Software Engineering Principles

Software Engineering is systematic approach to software development.

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC):

1. Requirement Analysis
   - Gather functional and non-functional requirements
   - SRS (Software Requirement Specification) document
   - Stakeholder interviews, surveys
2. Design
   - High-level design (Architecture)
   - Low-level design (Detailed modules)
   - UML diagrams: Use case, Class, Sequence
3. Implementation (Coding)
   - Writing actual code
   - Following coding standards
   - Version control (Git)
4. Testing
   - Unit testing: Individual modules
   - Integration testing: Combined modules
   - System testing: Complete system
   - Acceptance testing: User validation
5. Deployment
   - Release to production
   - Installation and configuration
   - User training
6. Maintenance
   - Bug fixes
   - Updates and enhancements
   - Performance optimization

SDLC MODELS:

1. Waterfall Model
   - Sequential phases
   - Easy to understand and manage
   - Rigid, no going back
   - Good for: Small, clear requirements
2. Agile Model
   - Iterative and incremental
   - Flexible, customer collaboration
   - Sprints (2-4 weeks)
   - Good for: Dynamic requirements
3. Spiral Model
   - Combines iterative and waterfall
   - Risk analysis in each iteration

  - Prototype development
  - Good for: Large, complex projects

4. V-Model (Verification & Validation)
  - Testing planned parallel to development
  - Each development phase has testing phase
  - Good for: Safety-critical systems

5. DevOps Model
  - Continuous Integration/Deployment (CI/CD)
  - Collaboration between Dev and Ops
  - Automation, monitoring
  - Good for: Rapid releases

TESTING TYPES:

1. Black Box Testing
  - Tests functionality without knowing code
  - Techniques: Equivalence partitioning, Boundary value

2. White Box Testing
  - Tests internal code structure
  - Techniques: Statement coverage, Path coverage

3. Gray Box Testing
  - Combination of black and white box
  - Partial knowledge of internals

4. Regression Testing
  - Ensures new code doesn't break existing features
  - Run after every change

5. Performance Testing
  - Load testing: Expected load
  - Stress testing: Beyond capacity
  - Scalability testing: Growth handling

SOFTWARE METRICS:

1. LOC (Lines of Code)
  - Measure size
  - Language dependent

2. Cyclomatic Complexity
  - Measures code complexity
  - Number of independent paths

3. Function Points
  - Measures functionality delivered
  - Language independent

4. Defect Density
  - Defects per KLOC (1000 lines)
  - Quality indicator

PROJECT MANAGEMENT:

1. Work Breakdown Structure (WBS)
  - Hierarchical decomposition of tasks

   - Helps in estimation
2. PERT/CPM Charts
   - Critical Path Method
   - Identifies longest path
   - Helps in scheduling
3. Gantt Charts
   - Visual timeline
   - Shows task dependencies
   - Easy to understand
4. Risk Management
   - Identify risks early
   - Mitigation strategies
   - Contingency planning

VERSION CONTROL:

Git Commands:
- git init: Initialize repository
- git clone: Copy repository
- git add: Stage changes
- git commit: Save changes
- git push: Upload to remote
- git pull: Download from remote
- git branch: Create branches
- git merge: Combine branches

DESIGN PRINCIPLES:

1. DRY (Don't Repeat Yourself)
   - Avoid code duplication
   - Use functions, classes
2. KISS (Keep It Simple, Stupid)
   - Simplicity is key
   - Avoid over-engineering
3. YAGNI (You Aren't Gonna Need It)
   - Don't add functionality until needed
   - Prevents bloat
4. SOLID Principles
   - Single Responsibility
   - Open/Closed
   - Liskov Substitution
   - Interface Segregation
   - Dependency Inversion