LECTURE 2:

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

What Is SDLC?
SDLC (Software Development Life Cycle) is a structured process used by software teams to plan, design, build, test, deploy, and maintain software systems.
It provides a clear, step-by-step framework that ensures the final product is high-quality, meets user needs, and is delivered on time.

SDLC provides framework for planning, designing, testing & deploying systems. It ensures that software is **high-quality**, **cost-effective**, and **delivered on time**

SDLC is a “roadmap” for software creation. Every serious software company uses a form of SDLC.

Objectives of SDLC

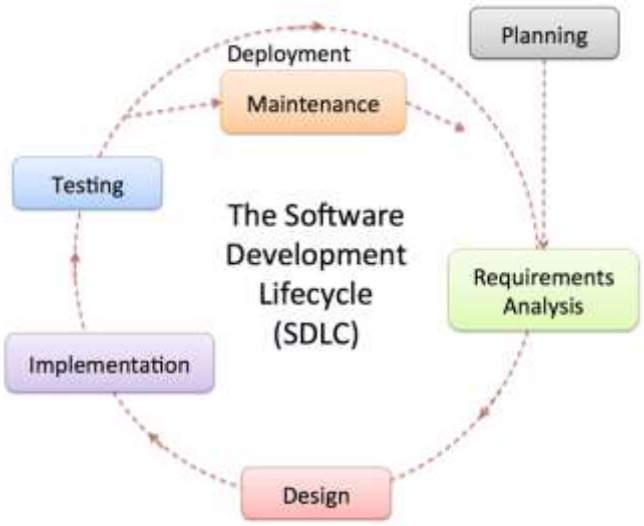
Deliver reliable, secure software
Ensure the system performs correctly under all expected conditions and protects data from unauthorized access, errors, and failures.

Reduce project risk
Identify and manage potential problems early—such as technical issues, unclear requirements, or resource gaps—to prevent project failure.

Improve communication across teams
Provide a structured process that keeps developers, testers, managers, and clients aligned through documentation, reviews, and regular feedback.

Ensure maintainability
Produce software that is easy to update, fix, and enhance in the future through clean code, proper documentation, and modular design.

Control project cost & timeline
Use planning, phase breakdowns, and monitoring to manage resources efficiently so the project stays within budget and is delivered on schedule.



SDLC Phase 1: Requirement Analysis

- Gathering user needs and technical constraints
- Identifies WHAT the system must do
- Produces SRS (Software Requirements Specification) document

Its important to interview stakeholders, creating use cases, and clarifying assumptions.

Techniques for Requirements Gathering

- Interviews
- Questionnaires
- Observation
- Document reviewl
- Joint Application Development (JAD) sessions

SDLC Phase 2: System Design

System Design is the second major phase of the Software Development Life Cycle.
It transforms the requirements collected in Phase 1 into a blueprint for building the actual system.

This phase answers the question:
“How will the system work and look?”

Purpose of the System Design Phase

To create a detailed plan that guides developers and testers during implementation.

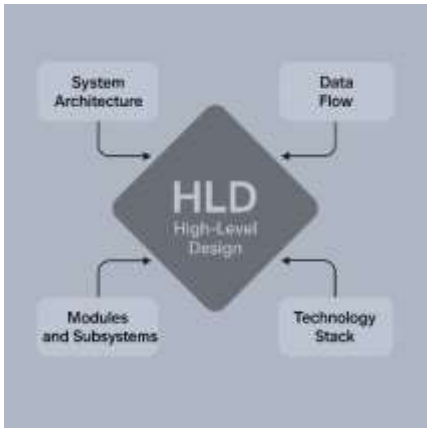
Key Activities in System Design

1. High-Level Design (HLD)

Focuses on the overall system structure:

- System architecture
- Modules/subsystems
- Data flow between components
- Technology stack (e.g., programming languages, frameworks)
- Integration with external systems

This gives a "big picture" view of the system.



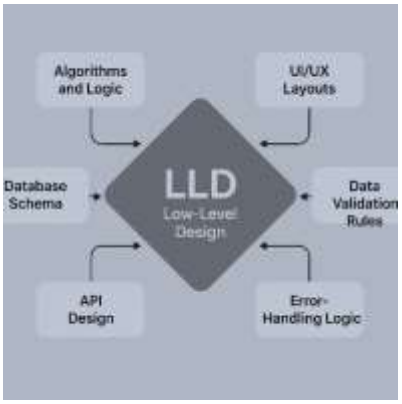
2. Low-Level Design (LLD)

Provides detailed specifications:

- Algorithms and logic for each module
- Database schema (tables, fields, relationships)
- API (Application Programming Interface) design
- UI/UX screen layouts
- Data validation rules
- Error-handling logic

LLD is what developers use directly to write the code.

Illustration



Outputs (Deliverables) of System Design

- System Architecture Diagram
- Database ER Diagram
- UML Diagrams (class diagrams, sequence diagrams, activity diagrams)
- UI mockups/wireframes
- Technical design document (TDD)
- Security and performance design considerations

Example
Design of login module: flow diagrams + database design.

SDLC Phase 3: Implementation (Coding)
Developers translate design into code
Programmers use the system's design documents (HLD/LLD) to write the actual source code that implements the required features and functionality.

Follow coding standards
Developers adhere to agreed-upon style rules and best practices (naming conventions, formatting, documentation) to produce clean, consistent, and maintainable code.

Use version control (Git)
Git is used to track code changes, manage branches, collaborate safely, and prevent code loss by storing all updates in a shared repository like GitHub or GitLab.

Conduct peer reviews
Before merging code, developers review each other's work to find bugs, improve quality, ensure standards are met, and share knowledge among the team.

SDLC Phase 4: Testing

Testing is the phase where the developed software is thoroughly checked to ensure it works correctly, meets requirements, and is free from defects. It verifies both functionality and quality before deployment.

Purpose of the Testing Phase

- To detect and fix bugs early
- To ensure the system meets user and business requirements
- To validate performance, reliability, and security
- To ensure the software is ready for deployment

Types of Testing

Non-functional Testing

Evaluates how the system performs rather than what it does.
It includes usability, reliability, scalability, speed, and overall user experience.

Performance Testing

Assesses the system's speed, responsiveness, and stability under different loads.
It checks how the system performs when many users or processes are active.

Security Testing

Ensures the software is protected against threats by checking vulnerabilities, authentication, authorization, data protection, and resistance to attacks.

Regression Testing

Performed after code changes or fixes to ensure new updates have not broken existing features.
It re-tests previously working functions.

SDLC Phase 5: Deployment

The Deployment Phase is where the fully tested software is released to the end users or production environment.
This phase ensures the system goes live smoothly and is ready for real-world use.

It answers the question:
"How do we deliver the software to users safely and correctly?"

Purpose of the Deployment Phase

- To install and configure the system in the real environment

- To make the application accessible to users
- To ensure a smooth transition from development to production
- To minimize downtime and deployment risks

Key Activities in Deployment

1. Prepare Production Environment
Set up servers, networks, security settings
Configure databases and hosting environments
Ensure all dependencies and versions are correct

2. Release the Software
Deployment may happen in stages:
Pilot release: A small group of users tests the live system
Full release: Everyone gets access once confirmed stable
Phased rollout: Gradual release to reduce risk

3. Data Migration
Move data from old systems to the new system
Clean, validate, and verify migrated data
Backup old and new data for safety

4. User Training & Documentation
Train end users, administrators, or support teams
Provide user guides, manuals, tutorials

5. Configuration & Integration
Connect the system to external APIs, payment gateways, sensors, or third-party services
Adjust settings based on environment-specific needs

6. Monitoring After Deployment
Watch system performance
Check for errors or unexpected behavior
Ensure everything works as expected in real use

Deployment Models (Examples)

On-premise deployment:
Installed on the company's servers
Cloud deployment:
Hosted on AWS, Azure, or Google Cloud
Continuous Deployment:

Automatic release from CI/CD pipeline A CI/CD pipeline is an automated process that software teams use to build, test, and deploy applications faster and with fewer errors.

Blue-Green Deployment: Two environments used to reduce downtime

SDLC Phase 6: Maintenance

- Bug fixing
- Security patching
- Feature enhancements
- Performance tuning