.NET / BLAZOR – GROUP G

INTERNET TECHNOLOGIES – COMP10020

RAUL GAL (B00310867)

SHANE CUMMINGS (B01740631)

COREY BLACK (B01651145)

ABDULLAH BARKAJI (B01651045)

INTRODUCTION TO .NET

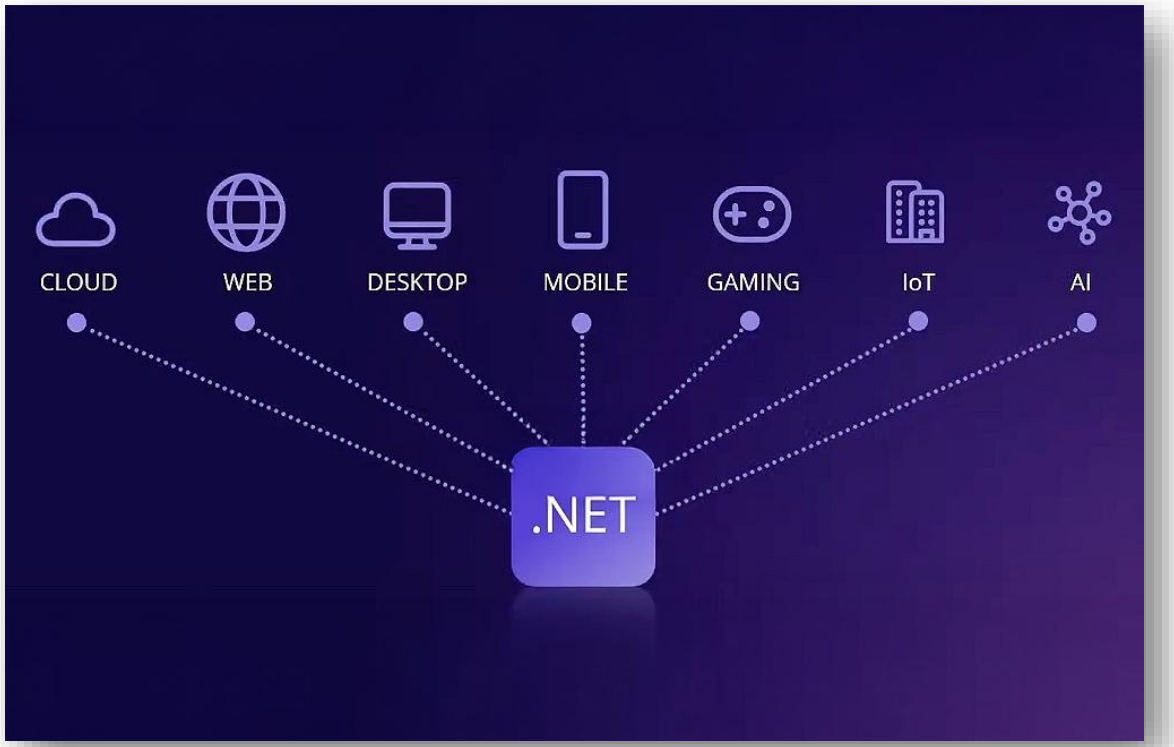
A CROSS-PLATFORM, OPEN-SOURCE ECOSYSTEM FOR MODERN APPLICATION DEVELOPMENT

What is .NET?

- A cross-platform, open-source framework by Microsoft
- Used to build web, desktop, mobile and cloud applications
- Supports multiple languages: C#, F#, and VB.NET
- Offers unified runtime and libraries for consistent performance

WHY .NET MATTERS

A UNIFIED DEVELOPMENT PLATFORM ACROSS DEVICES



Why .NET Matters?

.NET supports development across devices, platforms, and environments.

- **Cross-platform**
- **Unified framework**
- **High performance**
- **Security**
- **Community + Ecosystem**

WHAT IS BLAZOR?

A MODERN WEB UI FRAMEWORK POWERED BY C# AND .NET

Blazor lets you build interactive web apps in C# without writing JavaScript – running either in the browser or on the server.

- 💻 Write components using C#, not JavaScript
- 🧩 Built on .NET for rich interactive UIs
- 🌐 Runs in browser (WASM) or server (SignalR)

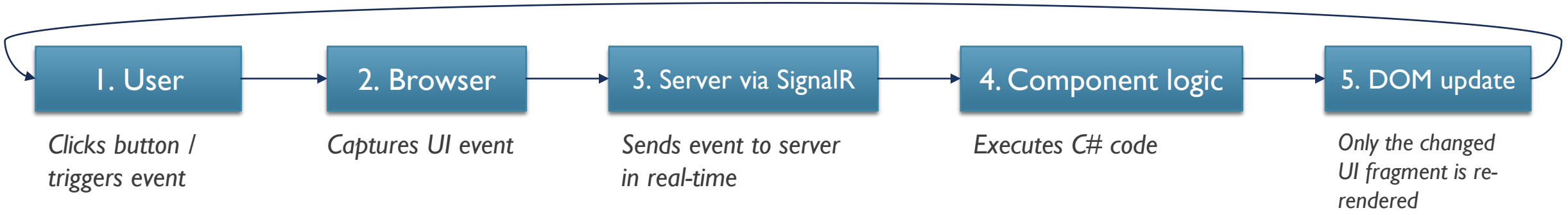
Counter.razor

```
1   <h3>Hello @name!</h3>
2
3   @code {
        1 reference
4       string name = "World";
5   }
6
```

Blazor sits between the browser UI and the .NET runtime, allowing C# to power the frontend.

BLAZOR'S ARCHITECTURE

BLAZOR PROCESSES UI EVENTS IN C# AND RETURNS DOM UPDATES THROUGH THE RUNTIME



- 🧩 UI Built from reusable C# components
- 🧠 Razor engine renders HTML
- ⚡ Events trigger C# logic
- 🔄 DOM updates re-render efficiently

Blazor differs from traditional MVC because it keeps a live connection to the server and only re-renders the exact UI fragment that changed.

🟣 Blazor Server model (SignalR over WebSockets)

Microsoft Docs (2024). *Blazor hosting models*. Available at: https://learn.Microsoft.com/aspnet/core/blazor

BLAZOR HOSTING MODELS

WEBASSEMBLY VS BLAZOR SERVER

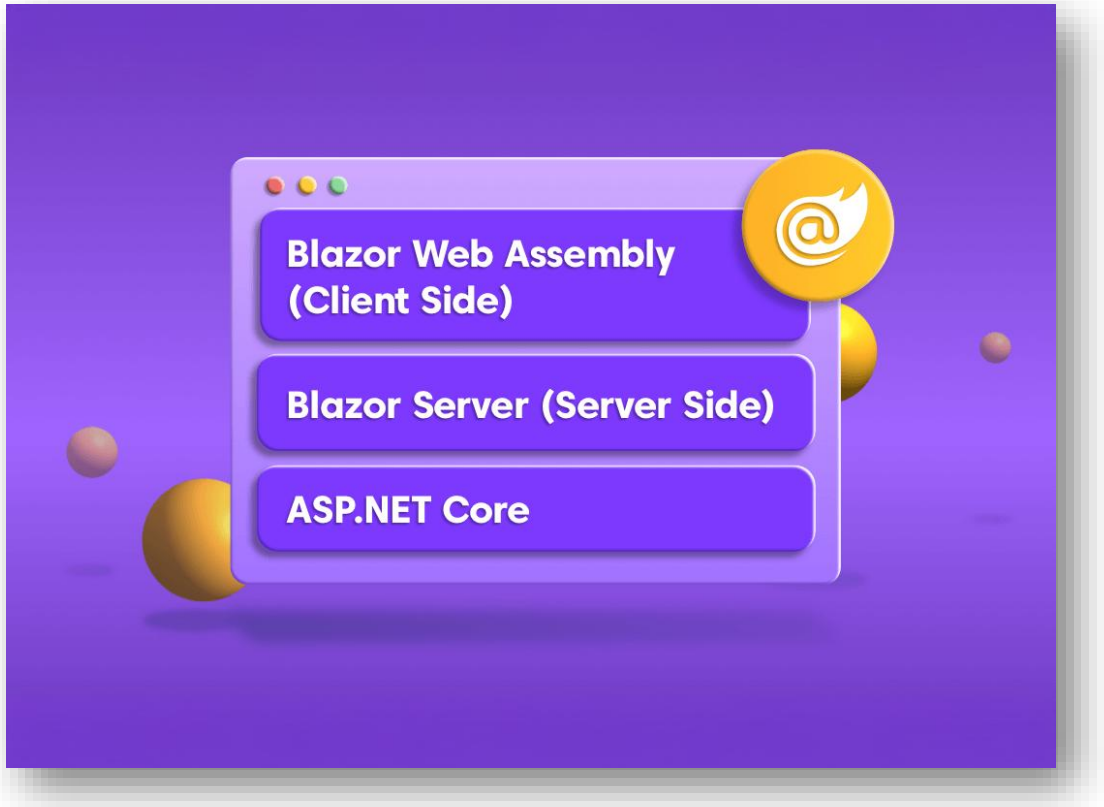
🌐 WebAssembly

- Runs client-side in browser
- Best for PWAs / offline/ heavy client logic

🔒 Blazor Server

- Runs on server via SignalR
- Best for secure internal dashboards/ real-time apps

Demo tie-in: We will later compare both models using the same component.



BLAZOR COMPONENTS

REUSABLE UI BUILDING BLOCKS IN BLAZOR

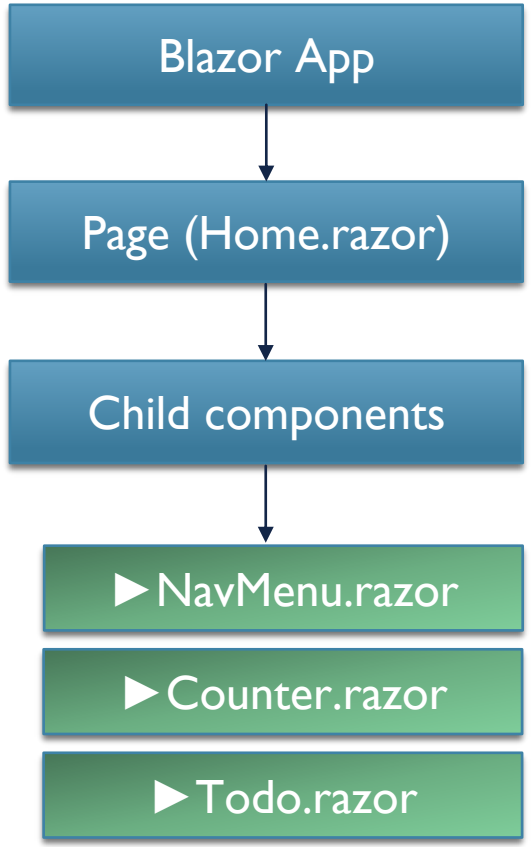
- 🧩 Self-contained building blocks
- 🎛️ Consist of UI + logic + styling
- ♻️ Encourages code reusability
- 📦 Examples: NavMenu.razor, Counter.razor, Todo.razor

Why they matter:

- Components are the core building blocks of a Blazor app, allowing modular, reusable UI across pages.

Demo link:

- In our tutorial, we'll build a simple custom component to show how markup + logic live together in .razor files.



Microsoft Learn (2024). *Razor Components in Blazor.* *Available at:* https://learn.microsoft.com/en-us/aspnet/core/blazor/components/?view=aspnetcore-9.0

DEMO PREVIEW

OF OUR .NET / BLAZOR APPLICATION

🧩 **What we will build:**

-A simple counter / form component

⚙️ **What it demonstrates:**

-Hot reload + inline C# event handling

-State management in a component

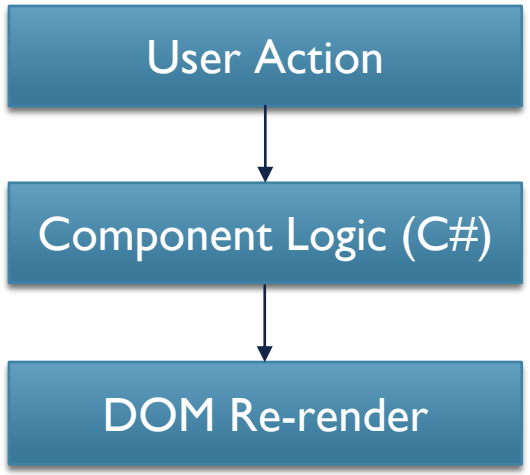
-UI re-renders on trigger

🌐 **Hosting model used:**

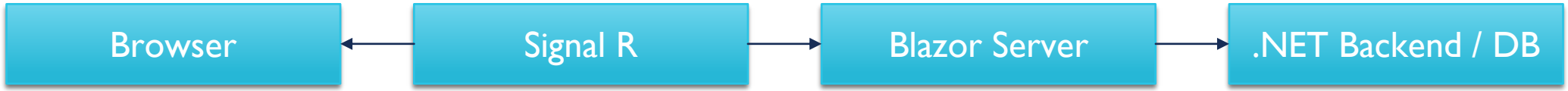
-Blazor server (real-time updates via SignalR)

📌 **Why it matters:**

-This is the core pattern used in dashboards, forms, and interactive UIs.



Blazor Server (live connection)



- 📡 UI events sent to server
- 💾 State lives on server
- ⚡ Fast initial load

Blazor WebAssembly (client-side runtime)

- ✅ Runs offline after load
- 🧠 **Client** executes C# in browser

INTEGRATION WITH .NET BACKEND

LIVE CONNECTION / VS CLIENT-SIDE RUNTIME

Microsoft .NET

REAL-WORLD USE CASES

BLAZOR POWERS MODERN, DATA-DRIVEN INTERNAL AND ENTERPRISE APPLICATIONS

Microsoft Customer Stories (2023). Blazor for enterprise-grade internal dashboards.
Available at: https://customers.microsoft.com/en-us/story

Blazor is a widely used framework, where secure, data-intensive, business-grade interfaces are required.

Common enterprise scenarios:

- 🏢 Internal admin dashboards – streamline internal workflows
- 🏥 Healthcare & finance portals – secure data-heavy interfaces
- 📡 Real-time dashboards – live IoT / metrics visualisation
- 🛠️ Enterprise intranet tooling – improves internal productivity

PROS & CONS OF BLAZOR

SERVER & WEBASSEMBLY

✅ General Pros

- 🧩 Full-stack C# (no context switching to JS)
- ⚙️ Strong tooling (Visual Studio, Hot Reload)
- ♻️ Component model = clean & reusable UI

✅ Server Advantages

- ⚡ Tiny initial load (great for thin clients)
- 🔌 Live connection for instant UI updates

✅ WASM Advantages

- 🛜 Runs entirely in the browser sandbox
- 📦 Can work offline after initial load

Blazor Server	Blazor WebAssembly
📡 Live connection	🛜 Offline capability
💾 Server-side state	🧠 Client-side state
⚡ Fast initial load	🚀 Fast runtime after load
🔒 Secure by default	🌍 Distributed by design

❗ Cons (grouped by type)

- 🌱 Ecosystem smaller than React/Vue (fewer 3rd party libraries)
- 📦 WASM: larger first load + limited browser-sandbox APIs
- 🌐 Server: persistent connection needed (SignalR)
- 🎨 Fewer ready-made UI kits vs JS world

EXAMPLE SCENARIO

WHERE BLAZOR FITS IN A REAL BUSINESS SYSTEM

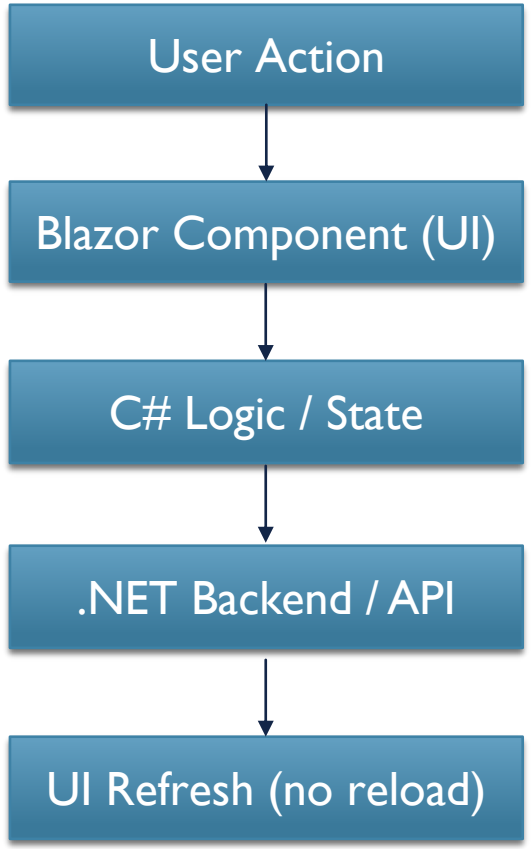
A company requires a secure internal employee dashboard that updates in real-time as staff activity changes.

Blazor is a strong fit because:

- Full-stack C# (backend + frontend)
- Live data-building to API results
- Secure authentication using .NET Identity
- Real-time UI updates via SignalR (Blazor Server)

Example feature:

A live status counter / KPI widget that updates without page reload



FUTURE OF BLAZOR

BLAZOR IS EVOLVING INTO A UNIFIED CROSS-PLATFORM UI FRAMEWORK IN .NET8+

Blazor's direction is focused on:

🚀 Unified development in .NET 8+ (one stack across web/desktop/mobile)

⚡ Faster runtime + Streaming SSR for near-instant interactivity

🏢 Deeper MAUI alignment > native desktop/mobile UI sharing

🏢 Continued enterprise adoption driven by tooling + security

Blazor Evolution Path

Today → **Unified UI Stack** → **Cross-platform Blazor apps**

.NET Blog (2024). Blazor in .NET 8: Unifying server and WebAssembly.
Available at: https://devblogs.microsoft.com/dotnet/blazor-dotnet-8

GITHUB PREVIEW

OF OUR .NET / BLAZOR APPLICATION

This tutorial will walk through:

1. Creating the Razor component structure
2. Binding UI to internal state
3. Handling user events (C# only)
4. Deploying a Blazor WebAssembly app to GitHub Pages

Repository link (GitHub):

https://kawrae.github.io/blazor-demo/

What you'll see in action:

👀 How UI reacts instantly without page reload

🔁 State changes triggering automatic re-render

🧩 Real-time event handling in pure C#

⚙️ Simple deployment workflow for Blazor WASM

SUMMARY / KEY TAKEAWAYS

What Blazor is good for / why it matters:

- ✅ Build full-stack apps using only C#
- ✅ Reusable component-based UI model
- ✅ Real-time updates via SignalR (Server)
- ✅ Can run offline in browser with WebAssembly
- ✅ Secure by design through .NET ecosystem

When to choose which model:

🟦 **Use Blazor Server when:**

- You need secure internal dashboards
- Real-time updates are essential
- Users are always online

🟩 **Use Blazor WebAssembly when:**

- Offline/PWA support is needed
- Workloads are client-heavy
- Deploying to public-facing apps

Blazor unifies frontend + backend with C#, enabling modern, data-driven apps without JavaScript – both online and offline.

REFERENCES

Official Documentation

1. Microsoft Docs (2024). *Blazor hosting models*.
https://learn.microsoft.com/aspnet/core/blazor/hosting

2. Microsoft Learn (2024). *Razor components in Blazor*.
https://learn.microsoft.com/aspnet/core/blazor/components

Use-case / Adoption

3. Microsoft Customer Stories (2023). *Blazor for enterprise-grade internal dashboards*.
https://customers.microsoft.com

Future Roadmap

4. .NET Blog (2024). *Blazor in .NET 8: Unifying server and WebAssembly*.
https://devblogs.microsoft.com/dotnet/blazor-dotnet-8

QUESTIONS AND DISCUSSION

- Want to see Server vs WASM performance trade-offs?
- Curious how to secure the API (JWT / Identity)
- How would we paginate or search the list
- What changes if the API is external (CORS, base address)

THANK YOU

GROUP G –

RAUL GAL (*B00310867*)

SHANE CUMMINGS (*B01740631*)

COREY BLACK (*B01651145*)

ABDULLAH BARKAJI (*B01651045*)