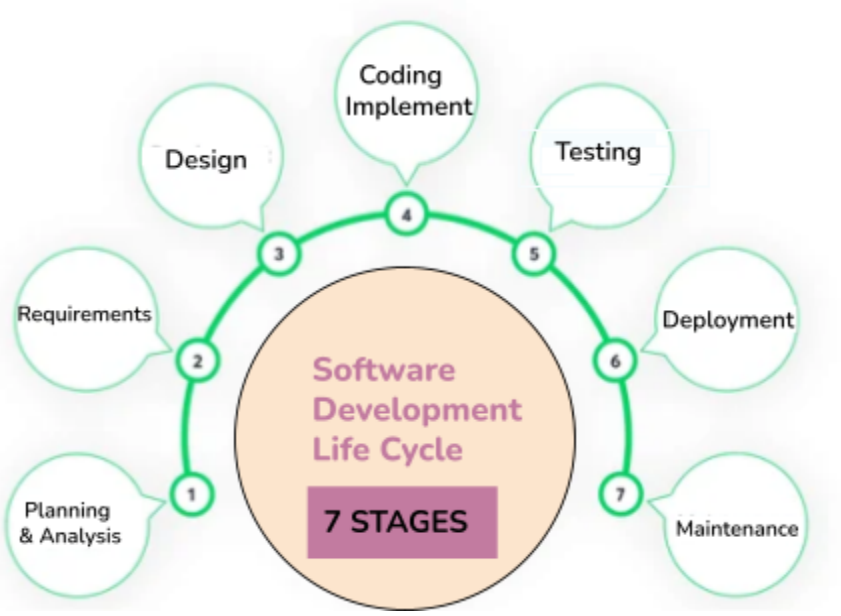
ASSIGNMENT - SOFTWARE DEVELOPMENT LIFE CYCLE

1) **SDLC Overview - Create a one-page infographic that outlines the SDLC phase (Requirements, Design,Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.**



The Software Development Life Cycle (SDLC) is a systematic process used in software engineering to guide the development of high-quality software. It consists of several phases, each with its own set of activities and deliverables.

Let's explore the importance of each phase and how they interconnect:

1. Planning: The first step in the software development life cycle is planning. It's when you gather the team to brainstorm, set goals, and identify risks. At this stage, the team will work together to devise a set of business goals, requirements, specifications, and any high-level risks that might hinder the project's success.

2. Requirement Analysis: This phase involves gathering and documenting the requirements of the software. It is crucial to understand the needs and expectations of the stakeholders. Proper requirement gathering ensures that the software meets the desired functionality and user experience.

ASSIGNMENT - SOFTWARE DEVELOPMENT LIFE CYCLE

3. Design: In the design phase, the software architecture and system components are planned. This includes creating detailed technical specifications, database design, and user interface design. A well-designed system ensures scalability, maintainability, and performance.

4. Coding: The coding phase involves writing the actual code based on the design specifications. It is important to follow coding standards and best practices to ensure code quality, readability, and maintainability. Thorough testing of the code is also essential to identify and fix any bugs or issues.

5. Testing: The testing phase is crucial for identifying defects and ensuring the software functions as expected. Different types of testing, such as unit testing, integration testing, and system testing, are performed to validate the software against the requirements. Testing helps in improving the quality and reliability of the software.

6. Deployment: In the deployment phase, the software is released and made available to users. This involves activities like installation, configuration, and data migration. Proper deployment ensures a smooth transition from development to production and minimizes any disruptions to the users.

7. Maintenance: The maintenance phase involves ongoing support and enhancements to the software. It includes bug fixes, updates, and adding new features based on user feedback and changing requirements. Regular maintenance ensures the software remains functional, secure, and up-to-date.

Interconnections:

- Requirements drive the design phase, as the design is based on the gathered requirements.
- The design phase informs the implementation phase, providing guidelines and specifications for writing code.
- Testing is closely linked to both the implementation and design phases. Testing verifies that the implementation meets the design specifications and ensures that the software functions correctly.
- Deployment depends on the successful completion of all previous phases. The software must pass testing and meet the specified requirements before it can be deployed to production

2) **Develop a case study analyzing the implementation of SDLC phases in a real-world engineering project. Evaluate how Requirement Gathering, Design,**

ASSIGNMENT - SOFTWARE DEVELOPMENT LIFE CYCLE

Implementation, Testing, Deployment, and Maintenance contribute to project outcomes.

Case Study: The Development of the Windows Operating System by Microsoft

The development of the **Windows Operating System by Microsoft** is a prime example of the Software Development Life Cycle (SDLC). Each phase of the SDLC played a crucial role in the successful development and deployment of the operating system.

Phase 1 Requirement Gathering: Understanding and archiving the product requirements was part of the first major phase of the Windows Framework SDLC. Microsoft had to define the functional requirements, decide which applications and tools would be necessary, and describe the user interface (UI) of the working Window Operating System.

Phase 2 Design: The design phase involved creating detailed plans and specifications for the system's hardware and software. The design had to take into account the various hardware configurations and user needs.

Phase 3 Implementation: Build the operating system and its software systems was the task of the implementation phase. A group of engineers was needed to complete this difficult task. All of the system's functions, including managing hardware resources and presenting a user interface, had to be managed by the software.

Phase 4 Testing: A crucial stage of the SDLC is testing. Testing for the Windows operating system included a variety of tasks, from comprehensive system testing to testing individual components. To make sure it could function on different machines, the system was also tested with different hardware setups.

Phase 5 Deployment: Distribution of the software to users was part of the Windows Operating System deployment phase. Direct sales, OEM partnerships, and online downloads were among the channels used to accomplish this.

Phase 6 Maintenance: The maintenance phase is the last one in the SDLC. This entails providing updates for the Windows operating system in order to address bugs, enhance functionality, and add new features. This is a continuous process that lasts beyond the first release.

3) **Research and compared SDLC models suitable for engineering projects. Present findings on Waterfall, Agile, Spiral and V- Model Approaches, emphasizing their advantages, disadvantages and applicability in different engineering contexts.**

ASSIGNMENT - SOFTWARE DEVELOPMENT LIFE CYCLE

Waterfall Model

A linear sequential flow is what the Waterfall model represents. The "Waterfall Model" refers to this approach because each phase in it begins only after the predetermined set of goals have been met and it has been approved.

Advantages:

- The Waterfall model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- This model works well for smaller projects where requirements are very well understood.

Disadvantages:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- This model is not suitable for the projects where requirements are at a moderate to high risk of changing.

Applicability: For projects with well-defined requirements and a stable product definition, the waterfall model works best. It is appropriate for tasks with a defined goal and consistent specifications.

Agile Model

The Agile model is a type of Iteration or Incremental model. Software is developed in incremental, rapid cycles, resulting in small incremental releases with each release building on previous functionality.

Advantages:

- The Agile model promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resources are saved as the defect density is less.

Disadvantages:

- The Agile model is highly dependent upon customer interaction, so if the customer is not clear, the team can be driven in the wrong direction.
- There is a lack of emphasis on necessary designing and documentation.

Applicability: The Agile model is best-suited for engineering projects with incremental development, where requirements evolve through collaboration between the self-organizing cross-functional teams.

ASSIGNMENT - SOFTWARE DEVELOPMENT LIFE CYCLE

Spiral Model

The Spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The Spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.

Advantages:

- The Spiral model allows for a high amount of risk analysis.
- It is good for large and mission-critical projects.
- This model provides strong approval and documentation control.

Disadvantages:

- The Spiral model can be a costly model to use.
- Risk analysis requires highly specific expertise.
- The project's success is highly dependent on the risk analysis phase.

Applicability: The Spiral model is used in high risk projects. It is ideal for projects which have some sort of risk involved, where there is a need to do Risk Analysis.

V-Model

The V-model is an SDLC model where the execution of processes happens in a sequential manner in a V-shape. It is also known as the Verification and Validation model.

Advantages:

- The V-Model is simple and easy to use.
- Each phase has specific deliverables.
- This model has a higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.

Disadvantages:

- The V-Model is very rigid.
- Little flexibility and adjusting scope is difficult and expensive.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.

Applicability: The V-Model is used when ample technical resources are available with needed technical expertise. It is a good model for testing point of view as the testing activities like planning, test designing happens well before coding.