Key Learnings from Building AuthApi (Clean Architecture Authentication API)

1. Clean Architecture Enables Long-Term Maintainability

Separating the system into Domain, Application, Infrastructure, and API layers proved essential for scalability and maintainability. Each layer has a clear responsibility, reducing coupling and improving testability. The Domain layer remains independent of external dependencies, ensuring business logic is protected from framework changes.

2. Provider Abstraction Reduces Vendor Lock-In

Implementing the IAuthProvider interface allowed dynamic switching between Supabase and a Local authentication provider. This design increases resilience and protects against third-party service outages. It reinforces the importance of abstraction in cloud-native systems.

3. Security Must Be Multi-Layered

Security was implemented through JWT validation, API key protection for admin endpoints, rate limiting, secure password hashing (BCrypt), refresh token rotation, and audit logging. This demonstrates that authentication security is not a single feature but a collection of layered safeguards working together.

4. Rate Limiting is Essential for Public APIs

Exposing authentication endpoints publicly requires defensive design. Rate limiting prevents brute force attacks and service abuse while maintaining availability for legitimate users. Public APIs must anticipate misuse and design accordingly.

5. Health Monitoring Improves Operational Reliability

The inclusion of /health and /health/dependencies endpoints improves observability. Monitoring database and external auth provider availability supports automated scaling and rapid incident response.

6. Deployment Should Be Infrastructure-Aware

Deploying on Fly.io with Docker and environment-based configuration reinforced the importance of containerization and environment isolation. Secure secret management is critical for production systems.

7. Engineering Maturity Comes from Thinking Beyond Code

Beyond implementing endpoints, the project required consideration of security trade-offs, operational resilience, scalability, logging, and fallback strategies. Building AuthApi highlighted that production systems require architectural foresight, not just functional correctness.