- Low-Level System Design Roadmap with C# and .NET
  - Fundamentals of System Design
    - Understanding Functional & Non-Functional Requirements
    - SOLID Principles (SRP, OCP, LSP, ISP, DIP)
    - Design Patterns (Singleton, Factory, Observer, etc.)
    - Trade-offs in Design (Performance vs Maintainability)
  - Core Programming Concepts
    - Data Structures & Algorithms in C#
      - Arrays, Lists, Stacks, Queues, HashTables
      - Trees, Graphs, Heaps, Tries
      - Sorting & Searching Algorithms
      - Time Complexity Analysis (Big-O Notation)
    - Memory Management & Garbage Collection
    - Threading & Parallel Programming
      - Task Parallel Library (TPL)
      - Async/Await Deep Dive
      - Thread Safety & Synchronization
  - Object-Oriented System Design (OOSD)
    - UML Diagrams (Class, Sequence, Component)
    - Layered Architecture in .NET
    - Domain-Driven Design (DDD)
      - Entities & Value Objects
      - Aggregates & Repositories
      - Domain Events & Services
  - Designing Scalable Systems
    - Caching Strategies (In-Memory, Redis)
    - Load Balancing & Rate Limiting
    - API Design Principles (REST, gRPC, GraphQL)
    - CQRS & Event Sourcing
  - High-Performance Coding in .NET
    - Span & Memory for Efficient Memory Usage
    - StringBuilder vs String Concatenation
    - Value Types vs Reference Types
    - Pooled Object Usage for Optimized Memory
  - Data Storage & Persistence
    - Database Indexing & Query Optimization
    - Entity Framework Core Internals
    - NoSQL (MongoDB, Redis) vs SQL (PostgreSQL, SQL Server)
  - Networking & Communication
    - TCP/UDP Socket Programming in C#

| ├── 🟢 gRPC vs REST vs WebSockets
| ├── 🟢 Message Queues (RabbitMQ, Kafka, Azure Service Bus)

├── 🔵 Security & Authentication

| ├── 🟢 OAuth2 & OpenID Connect
| ├── 🟢 Secure Data Storage (Encryption & Hashing)
| ├── 🟢 Common Attack Prevention (SQL Injection, CSRF, XSS)

├── 🔵 Cloud & DevOps for System Design

| ├── 🟢 CI/CD Pipelines (GitHub Actions, Azure DevOps)
| ├── 🟢 Kubernetes & Docker for Microservices
| ├── 🟢 Serverless Architectures (Azure Functions, AWS Lambda)

├── 🔵 Debugging & Performance Monitoring

| ├── 🟢 Profiling with dotTrace & PerfView
| ├── 🟢 Logging & Telemetry (Serilog, Application Insights)
| ├── 🟢 Benchmarking with BenchmarkDotNet

├── 🔵 Soft Skills & Code Quality

| ├── 🟢 Code Reviews & Clean Code Practices
| ├── 🟢 Writing Efficient Unit & Integration Tests
| ├── 🟢 Problem-Solving & System Thinking

Master these concepts to become a Low-Level System Design Expert in C# and .NET!

Follow for more deep dives into .NET & System Design
#CSharp #LowLevelDesign #DotNet #SystemDesign