Systems Design

(Framework-agnostic)

ROJO, J/ALIWATE, D/EDO, A EDO/UY,
ULRICH/ODARVE, C

Framework-agnostic

- Not tied to any specific software framework

(e.g., not only Laravel, not only Django, not only Spring Boot)

A framework-agnostic lesson, design, or system:

- Works regardless of the framework used
- Focuses on principles and concepts, not tools
- Can be applied to any language or framework

WHAT IS SYSTEMS DESIGN?
(Traditional Definition)

- As a profession, therefore, engineering is concerned primarily with design-- the design of processes, structures, machines, circuits, and software
- --and with the purposeful combinations of these elements to result in what we call systems.

WHAT IS SYSTEMS DESIGN?

Traditional focus:

- What components do we design and how do we combine them?

Modern focus adds:

- For whom are we designing, how do they experience it, how does it evolve, and how do we improve it continuously?

Modern Software & Systems Design

Systems and software design is a **human-centered, iterative, and collaborative process** that transforms user needs and business goals into **reliable, scalable, and maintainable software systems**.

What Is Systems & Software Design?

In modern application development, this process is commonly supported by software frameworks, which provide standard structures, tools, and conventions to efficiently manage application logic, data, security, and user experience.

Software design integrates:

- Engineering principles
- User Experience (UX) design
- Data modeling and management
- Security and authentication
- Continuous feedback and improvement

Modern frameworks support this integration through ***architectural patterns, reusable components, built-in security mechanisms, and support for agile and iterative development***.

Role of Software Frameworks in Modern Software Design

- A **software framework** is a reusable platform that provides a **foundation for building applications**, allowing developers to focus on solving domain problems rather than low-level infrastructure concerns.

Key Contributions of Frameworks to Software Design

- Architectural structure (e.g., MVC, MVVM, layered architecture)
- Separation of concerns for maintainability
- Standardized data access and persistence
- User interface templating or presentation layers
- Built-in authentication and authorization
- Routing and request handling
- Support for APIs and distributed systems
- Integration with DevOps and CI/CD pipelines

Phase 1: Requirements Analysis

- User stories translated into application components and workflows
- Functional requirements mapped to features and modules
- Data requirements reflected in data models and schemas
- Early definition of validation and business rules

Phase 2: Conceptual & Architectural Design

- Architectural patterns define system structure
- Modular design through layers or services
- Clear separation between presentation, logic, and data
- API-first or service-oriented architecture when needed

Phase 3: Detailed Design & Data Modeling

- Database schema design and migrations
- Object–relational or data-mapping layers
- Clearly organized business logic
- Consistent UI or presentation templates

Phase 4: Analysis, Validation & Security

- Input validation and data sanitization
- Authentication and authorization mechanisms
- Protection against common vulnerabilities
- Centralized error handling and logging

Phase 5: Prioritization & Agile Implementation

- Incremental feature development
- Rapid prototyping using framework tools
- Reusable modules and components
- Agile sprint-based implementation

Phase 6: Testing & Deployment & DevOps

- Unit, integration, and system testing tools
- Test data generation and environment isolation
- Automated deployment pipelines
- Cloud and container deployment support

Phase 7: Maintenance & Improvement

Framework Support:

- Refactoring through clear architecture
- Logging, monitoring, and diagnostics
- Dependency and version management
- Continuous updates and enhancements

Key Takeaway

- Modern software design is an iterative, user-driven process.
- Software frameworks operationalize this process by providing structure, consistency, security, and scalability—enabling development teams to build high-quality systems efficiently without reinventing core system components.

Suggested Activities

Suggested Activities

Reference books

- Sage A.P. and Rouse, W.B. Handbook of Systems Engineering and management, John Wiley & Sons, .
- Kendall And Kendall, System Analysis and Design