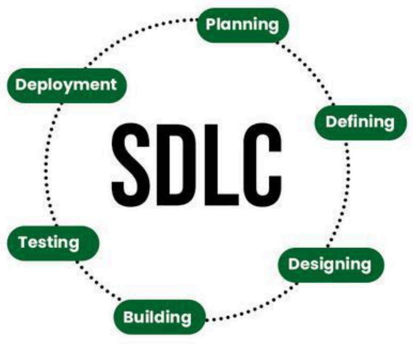
Software Development Life Cycle (SDLC)

1. Introduction



The Software Development Life Cycle (SDLC) is a structured process used to develop high-quality software. It defines phases that guide a project from initial idea to final deployment and maintenance. SDLC helps improve quality, planning, and project management by providing a clear framework.

---

2. Phases of SDLC

2.1 Requirement Analysis

This phase involves gathering detailed requirements from stakeholders, clients, and users. Analysts study the needs and document functional and non-functional requirements.

Activities:

- Identify user needs
- Analyze feasibility
- Prepare requirement specification documents

2.2 Planning

Planning outlines the scope, budget, deadlines, and resources. It sets the foundation for the whole project.

Activities:

- Cost estimation
- Scheduling
- Risk analysis

2.3 System Design

In this phase, the system architecture is created. Designers prepare diagrams, database structure, and UI/UX layout.

Activities:

- High-level design (HLD)
- Low-level design (LLD)
- Database design

2.4 Development (Coding)

This is where actual code is written based on design documents. Developers build modules and integrate them.

Activities:

- Writing code
- Version control
- Unit testing by developers

2.5 Testing

Testing ensures the software is error-free and meets requirements. Testers verify functionality, performance, and security.

Types of Testing:

- Functional testing
- Integration testing
- System testing
- User acceptance testing

2.6 Deployment

After testing, the software is delivered to users. Deployment may happen in stages depending on project type.

Activities:

- Releasing software
- Installation or cloud deployment
- Beta rollout (if required)

2.7 Maintenance

Once software is live, maintenance ensures it continues working smoothly. Teams fix bugs, update features, and improve performance.

Activities:

- Bug fixing
- Feature updates
- Performance enhancements

---

3. SDLC Models

Different models exist based on project needs.

3.1 Waterfall Model

A linear model where each phase must be completed before starting the next.

3.2 Agile Model

An iterative model with continuous development, testing, and user feedback.

3.3 Spiral Model

Focuses on risk analysis combined with iterative development.

3.4 V-Model

A development model where testing occurs in parallel with each development phase.

3.5 Iterative Model

Software is developed in repeated cycles, improving with each version.
