Languages & Frameworks Research

	Python (Flask)	JavaScript - Node.js (Express)	Laravel (PHP)	.NET (C#)
Ease of use	Easiest & Fastest	Still easy, but requires a bit more setup	Medium difficulty	More complex, slower to set up
Setup	Very easy, requires only Flask & requests	Minimal setup, but needs Multer/Axios	Easy if familiar with PHP, requires Guzzle	Best for enterprise, but requires setup
API Handling	Simple API calls with `requests` module	Asynchronous API calls, good for real-time data	API calls via Guzzle, good documentation	Strongly typed API handling, great security
Best Use Case	Best for AI, ML, quick prototypes	Best for full-stack JS apps (React, Vue, Angular)	Best for PHP-based apps & web platforms	Best for Microsoft ecosystem & large apps
Performance	Fast for small apps	Very fast for web apps	Moderate speed, heavier than Flask/Node	High performance for complex systems
Scalability	Scalable but not the best for high traffic	Highly scalable (event-driven)	Scalable, but slower than Node.js	Highly scalable for enterprise
Cons (Setup)	Requires Python setup	Needs middleware for file handling	More dependencies, needs PHP & Composer	Requires .NET runtime & setup
Cons (Performance)	Not great for large apps	Memory-heavy compared to Flask	Not as fast as Flask or Node.js	Slower development than Flask/Node
Cons (Other Issues)	Not full-stack, requires frontend framework (Flask is backend, so you'd need to pair with a frontend framework like React.js, Vue.js, or Angular for a modern web app, or Jinja templates (Flask's default HTML templates) for a simple, server-rendered UI.	Debugging async code can be tricky	Requires Blade or frontend integration	Harder for beginners, steep learning curve