Talha Rehman
FA22-BSE-159

1. Selection of Framework/Software

For this assignment, we have selected **ASP.NET Core** as the framework to analyze for its architectural evolution. ASP.NET Core is an open-source, cross-platform web framework developed by Microsoft for building modern web applications and APIs. It has undergone significant architectural changes from its initial release in 2016 to the latest stable versions, making it an ideal candidate for this assignment.

2. Contribution by Group Members

Talha Rehman (FA22-BSE-159) Creation of the architectural diagrams and comparison of the major releases.

Zoya Kayani (FA22-BSE-042):

Research on ASP.NET Core 1.0, including the architectural changes.

Saud ur Rehman (FA22-BSE-048)

Research on ASP.NET Core 3.0 and 5.0, focusing on performance improvements and new features.

Nimra Jadoon (FA22-BSE-011):

Research on ASP.NET Core 6.0 and 7.0, including cloud-native capabilities and minimal hosting model.

Detailed Evolution from First Release to Current

Overview:

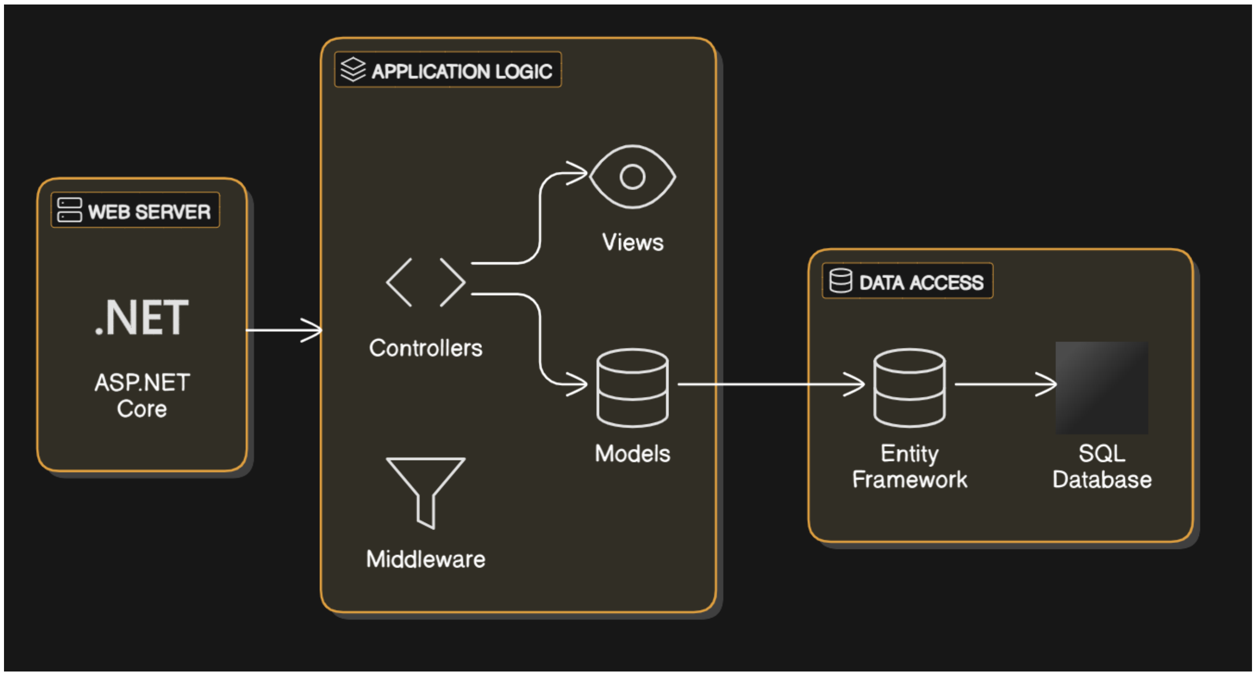
- **Release Numbers**: Each version's identifier (e.g., 1.0.0, 2.0.0, etc.).
- **Major Features**: Key functionalities and features introduced in each release.
- **Architectural Diagrams**: Visual representations of the system architecture at various stages of its evolution.
- **Release Notes Summary**: A detailed summary of the release notes, highlighting the most significant updates and changes.

Talha Rehman
FA22-BSE-159

1. First Release (Initial Version)

- **Release Number**: **1.0.0** (June 2016)
- **Major Features**:
  - **Cross-Platform Framework**: ASP.NET Core 1.0 was designed to be cross-platform, making it possible to run applications on **Windows**, **macOS**, and **Linux**. This was a major shift from the traditional **ASP.NET**, which was Windows-only.
  - **Lightweight and Modular**: The framework was lightweight and modular, allowing developers to include only the necessary components. This modularity improved performance and reduced the application's memory footprint.
  - **Middleware Architecture**: A key feature of ASP.NET Core 1.0 was the **middleware-based architecture**, which allowed for a flexible and customizable request pipeline. Each HTTP request passed through a series of middleware components (e.g., authentication, logging, etc.).
  - **Built-in Dependency Injection (DI)**: Unlike traditional ASP.NET, ASP.NET Core 1.0 integrated **dependency injection** as a core feature, enabling better management of object lifetimes and promoting loosely coupled designs.
  - **Unified Web API and MVC Framework**: ASP.NET Core 1.0 unified **MVC (Model-View-Controller)** and **Web API** into a single framework. This allowed developers to use a single set of components to handle both API requests and dynamic web pages.

Architectural Diagram:



- **Release Notes Summary**:
    - ASP.NET Core 1.0 laid the foundation for future releases by focusing on **cross-platform compatibility**, **modularity**, and **performance**. It introduced a new **middleware pipeline**, a more **flexible framework**, and **dependency injection** as a first-class citizen.
    - **Key Highlights**:
        - Cross-platform support.
        - Modular, lightweight framework.
        - Middleware-based request processing pipeline.
        - Built-in dependency injection.
        - Unified MVC and Web API framework.
    - **Reference**: ASP.NET Core 1.0 Release Notes.

---

2. Subsequent Major Releases

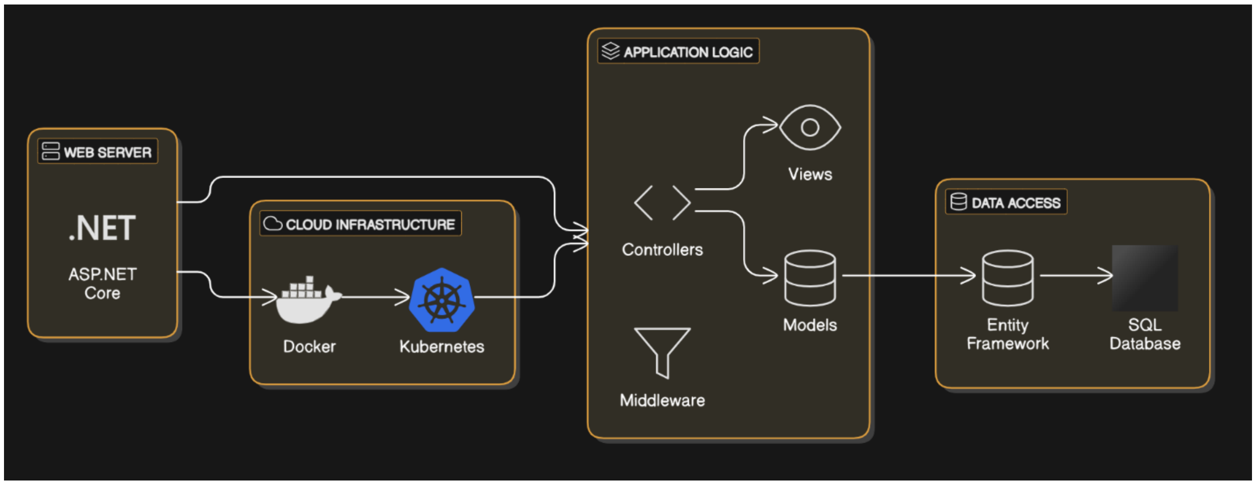
For each major release (e.g., 2.0.0, 3.0.0, etc.), provide the following:

ASP.NET Core 2.0

- **Release Number**: **2.0.0** (August 2017)
- **Major Features**:
    - **Razor Pages**: Introduced **Razor Pages**, a simpler alternative to MVC for building dynamic web pages. Razor Pages helped streamline scenarios where there was no need for full-fledged MVC functionality.
    - **Cross-Platform Performance Improvements**: ASP.NET Core 2.0 continued to improve performance, particularly for **Linux** and **macOS** environments. The framework was now more stable and efficient across all supported platforms.
    - **SignalR**: SignalR, for building real-time web applications, was integrated into ASP.NET Core 2.0. It enabled features like live notifications, instant messaging, and real-time dashboards.
    - **ASP.NET Core Identity**: ASP.NET Core 2.0 introduced **ASP.NET Core Identity**, a set of APIs for handling user authentication and authorization, user registration, and roles.
    - **Improved Dependency Injection**: Enhancements to the built-in DI system made it easier to configure and manage services.

Talha Rehman
FA22-BSE-159

Architectural Diagram:

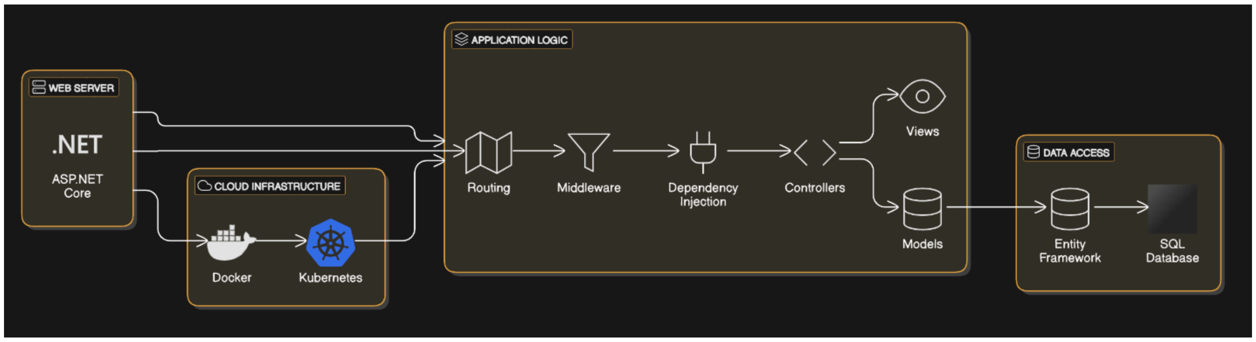


- **Release Notes Summary**:
  - ASP.NET Core 2.0 improved the framework's capability for building real-time applications with **SignalR** and simplified the development of web pages with **Razor Pages**. It also enhanced **cross-platform support**, making it a more stable and performant framework.
  - **Key Highlights**:
    - **Razor Pages** for simplified page-based development.
    - **SignalR** for real-time web functionality.
    - Improved **cross-platform** performance.
    - **ASP.NET Core Identity** for simplified authentication and authorization.
  - **Reference**: ASP.NET Core 2.0 Release Notes.

ASP.NET Core 3.0

- **Release Number**: **3.0.0** (September 2019)
- **Major Features**:
  - **Full Transition to .NET Core**: ASP.NET Core 3.0 marked the **end of support for the .NET Framework** and fully embraced **.NET Core** as the only platform for building web applications.
  - **Blazor (Preview)**: The **Blazor** framework, which allows developers to build interactive web UIs with **C#** instead of JavaScript, was introduced as a preview feature.
  - **Endpoint Routing**: The new **endpoint routing** system unified routing for MVC, Razor Pages, and Web API, making it easier to configure and manage routes.
  - **Worker Services**: Introduced **Worker Services**, a new feature for running long-running background tasks outside of the web request pipeline.

    - **Improved Performance**: ASP.NET Core 3.0 continued to improve performance with optimizations for **memory usage**, **HTTP request handling**, and **routing**.
- **Architectural Diagram**:



- **Release Notes Summary**:
    - ASP.NET Core 3.0 introduced **Blazor** for building client-side applications using C#, and improved its routing system with **endpoint routing**. The release also focused on performance enhancements and the continued transition to **.NET Core**.
    - **Key Highlights**:
        - **Blazor (Preview)**: C#-based client-side web UI framework.
        - **Full transition to .NET Core**.
        - **Endpoint Routing**: Unified routing for all types of requests.
        - **Worker Services** for background tasks.
    - **Reference**: ASP.NET Core 3.0 Release Notes.

---

3. Latest Release (Current Version)

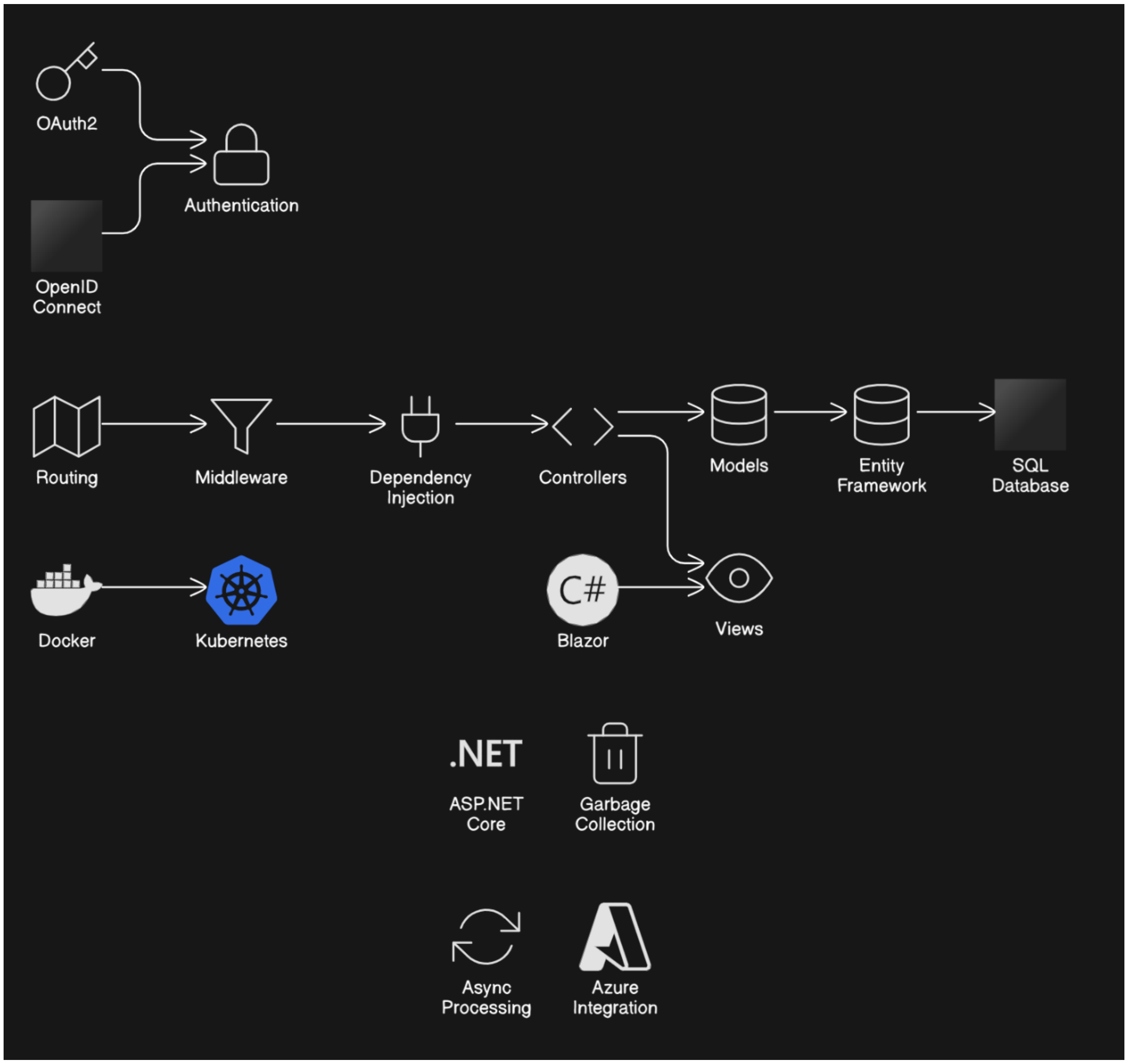
ASP.NET Core 5.0

- **Release Number**: **5.0.0** (November 2020)
- **Major Features**:
    - **Blazor Full Release**: Blazor was fully released, enabling developers to build interactive client-side web UIs using C# rather than JavaScript.
    - **Minimal APIs**: A new feature called **Minimal APIs** was introduced, which simplified the creation of lightweight, high-performance RESTful APIs with minimal code.

- **Performance Enhancements**: Continued focus on **performance** improvements, particularly in **HTTP/2** handling, **garbage collection**, and **asynchronous processing**.
- **Cloud-Native Development**: ASP.NET Core 5.0 enhanced cloud-native development capabilities with better integration with **Azure** and support for **containerized applications**.
- **Security Improvements**: New security features,

such as built-in support for **OAuth2** and **OpenID Connect**, were included to improve authentication and authorization workflows.

Architectural Diagram:



Talha Rehman
FA22-BSE-159

- **Release Notes Summary**:
  - ASP.NET Core 5.0 focused on improving performance and **cloud-native** development, with new features like **Blazor** and **Minimal APIs**. The release also introduced new security improvements and enhanced support for **containerized applications**.
  - **Key Highlights**:
    - Full release of **Blazor**.
    - Introduction of **Minimal APIs** for lightweight APIs.
    - Significant **performance** improvements.
    - **Cloud-native** features and better **Docker** and **Kubernetes** support.
  - **Reference**: ASP.NET Core 5.0 Release Notes.
