Comparative Analysis: Razor Pages vs ASP.NET Core Web API

1. Introduction:

In developing modern web applications, especially platforms like Profoliofy, selecting the appropriate technology stack directly influences scalability, maintainability, performance, and developer productivity. This document presents a comparison between two popular approaches within the Microsoft ecosystem:

- Razor Pages (ASP.NET Core)
- ASP.NET Core Web API + Frontend (e.g., React + Tailwind CSS)

2. Architecture Overview:

Stack	Architecture Description
Razor Pages	Monolithic web app where backend logic and frontend UI are served together using .cshtml views.
ASP.NET Core Web API	Backend-only RESTful API that communicates with a decoupled frontend (React or others) via HTTP/JSON.

3. Comparison Table:

This section compares Razor Pages and ASP.NET Core Web API + React stack across various dimensions.

Feature	Razor Pages	ASP.NET Core Web API + React
Development Speed	☑ Faster for small to medium CRUD apps	❌ Slower initial setup due to decoupled layers
Separation of Concerns	❌ Tight coupling of UI and logic	☑ Clean separation; backend/frontend independent
Frontend Power	❌ Limited to HTML/CSS + minimal JS	☑ Full modern UI with React/Vue capabilities
Learning Curve	☑ Easier for .NET developers	❌ Requires strong JS + React understanding
API Reusability	❌ Limited	☑ APIs reusable by mobile apps, admin panels, etc.
Performance	☑ Faster for server-rendered pages	☑ Better performance for dynamic SPAs
SEO	☑ Excellent due to SSR	❌ Needs extra setup (e.g., SSR with Next.js)
Maintainability	❌ Harder as code grows	☑ Easier modular management
Community	☑ Full Microsoft support	☑ Larger JS ecosystem for UI
Testing	❌ Difficult to test UI logic	☑ Separate unit/integration tests

4. Real-World Use Cases

Examples of companies and products using each stack.

- Razor Pages in Production:
  - - Microsoft Identity UI: Login/Register pages
  - - Internal Admin Panels: CRUD + dashboards for small orgs
- ASP.NET Core Web API + React in Production:
  - - Stack Overflow Teams: Scalable Q&A systems
  - - Visual Studio Code Spaces: Developer environment management
  - - Ideal for Profoliofy: UI-driven, dynamic, and scalable portfolio builder

5. Pros and Cons Summary

☑ Razor Pages Pros

- - Rapid development and fewer moving parts.
  - Best suited for form-based, CRUD-heavy apps.
  - Easier hosting and deployment.
  - Integrated with .NET Identity.

❌ Razor Pages Cons

- - Poor separation of concerns.
  - Limited frontend capabilities.
  - Difficult to reuse APIs.
  - Not ideal for large teams.

☑ ASP.NET Core Web API + React Pros

- - Clean separation of concerns.
  - Highly scalable and team-friendly.
  - Reusable APIs across platforms.
  - Powerful UI capabilities.

❌ ASP.NET Core Web API + React Minimal Trade-offs

- - Slightly longer setup time.
  - Requires frontend development knowledge (React).

6. Final Verdict & Recommendation

For a product like Profoliofy, ASP.NET Core Web API with a frontend framework like React is the superior choice. Razor Pages offers simplicity but lacks long-term scalability, flexibility, and separation of roles. Choosing the API-centric architecture enables modular development, powerful UI customization, and future-proofing for mobile and AI integration.

7. Summary Statement

While Razor Pages can serve as a quick start for tightly scoped applications, ASP.NET Core Web API with a decoupled frontend stack aligns with modern development best practices. It provides superior long-term value for startups aiming for scalability, modularity, and cross-platform accessibility.

8. AI Integration Capability

Integrating Artificial Intelligence into modern web platforms has become a competitive necessity. ASP.NET Core Web API offers significantly better flexibility for AI-based modules and services compared to Razor Pages.

- Why ASP.NET Core Web API is better for AI Integration:
    - - Clean API endpoints make it easy to integrate with Python-based AI services (e.g., via FastAPI, Flask, or Azure ML).
    - - Backend can communicate asynchronously with ML models hosted externally or internally.
    - - Easy to plug in AI features like resume parsing, auto-template recommendations, SEO scoring, and user behavior analytics.
- Why Razor Pages falls short:
    - - Mixing AI logic within the monolithic structure adds unnecessary coupling and complexity.
    - - Not ideal for handling asynchronous or large-scale data-driven tasks.