Software Development Life Cycle (SDLC)

A Guide to Efficient Software Development

1. Define Requirements

- Importance:-This phase is critical for converting the information gathered during the planning and analysis phase into clear requirements for the development team. This process guides the development of several important documents: a software requirement specification (SRS) or product specification, a Use Case document, and a Requirement Traceability Matrix document.
- Interconnection:- Requirements inform every subsequent phase, guiding design, implementation, testing, and deployment.

2. Design

- Importance:-The design phase is where you put pen to paper—so to speak. The original plan and vision are elaborated into a software design document (SDD) that includes the system design, programming language, templates, platform to use, and application security measures. This is also where you can flowchart how the software responds to user actions.In most cases, the design phase will include the development of a prototype model. Creating a pre-production version of the product can give the team the opportunity to visualize what the product will look like and make changes without having to go through the hassle of rewriting code.
- Interconnection:- Design bridges the gap between requirements and implementation, guiding developers in building the software.

3. Implementation

- Importance:- Implementation brings the design to life, transforming concepts and plans into functioning code.

- Interconnection:- Implementation directly reflects the design, ensuring that the software aligns with the intended functionality and user experience.

4. Testing

- Importance:- Before getting the software product out the door to the production environment, it’s important to have your quality assurance team perform validation testing to make sure it is functioning properly and does what it’s meant to do. The testing process can also help hash out any major user experience issues and security issues. In some cases, software testing can be done in a simulated environment. Other simpler tests can also be automated.

  The types of testing to do in this phase:

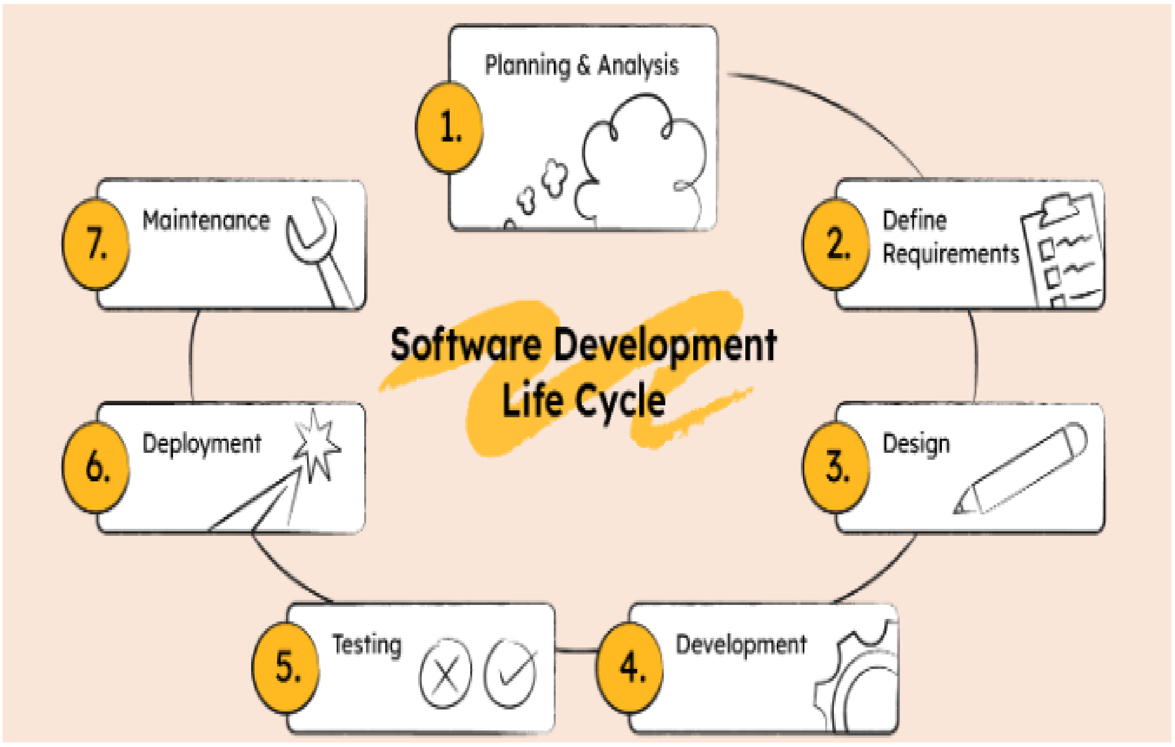
a. Performance testing: Assesses the software's speed and scalability under different conditions
b. Functional testing: Verifies that the software meets the requirements
c. Security testing: Identifies potential vulnerabilities and weaknesses
d. Unit-testing: Tests individual units or components of the software
e. Usability testing: Evaluates the software's user interface and overall user experience
f. Acceptance testing: Also termed end-user testing, beta testing, application testing, or field testing, this is the final testing stage to test if the software product delivers on what it promises

- Interconnection:- Testing verifies that the software meets the specified requirements, providing feedback for refinement and improvement.

5. Deployment

- Importance:- During the deployment phase, your final product is delivered to your intended user. You can automate this process and schedule your deployment depending on the type. For example, if you are only deploying a feature update, you can do so with a small number of users (canary release). If you are creating brand-new software, you can learn more about the different stages of the software release life cycle (SRLC).

- Interconnection:- Deployment marks the transition from development to operation, with feedback informing future updates and enhancements.


Software Development Life Cycle
1. Planning & Analysis
2. Define Requirements
3. Design
4. Development
5. Testing
6. Deployment
7. Maintenance