization in modern applications. **benefits**: Centralized Authentication and Authorization, Support for Modern Security Protocol, Token Issuance and Validation.

**API Gateway:**

An API Gateway like Ocelot is crucial in a microservices architecture for several reasons:

* **Centralized Routing and Load Balancing**: Manages and optimizes request distribution.
* **Simplified Client Interaction**: Provides a unified API surface, reducing client-side complexity.
* **Security**: Centralizes and enforces authentication and authorization.
* **Cross-Cutting Concerns**: Handles logging, monitoring, rate limiting, and caching.
* **Protocol Translation**: Manages different communication protocols.
* **Flexibility and Extensibility**: Allows custom middleware and scaling capabilities.

Using Ocelot ensures a more maintainable, scalable, and secure microservices ecosystem by addressing these key aspects centrally.

**Key Relationships:**

* **Dependency Relationship:** User Service depends on IUserRepository for accessing and manipulating User data and Product Service depends on IProductRepository for accessing and manipulating Product.
* **Association Relationship:** DbContext has a composition relationship with DbSet<User> and DbSet<Product> , indicating that it manages a collection of User and Product entities