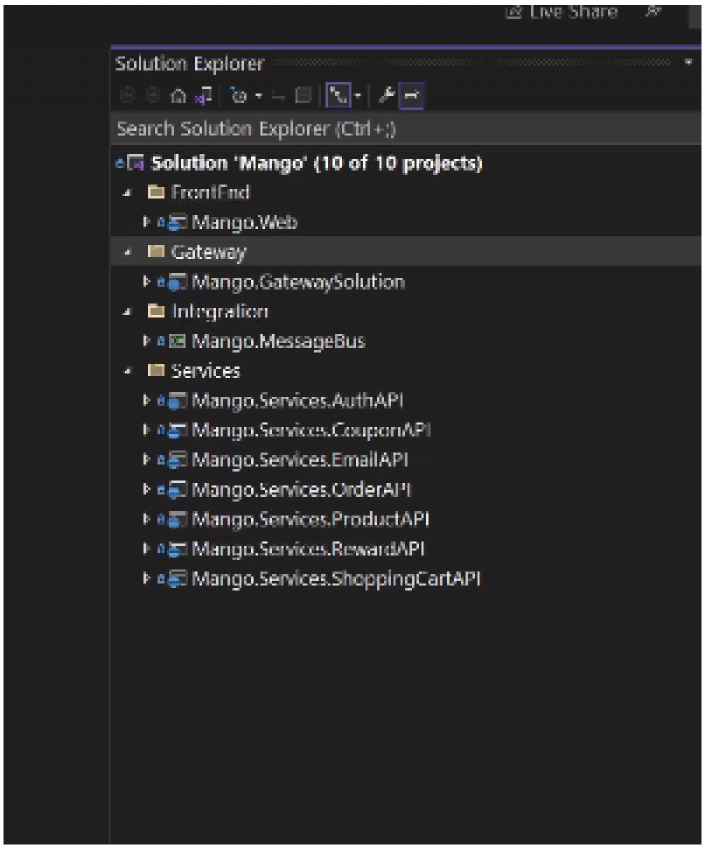
# .Net Microservices Section 1: Introduction

- **Introduction to the Course**
  - Course Overview
    - Purpose and Goals
      - Introduce microservices architecture.
      - Provide real-world project experience.
      - Understand how microservices communicate and solve problems.
    - Instructor Introduction
      - Instructor: Brogan.
      - Course Title: .NET Core Microservices: The Complete Guide.
  - Importance of Microservices
    - Buzzword in Programming
      - Microservices are increasingly important in job markets and system design.
      - Often suggested for lagging or large systems.
    - Motivation Behind the Course
      - Understanding what microservices are and how to implement them correctly.
      - Providing a real-world project to see microservices in action.
- **2. Understanding Microservices**
  - Basic Concept
    - Individual Projects
      - Microservices are composed of individual APIs.
      - Building small APIs and understanding their communication.
    - Key Concepts
      - Individual API communication.
      - How microservices architecture comes together.
  - Course Structure
    - Building APIs

- Seven small APIs, including one for authentication using Dotnet Identity.
- Focus on API communication and solving problems.

- Keeping Content Up-to-Date
  - Ensuring the course content is current to avoid struggles with new versions.
  - Excitement about learning microservices through real-world projects.

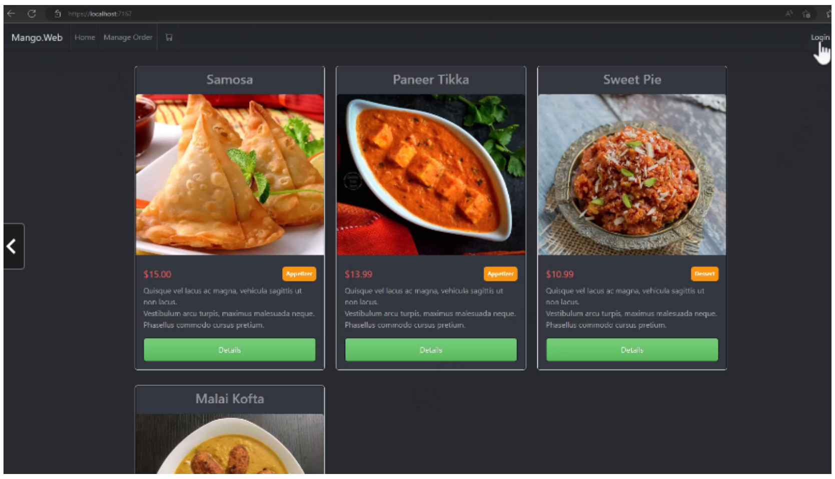
## 3. Course Project Overview

- Application Walkthrough
  - 

```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Mango' (10 of 10 projects)
  FrontEnd
    Mango.Web
  Gateway
    Mango.GatewaySolution
  Integration
    Mango.MessageBus
  Services
    Mango.Services.AuthAPI
    Mango.Services.CouponAPI
    Mango.Services.EmailAPI
    Mango.Services.OrderAPI
    Mango.Services.ProductAPI
    Mango.Services.RewardAPI
    Mango.Services.ShoppingCartAPI
```

- Running the Application
  - Multiple APIs and a web project.
  - Running all microservices and the web project simultaneously.
- User Interaction
  - Logging in as an admin user.

- Landing page with products, login, register buttons, and manage order functionality.

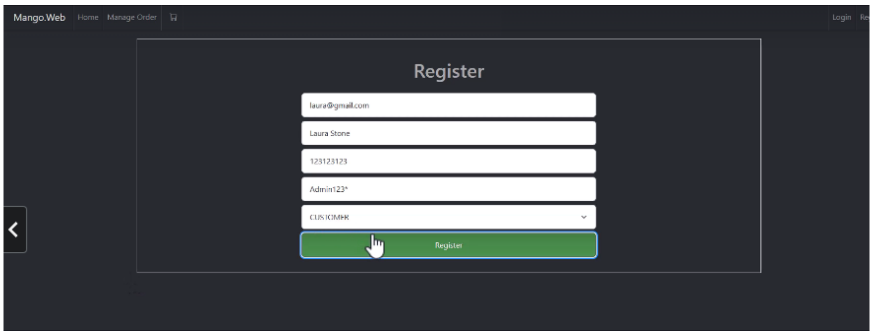


$15.00
Quisque vel lacus ac magna, vehicula sagittis ut non lacus.
Vestibulum arcu turpis, maximus malesuada neque. Phasellus commodo cursus pretium.

$13.99
Quisque vel lacus ac magna, vehicula sagittis ut non lacus.
Vestibulum arcu turpis, maximus malesuada neque. Phasellus commodo cursus pretium.

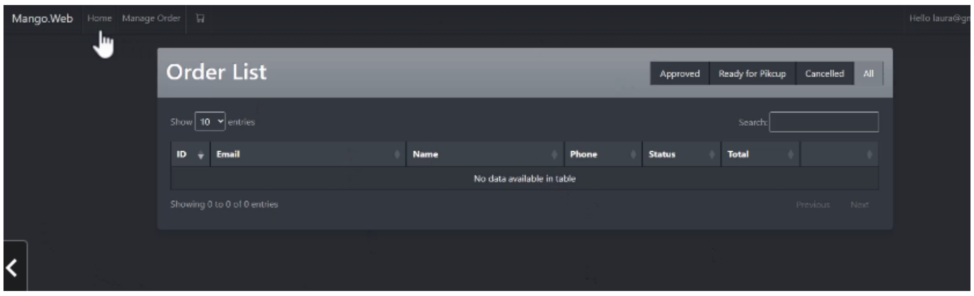
$10.99
Quisque vel lacus ac magna, vehicula sagittis ut non lacus.
Vestibulum arcu turpis, maximus malesuada neque. Phasellus commodo cursus pretium.

- Registering a customer user and logging in.



laura@gmail.com
Laura Stone
123123123
Admin123*
CUSTOMER

- Managing orders, adding items to the shopping cart, applying coupon codes, and emailing the shopping cart.



Show 10 entries
Search:

| ID | Email | Name | Phone | Status | Total | |
|---|---|---|---|---|---|---|
| No data available in table | | | | | | |

Showing 0 to 0 of 0 entries

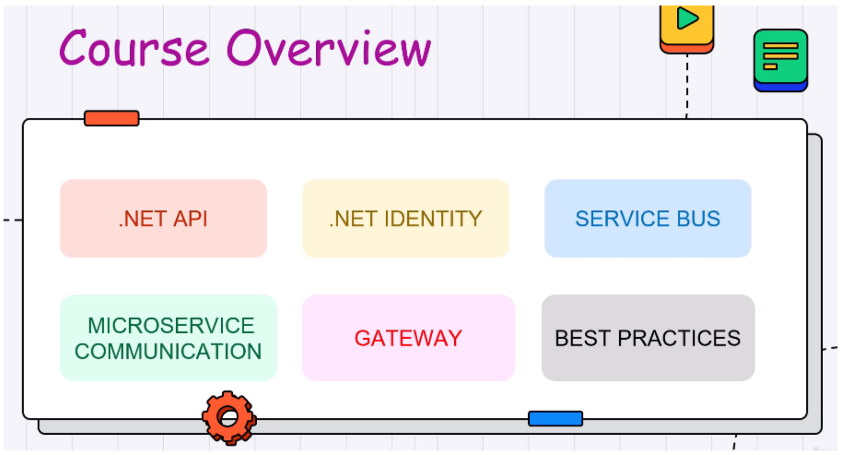
- Placing an order and redirecting to Stripe for payment.
- Completing the order and logging in as admin to manage orders.

Backend Functionality

- Microservice Communication
  - How microservices communicate to manage orders, rewards, and emails.
  - Automatic entries in rewards and email tables upon order placement.
- Complexities and Background Processes
  - Understanding the background processes and communications when placing orders or registering users.

- Focus on the final application and the microservices architecture.

## 4. Course Content Overview



- Building APIs with .NET Core
  - Authentication and Authorization
    - Building an API for authentication using .NET Identity.
    - Familiarity with basics of API and CRUD functionalities with Entity Framework Core.
  - Recommended Prerequisites
    - Free course on Dotnet Mastery for fundamentals of building a .NET API with Entity Framework Core.
- Communication Between APIs
  - Azure Service Bus
    - Understanding queues, topics, subscriptions, and messaging.
    - Requires an Azure subscription with minimal cost.
  - Microservice Communication
    - Getting comfortable with microservice communication using Service Bus.
- Gateway and Deployment
  - Ocelot Gateway
    - Encapsulating microservices behind a gateway using Ocelot.
    - Deploying all code on Azure to see how microservices come together.
  - Best Practices
    - Focus on best practices to avoid common issues and pitfalls.
    - Intentionally introducing bugs to demonstrate corrections.

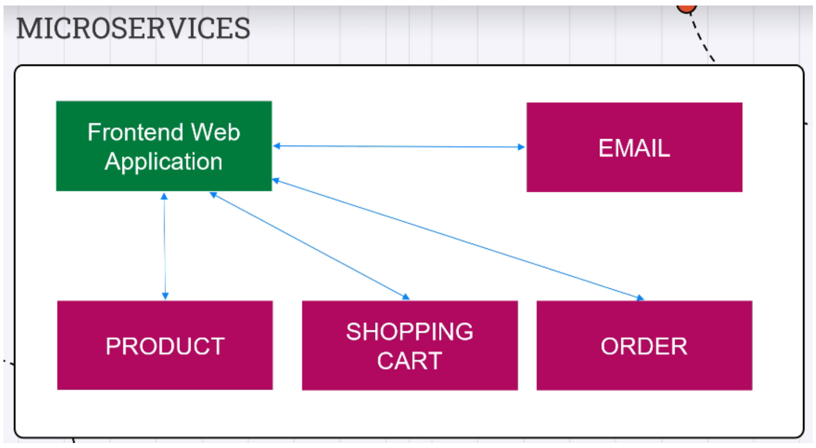
- Learning Approach
  - Lengthy Course
    - Covering a wide variety of topics with microservices.
    - Taking breaks and not overstressing to finish the course quickly.
    - Remembering the importance of slow and steady learning.

## 5. Advantages of Microservices

- Independent Deployability
  - Deploying Individual Microservices
    - Each microservice can be deployed individually without dependencies on other parts of the application.
    - Teams can work on and deploy microservices independently.
  - Comparison with Monolithic Applications
    - Monolithic applications require deploying the complete project.
    - Microservices allow for deploying small, individual components.
- Scalability
  - Efficient Scaling
    - Scaling specific microservices that need more resources.
    - Example: Scaling a microservice for bulk exports without affecting other microservices.
  - Comparison with Monolithic Applications
    - Monolithic applications require scaling the entire server.
    - Microservices allow for scaling individual services efficiently.
- Code Base and Fault Isolation
  - Smaller Code Base
    - Each microservice has a smaller code base compared to monolithic applications.
    - Microservices are typically responsible for one functionality.
  - Fault Isolation
    - Individual microservices can go down without affecting the entire application.
    - Debugging and bringing a microservice back online is faster, reducing downtime.

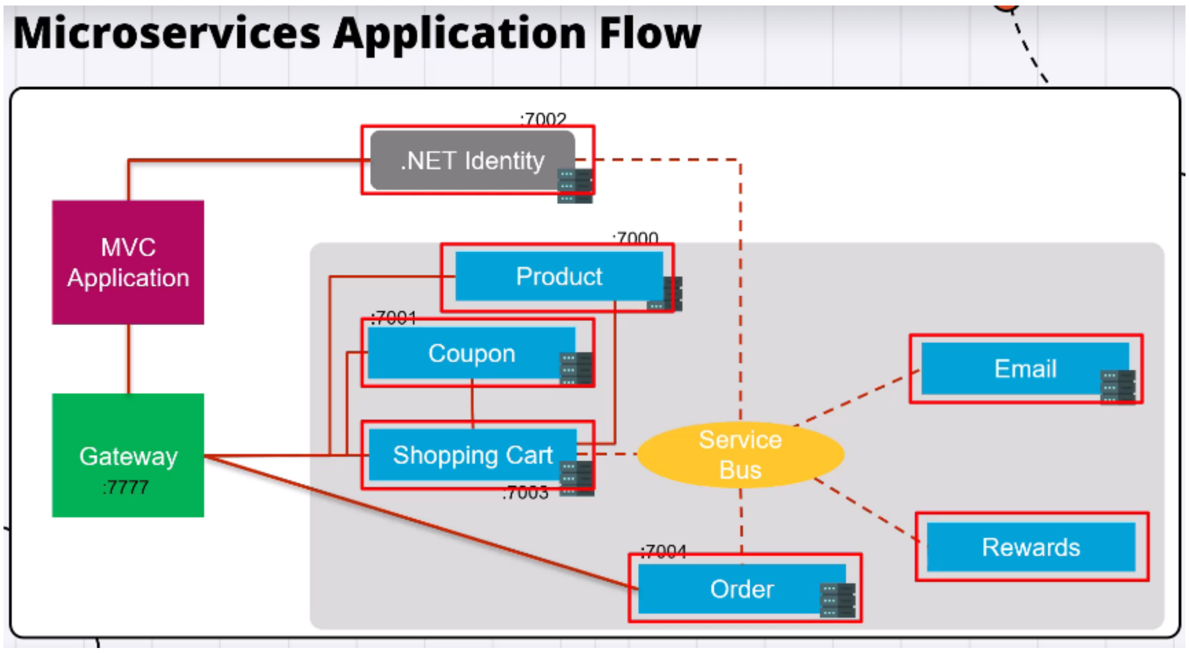
## 6. Example of Microservices Architecture

- Front-End Application with Multiple Microservices

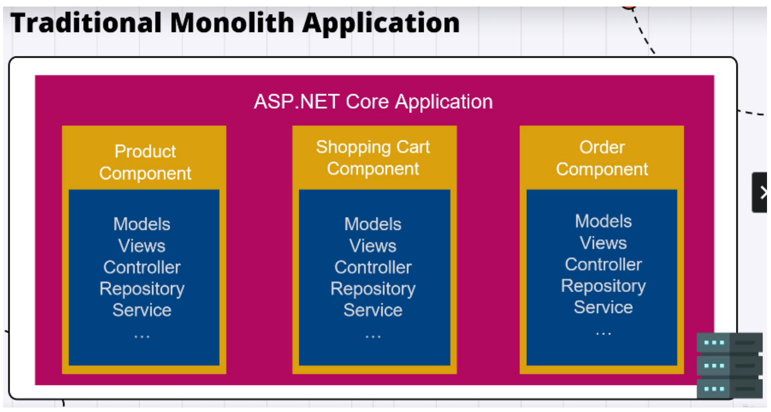


  - Microservices for Different Functionalities
    - Product management, shopping cart, order management, and email sending.
  - Handling Microservice Failures
    - Example: Email microservice failure and message storage in a message broker.
    - Other microservices continue to function even if one microservice goes down.
  - Comparison with Monolithic Applications
    - Monolithic applications go down entirely if the server fails.
    - Microservices architecture keeps the application running even if one microservice fails.

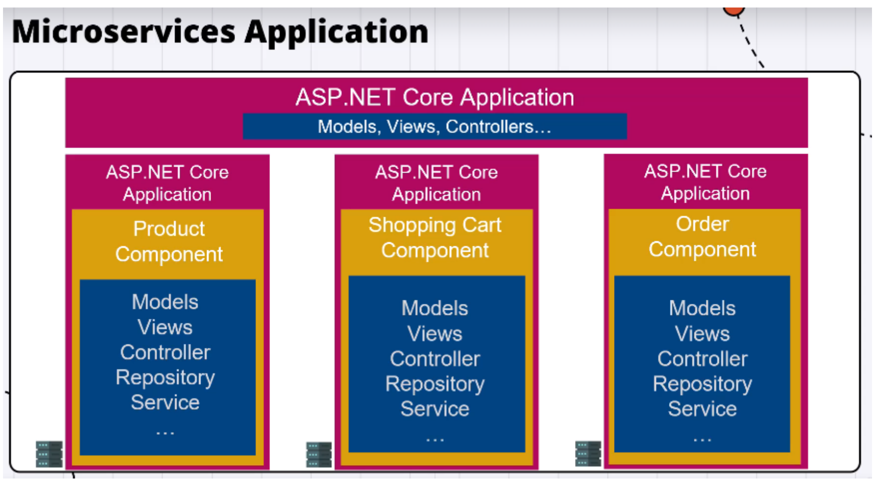
## 7. Microservices vs. Monolithic Architecture



- Monolithic Architecture


Traditional Monolith Application
ASP.NET Core Application
Product Component
Models Views Controller Repository Service …
Shopping Cart Component
Models Views Controller Repository Service …
Order Component
Models Views Controller Repository Service …

  - Centralized Data and Code
    - All data and code are centralized in one place.
    - Easier for developers to run and debug the complete application.
  - Advantages and Disadvantages
    - Works fine for smaller applications.
    - Challenges arise as the application grows, leading to tight dependencies and scaling issues.
- Microservices Architecture


Microservices Application
ASP.NET Core Application
Models, Views, Controllers…
ASP.NET Core Application
Product Component
Models Views Controller Repository Service …
ASP.NET Core Application
Shopping Cart Component
Models Views Controller Repository Service …
ASP.NET Core Application
Order Component
Models Views Controller Repository Service …

  - Loosely Coupled Services
    - Individual microservices with their own databases and technology stacks.
    - Communication between microservices using REST API calls or message brokers.
  - Advantages
    - Easier maintenance and scaling of individual services.
    - Flexibility in using different technologies and databases.
    - Efficient resource allocation and fault isolation.

- **8. Tools and Prerequisites**
  - Required Tools
    - Visual Studio and .NET 8
      - Visual Studio 2022 preview version for .NET 8.
      - Visual Studio 2022 with .NET 7 for those not using the preview version.
    - SQL Server
      - SQL Server and SQL Server Management Studio for database management.
    - Azure Subscription
      - Required for Azure Service Bus and other Azure services.
      - Minimal cost involved, with potential free credits for new Azure users.
  - Prerequisites
    - Basic Understanding of .NET Core
      - Familiarity with MVC programming and web application development.
    - Entity Framework Core
      - Basic understanding of CRUD functionalities using Entity Framework Core.
    - API Development
      - Basic understanding of building .NET APIs with Entity Framework Core.
    - Recommended Resources
      - Free course on Dotnet Mastery for fundamentals of API and Entity Framework Core.
      - Project resources and course content available on courses.net.com.
- **9. Clarifications and Misconceptions**
  - Docker and Microservices
    - Misconception
      - Microservices and Docker are not related.
      - Docker can be used with monolithic applications as well.
    - Course Focus
      - Focus on microservices architecture and API communication.
      - Avoiding additional complexity by not covering Docker.
  - Technology Stack

- Consistent Technology Stack
    - Using .NET APIs, Entity Framework Core, and SQL Server for all microservices.
    - Avoiding multiple technology stacks to prevent confusion and maintain focus on microservices concepts.
- Flexibility in Microservices
    - Microservices can use different programming languages and databases.
    - Once the basic concepts are understood, implementing microservices in different technologies is straightforward.

## 10. Course Prerequisites and Resources

- Basic Understanding of .NET Core
    - MVC Programming
        - Familiarity with MVC programming and web application development.
    - Entity Framework Core
        - Basic understanding of CRUD functionalities using Entity Framework Core.
    - API Development
        - Basic understanding of building .NET APIs with Entity Framework Core.
- Additional Resources
    - Free Course on Dotnet Mastery
        - Covers fundamentals of API and Entity Framework Core.
    - Project Resources
        - Available on courses.net.com.
        - Includes GitHub code, snippets, images, and other course content.

## 11. Conclusion

- Excitement and Learning Journey
    - Joining the Course
        - Excitement about learning microservices and building a real-world project.
    - Next Steps
        - Walkthrough of the application and microservices architecture in the next video.
        - Continuing the journey of understanding and implementing microservices.