

# **The Architect's Blueprint**

**Building Scalable and Maintainable Web Applications**

## About Me

- Matt Vaughn, Application & Solutions Architect
- [AngularArchitecture.com](https://angulararchitecture.com)
- GitHub: [buildmotion](https://github.com/buildmotion)
- X: [@AngularArch](https://twitter.com/AngularArch)
- Presentation: <https://bit.ly/monorepos-unleashed>

# Overview of the Session

- Designing an Architecture
- Tools and Automation
- Leveraging Frameworks
- Advanced Code Generation Techniques
- Applying SOLID Principles
- Packages and Frameworks Overview

Plan?

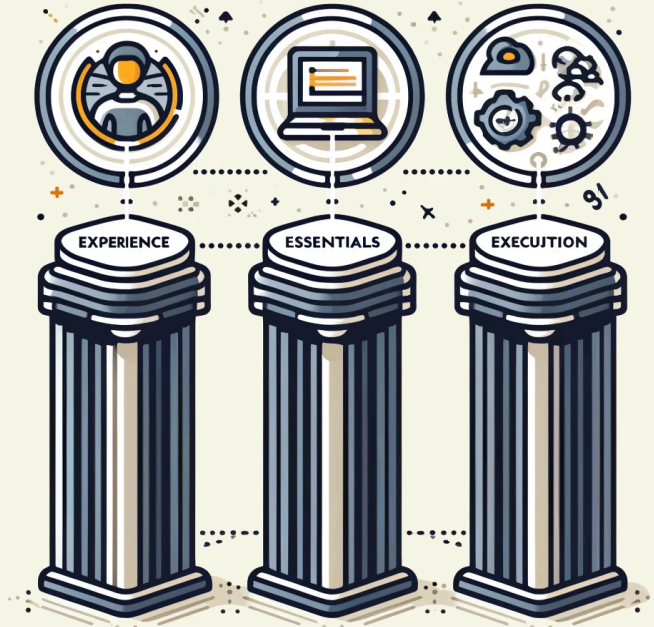


# Plan



"The plan is to go in there and win; everybody has a plan until they get hit. Then, like a rat, they stop in fear and freeze."

# The Elements of Architecture



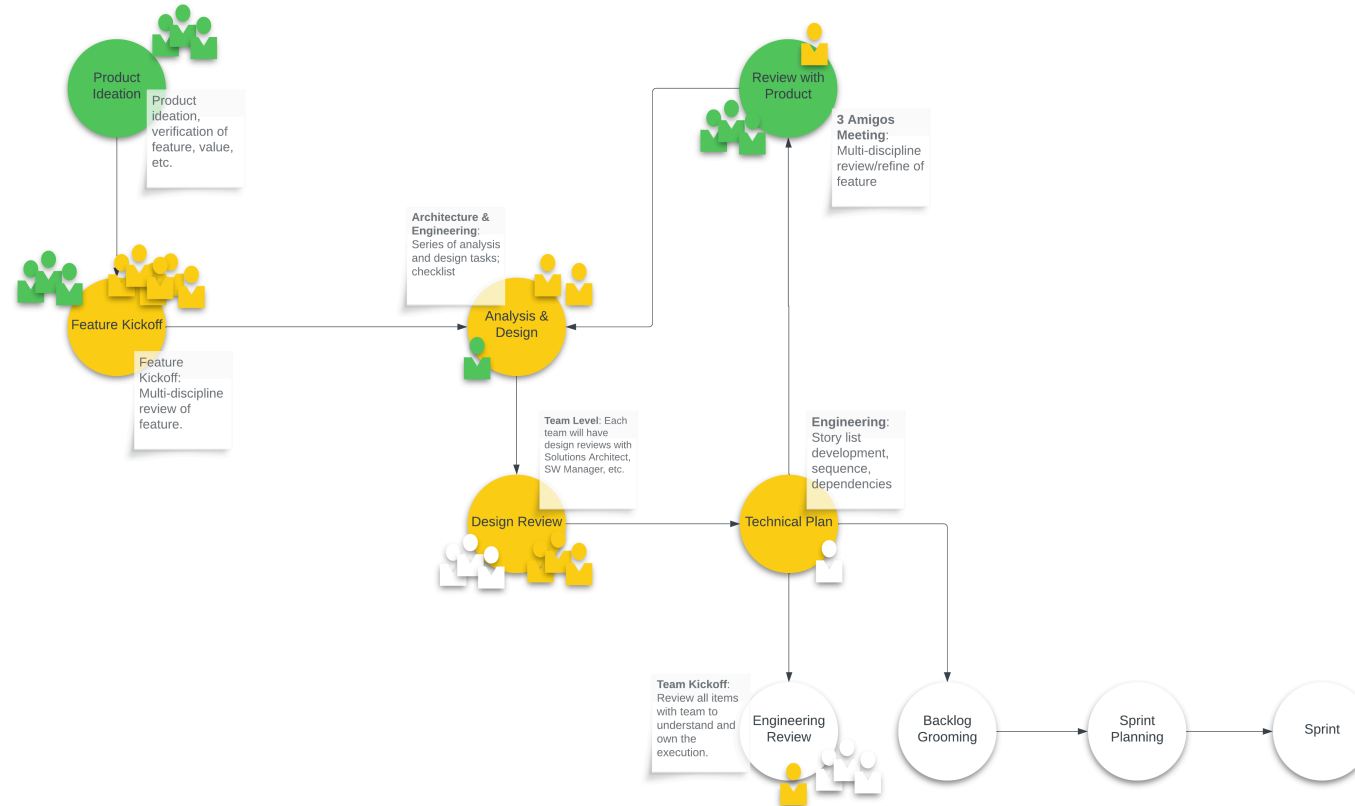
1. Experience
2. Essentials
3. Execution

# Experience



- Specific knowledge of business, domain, and customers
- Defines the W's: what, who, when, where, and why?
- Involves architects, developers, DevOps, DBAs, Product, SMEs

# Experience :: Communication Workflow

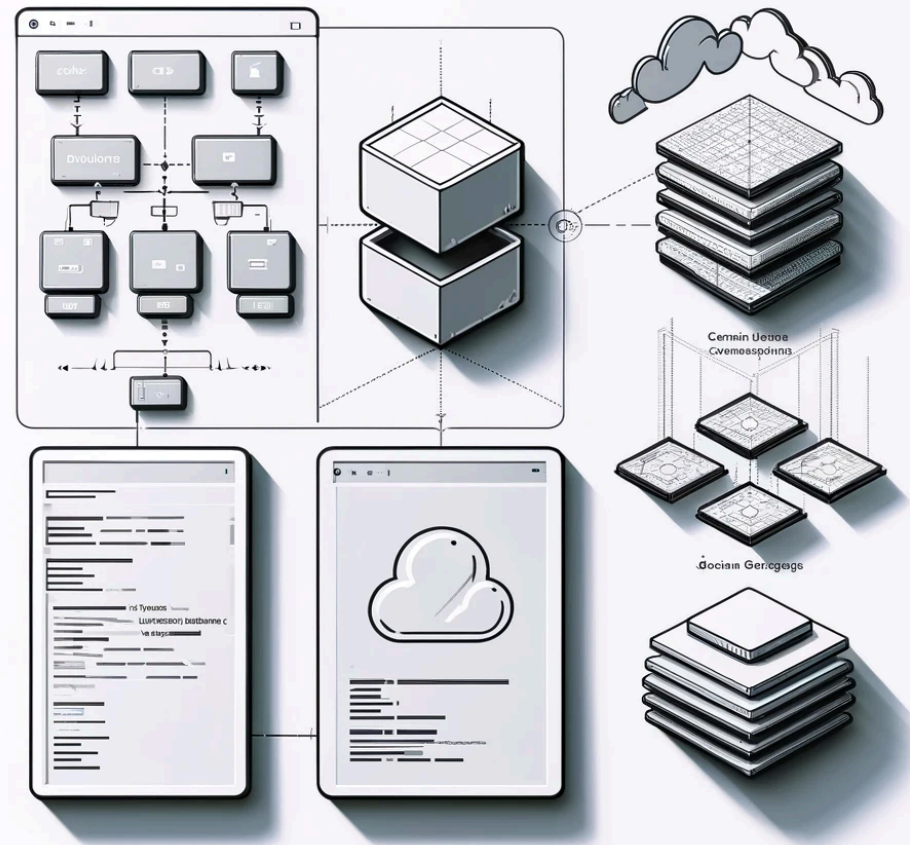




## Analysis and Design

- actors and personas (who)
- use cases and/or workflows
  - simple, detailed
- domain storytelling

# Design



## Design - How?

- C4, 4+1 Models
- UI mockup (low/high Fidelity)
- Domain modeling
  - Class/sequence diagrams
- Sequence; order of operation

# Essentials

- Tools and Materials
- Hosting Environments
- CI/CD Automation

## Execution

- Development workflow and developer experience
- Ensuring code quality and reliability
- Continuous evaluation and discipline

# Designing an Architecture

“Software architecture is the high-level structure of a software system, defining its parts, their interactions, and the guiding principles and patterns for design and evolution.”

- Definition and Principles
- Benefits
- Case Study
- Best Practices

# Definition and Principles of CLEAN Architecture

- Separation of Concerns
- Dependency Rule
- Entities, Use Cases, Interfaces, and Frameworks

## Separation of Concerns

- Features (Vertical/Horizontal):
  - UI/UX
  - business logic
  - Data access or API access
- How?: code organization...



## Dependency Rule

- IoC/dependency injection
- Providers and interfaces
- side effects
  - testing maintenance, extensibility, abstraction

## Benefits of CLEAN Architecture

- Enhanced Maintainability
- Scalability
- Improved Testability
- Flexibility

## Case Study: Implementation in a Real-World Project

- Initial State vs. Post-Implementation
- Key Changes and Their Impact

# Best Practices for Maintaining CLEAN Architecture

- Regular Refactoring
- Code Reviews, Frequent Commits/Pushes
- Documentation, training, playbook
- mentality/accountability

# Tools and Automation for Streamlining Development

- Importance of Automation
- Tools for Enforcing Coding Standards
- CI/CD
- Automated Testing Frameworks

# Importance of Automation in Development

- Consistency
- Efficiency
- Error Reduction

# Tools for Enforcing Coding Standards

- Soell Chickeng
- ESLint. Prettier
- Code Access

# Continuous Integration and Continuous Deployment (CI/CD)

- GitHub Actions
- Jenkins



# Automated Testing Frameworks

- Unit/specification
- End to end testing
- Integration testing

# Leveraging Frameworks for Robust Systems

- Front-End Frameworks
- Back-End Frameworks
- Seamless Communication
- UI Control Suites
- Logging and Monitoring Tools
- Performance and Analytical Tools
- Profiling Tools
- Quality and Reliability Strategy

# Front-End Frameworks: Angular, React, Vue.js

- Key Features and Use Cases
- Performance Optimization Techniques

## Back-End Frameworks: NestJS, Express.js

- Microservices Architecture
- API Design and Management

# Seamless Communication between Front-End and Back-End

- API response schema
- RESTful APIs vs. GraphQL

## Selection of UI Control Suites

- Kendo UI
- DevExtreme
- PrimeNG

## Logging and Monitoring Tools

- DataDog, New Relic
- Azure Application Insights

# Performance and Analytical Tools

- Lighthouse
- WebPageTest



# Profiling Tools

- Chrome DevTools
- Angular Profiler
- React Profiler

## Quality and Reliability Strategy

- Using Jest for Specification Tests
- Strategy for Ensuring Quality and Reliability

# Advanced Code Generation Techniques

- Introduction to Code Generation
- Tools for Code Generation: Nx, Yeoman
- Automating Boilerplate Code Production
- Examples and Demonstrations

# Introduction to Code Generation

- Benefits
- Common Use Cases

# Tools for Code Generation: Nx, Yeoman

- Key Features and Capabilities

# Automating Boilerplate Code Production

- Examples
- Best Practices

## Examples and Demonstrations

- Creating Custom Nx Plugins
- Generating Angular Components and Services

# Applying SOLID Principles in Front-End Development

- Overview of SOLID Principles
- Practical Examples in Angular
- Angular Dependency Injection and IoC
- Benefits of SOLID



# Overview of SOLID Principles

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

## Practical Examples in Angular

- Modular Architecture
- Service-Oriented Design

# Angular Dependency Injection

- Explanation of Dependency Injection (DI)
- How DI in Angular Relates to Inversion of Control (IoC)
- Practical Examples of DI in Angular

## Benefits of SOLID in Long-Term Maintenance and Scalability

- Improved Code Quality
- Easier Refactoring
- Enhanced Collaboration

# Packages and Frameworks Overview

- Front-End Frameworks
- Back-End Frameworks
- UI Control Suites
- Logging and Monitoring Tools
- Performance and Analytical Tools
- Profiling Tools

# Front-End Frameworks

## Angular

- Strengths: Strong typing, built-in RxJS support, CLI tools
- Alternatives: [React](#), [Vue.js](#)

## React

- Strengths: Flexibility, large ecosystem, JSX
- Alternatives: [Angular](#), [Svelte](#)

## Vue.js

- Strengths: Easy to learn, flexible, performant
- Alternatives: [React](#), [Svelte](#)

# Back-End Frameworks

## NestJS

- Strengths: Modular architecture, TypeScript, built-in support for microservices
- Alternatives: [Express.js](#), [Koa.js](#)

## Express.js

- Strengths: Minimalist, flexible, widely used
- Alternatives: [NestJS](#), [Hapi.js](#)

## Spring Boot (if Java is considered)

- Strengths: Convention over configuration, large ecosystem, integrated security
- Alternatives: [NestJS](#), [Django](#)

# UI Control Suites

## Kendo UI

- Strengths: Comprehensive, robust performance, extensive features
- Alternatives: [DevExtreme](#), [PrimeNG](#)

## DevExtreme

- Strengths: Wide range of controls, responsive design
- Alternatives: [Kendo UI](#), [Syncfusion](#)

## PrimeNG

- Strengths: Rich set of UI components, easy to integrate
- Alternatives: [Kendo UI](#), [DevExtreme](#)



# Logging and Monitoring Tools

## LogRocket

- Strengths: Session replay, performance monitoring
- Alternatives: [Sentry](#), [Datadog](#)

## Sentry

- Strengths: Error tracking, performance monitoring
- Alternatives: [LogRocket](#), [Rollbar](#)

## ELK Stack

- Strengths: Full-stack logging, search capabilities
- Alternatives: [Splunk](#), [Graylog](#)

# Performance and Analytical Tools

## Lighthouse

- Strengths: Performance audits, SEO checks
- Alternatives: [WebPageTest](#), [GTmetrix](#)

## New Relic

- Strengths: Full-stack monitoring, analytics
- Alternatives: [Datadog](#), [AppDynamics](#)

## Datadog

- Strengths: Cloud monitoring, metrics collection
- Alternatives: [New Relic](#), [Splunk](#)

# Profiling Tools

## Chrome DevTools

- Strengths: Built-in browser tool, real-time performance insights
- Alternatives: [Firefox Developer Tools](#), [Safari Web Inspector](#)

## Angular Profiler

- Strengths: Angular-specific performance insights
- Alternatives: [Chrome DevTools](#), [React Profiler](#)

## React Profiler

- Strengths: Visualizes component render times
- Alternatives: [Chrome DevTools](#), [Vue Devtools](#)

# Conclusion

- Recap of Key Points
- Q&A Session
- Final Thoughts and Takeaways

- C4 Models
  - [c4model.com](https://c4model.com)
  - [Visualizing software architecture with the C4 model](#) - Simon Brown, Agile on the Beach 2019
  - CLEAN Architecture: Uncle Bob's resources, case studies
  - Tools and Automation: Trends in CI/CD, large-scale project automation
  - Frameworks: Performance comparisons, real-world examples
  - UI Control Suites: Evaluations, in-depth analysis
  - Logging and Monitoring: Analysis of tools and strategies
  - Performance and Analytical Tools: In-depth reviews and comparisons
  - Profiling Tools: Optimizing performance
  - Code Generation: Tutorials on Nx, Yeoman