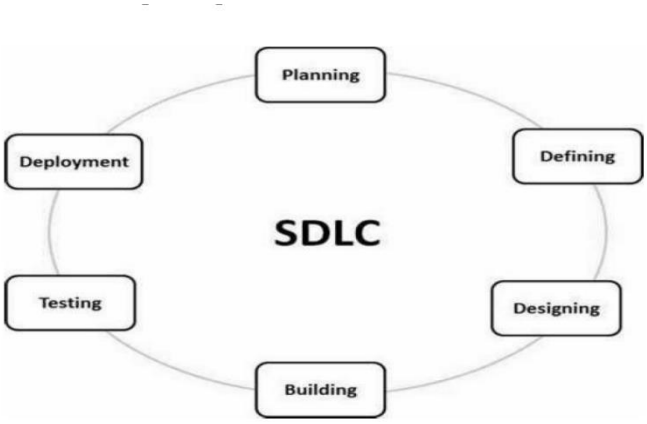
UNIT I – FUNDAMENTALS

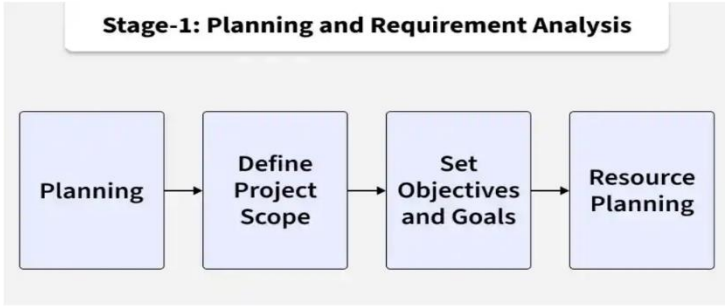
Principles of Testing – Phases of Software Project – Quality Assurance and Control – Verification and Validation - White Box Testing: Static Testing – Structural Testing – Challenges.

1.2 Phases of Software Project

- The Software Development Life Cycle (SDLC) is a structured process used to plan, design, develop, test, deploy, and maintain software. It ensures a systematic workflow and helps align software development with business goals and user requirements.
- Provides a clear and organized framework for managing development phases.
- Helps in early detection of defects, reducing overall cost and time.
- Ensures high-quality software delivery that meets user expectations.



Phase 1: Planning



The initial stage of software development, Planning, involves defining the software's purpose and scope, much like pinpointing our destination and plotting the best route. We uncover the tasks at hand during this phase and strategize for efficient execution.

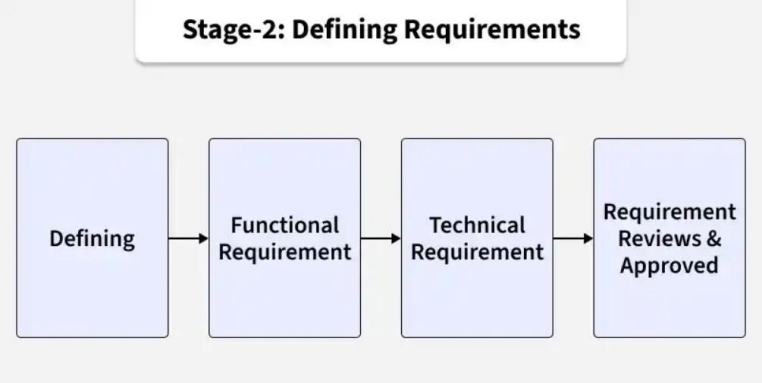
The team collaborates to understand the end-users' needs and the goals the software should meet. Essentially, we ask, "What problem will this software solve?" and "What value will it offer to the user?"

A feasibility study also takes place during the Planning phase. Developers and product teams evaluate technical and financial challenges that might affect the software's development or success.

During this phase, key documents such as the Project Plan and Software Requirement Specification (SRS) are created. These guides detail the software's functions, necessary resources, possible risks, and a project timeline.

The Planning phase fosters effective communication and collaboration within the team. By defining clear roles, responsibilities, and expectations, it lays a solid foundation for an efficient software development process.

Phase 2: Requirements Analysis



Phase 2 of the SDLC, Requirements Analysis, seeks to identify and record the precise requirements of the final users. In this phase, the team is looking to answer, "What are the expectations of our users from our software?" This is called requirements gathering.

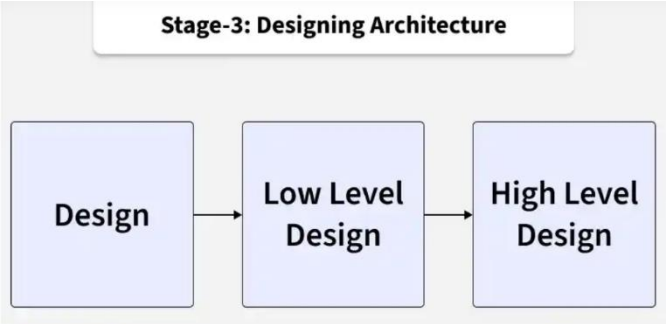
The project team collects information from stakeholders, including analysts, users, and clients. They conduct interviews, surveys, and focus groups to understand the user's expectations and needs. The process involves not only asking the right questions but also accurately interpreting the responses.

After collecting the data, the team analyzes it, distinguishing the essential features from the desirable ones. This analysis helps the team understand the software's functionality, performance, security, and interface needs.

These efforts result in a Requirements Specification Document. It outlines the software's purpose, features, and functionalities, acting as a guide for the development team and providing cost estimates if needed. To ensure its reliability, the document is validated for accuracy, comprehensiveness, and feasibility.

The success of the Requirements Analysis phase is pivotal for the entire project. Done right, it leads to a software solution that meets users' needs and exceeds their expectations.

Phase 3: Design



The Design phase is all about building the framework. The development team is responsible for software engineering and outlines the software's functionality and aesthetic. This ultimately results in the software product. The emphasis lies on outlining the software's structure, navigation, user interfaces, and database design. This phase ensures that the software is user-friendly and performs its tasks efficiently.

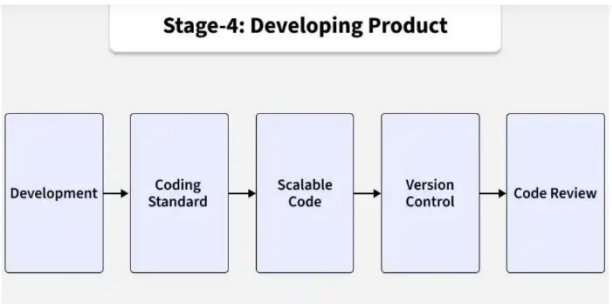
During this phase, the team undertakes key tasks such as creating data flow diagrams, developing entity-relationship diagrams, and designing user interface mock-ups. The team also identifies system dependencies and integration points while defining software limitations, including hardware constraints, performance requirements, and other system-related factors.

The culmination of these tasks is an exhaustive Software Design Document (SDD). This document serves as the roadmap for the team during the coding phase. It meticulously details the software's design, from system architecture to data design, and even user interface specifics.

High-Level Design (HLD) defines the system architecture, technology stack, database design, and major modules. Low-Level Design (LLD) specifies component logic, APIs, data structures, and workflows

The Design phase is the link between the software's purpose (established in the Planning and Requirements Analysis phases) and its execution (defined in the coding phase). It's an essential step in creating software that works efficiently and provides an excellent user experience.

Phase 4: Coding



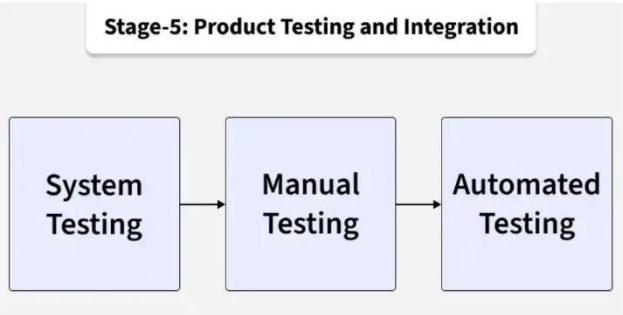
The Coding phase in the Software Development Life Cycle (SDLC) is when engineers and developers get down to business and start converting the software design into tangible code.

This development phase aims to develop software that is functional, efficient, and user-friendly. Developers use an appropriate programming language, Java or otherwise, to write the code, guided by the SDD and coding guidelines. This document, acting as a roadmap, ensures the software aligns with the vision set in earlier phases.

Another key aspect of this phase is regular code reviews. Team members carefully examine each other's work to identify any bugs or inconsistencies. These meticulous assessments uphold high code standards, ensuring the software's reliability and robustness. This phase also includes preliminary internal testing to confirm the software's basic functionality.

At the end of this phase, a functional piece of software comes to life. It embodies the planning, analyzing, and designing efforts of the preceding stages. Though it may not be flawless, it represents a significant stride towards a valuable software solution.

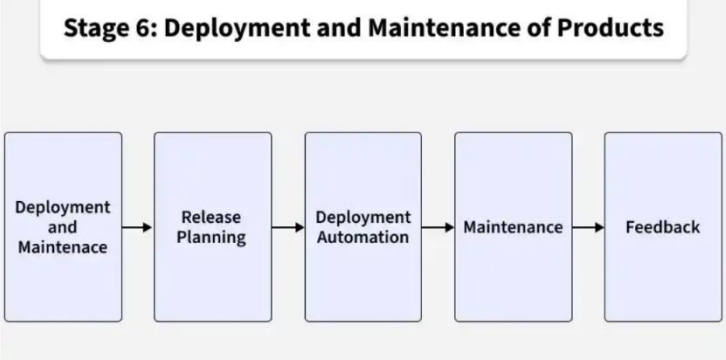
Phase 5: Testing



The Testing phase of the SDLC ensures that the software is reliable, secure, and free from defects before it is released to users. During this phase, the software is carefully checked to identify bugs, errors, and performance issues that may have occurred during development.

Testing begins by creating test cases based on software requirements and expected outcomes. Different testing methods such as unit testing, integration testing, system testing, and acceptance testing are performed to verify that both individual components and the complete system work correctly. If any bugs are found, they are documented and sent to developers for correction. After fixing the issues, the software is tested again to confirm proper functionality. This cycle continues until the software meets all quality and performance standards.

Phase 6: Deployment



After crafting a product with precision, it's time to present it to the users by pushing to the production environment. The Deployment phase involves rolling out the meticulously tested and fine-tuned software to its end-users.

A specific strategy is executed for the software's deployment to ensure minimal disruption to the user experience. Depending on the software and its audience, we might use different methods such as Big Bang, Blue-Green, or Canary deployments.

However, deployment isn't just about launching the software. It's about ensuring users can operate it with ease. This responsibility might involve creating user manuals, conducting training sessions, or offering on-site support.

The Deployment phase doesn't signal the end, but rather a notable milestone. It signifies the shift from a project phase to a product phase, where the software begins to fulfill its purpose.

Phase 7: Maintenance

The Maintenance phase of the Software Development Life Cycle focuses on providing continuous support and improvements to ensure the software performs efficiently and meets user expectations. During this phase, the software is updated and refined according to changing user needs and technological advancements.

Maintenance activities include fixing bugs, releasing updates, applying security patches, and improving system performance. User support is also an important part of this phase, helping users resolve issues and use the software effectively.

This phase also involves long-term planning, such as upgrading, redesigning, or replacing the software when necessary. These actions help keep the software relevant, reliable, and valuable over time.