Anthony Arroyo

Tampa, Florida USA| Tonyarroyo575@gmail.com | aarroyo.info |518-231-9305

Software Systems Engineer

Visual Studio | SQL | Azure Foundry

Skills and Certifications

Architectures	*Azure AI Engineering Certification*	*C# Certification*	*Software Development*
Object Oriented Design/UML	*Debugging*	*Web Services/APIs*	*Testing*

Technology

- C# 12, HTML5, SSMS, RESTful web services, Agile methodology, Architectures, Software Development Lifecycle, Management, SOA, OOD, DDD, Design Patterns, Model Context Protocol (MCP), Machine Learning, Version Control, Swagger API, Documentation, Dependency Injection, Project Management, Client Management, Git, Powershell, Azure, Mac, Windows

Core Compentencies

- Systems-first thinking with emphasis on long-term maintainability and evolution
- Translating ambiguous business problems into concrete domain models and workflows
- Designing backend systems around rules, state, and lifecycle management
- Building headless, API-driven platforms to support multiple future clients or interfaces
- Structuring applications for extensibility without premature complexity
- Isolating business logic to reduce coupling and improve testability
- Designing software around real operational processes
- Incremental system design with clear boundaries and ownership
- Strong focus on correctness, data integrity, and predictable behavior
- Applying Model Context Protocol (MCP) to integrate AI-driven workflows

Experience

Gracie Tampa Network
-Martial Arts Academy Management Platform
Saved instructors 33% of quarterly administrative time by architecting and developing a custom academy operations platform that streamlined student promotion workflows and centralized attendance, scheduling, and membership data into a unified system. Eliminated manual spreadsheets and promotion guesswork by establishing a single source of truth for student progress and eligibility. Built using layered Clean Architecture, separating API, application services, domain logic, and data access to ensure scalability, maintainability, and long-term extensibility.

- Reduced SQL management overhead by managing SQL schema using Entity Framework Core (Code First).
- Minimized manual promotion errors by automating eligibility logic based on attendance, time-in-rank, and instructor criteria, enforcing validation and domain rules to ensure accurate rank advancement.
- Designed a headless, loosely coupled .NET API and MCP-based AI integration, allowing the system to expose structured domain context to AI agents while preserving boundary integrity, scalability, and UI independence.
- Applied Repository and Service patterns to decoupled business rules from persistence concerns.

Frequent Music Discord
- AudioQuiplash Application
Increased community membership by 27% while hired to design and build a custom interactive knowledge-driven game for a creator's community. The system combined structured note-taking, tagging, and search mechanics to power fast content retrieval, progression logic, and repayable challenges. Built with a strong emphasis on efficient querying, data consistency, and long-term maintainability using clean architecture principles.
Responsibilities and technical focus included:

- Designed application states to manage idle, armed, triggered, and playback phases
- Implemented event-driven logic to handle randomized audio triggers and conditional execution
- Structured the application for extensibility, enabling new audio rules, triggers, or behaviors to be added without modifying core logic
- Focused on clear separation between control logic and audio execution, improving maintainability and testability

No Name Cannabis Company
- Notes & Documentation Management Application
Built a personal knowledge management system that improved information retrieval speed and reduced cognitive overhead through structured note-taking, tagging, and search. Designed for efficient querying, data consistency, and long-term maintainability using clean architecture principles.
Responsibilities and technical focus included:

- Designed a flexible note and tag domain model supporting many-to-many relationships
- Implemented search and filtering logic using LINQ and indexed queries
- Applied separation of concerns between controllers, services, and repositories
- Designed API endpoints optimized for incremental feature expansion
- Focused on system consistency and data integrity rather than UI-driven logic
- Structured the project to allow future enhancements such as full-text search and versioning