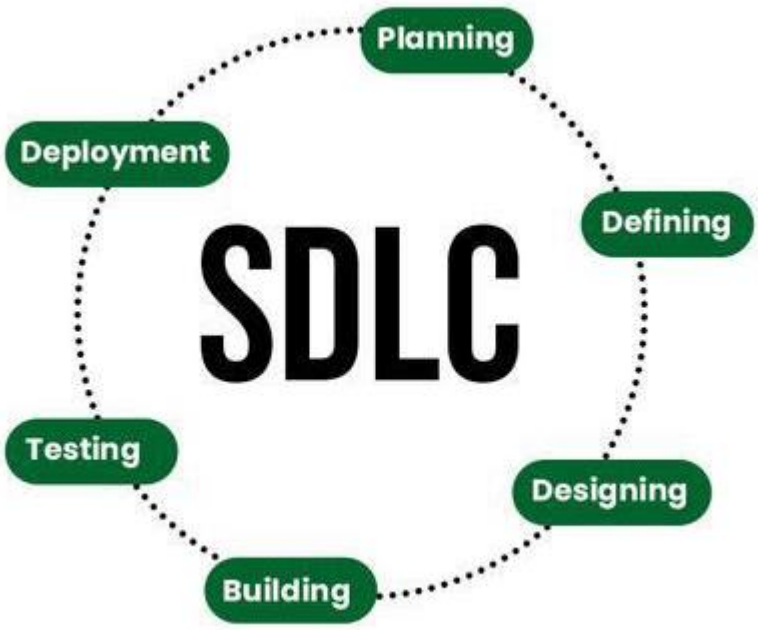
Assignment 2

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



Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a structured approach to software development that ensures the delivery of high-quality software within time and budget constraints. It consists of several interconnected phases, each with its own importance in the development process.

1. Requirements Phase:

Importance: This phase involves gathering and analysing user requirements to understand what the software needs to accomplish.

- Key Activities:

  Stakeholder interviews

  Requirement gathering sessions

  Use case analysis

- Outcome: A clear and comprehensive set of requirements documents that serve as the foundation for the entire project.

2. Design Phase:

- Importance: In this phase, the system architecture and design are defined based on the requirements gathered in the previous phase.
  - Key Activities:
    - System architecture design
    - Database design
    - User interface design
  - Outcome: Detailed design documents that guide the development team in building the software solution.

3. Implementation Phase:

- Importance: This phase involves the actual coding of the software based on the design specifications.
  - Key Activities:
    - Writing code
    - Unit testing
    - Code reviews
- Outcome: The development of the software product according to the specified requirements and design.

4. Testing Phase:

- Importance: Testing ensures that the software meets quality standards and functions correctly.
    - Key Activities:
        - Functional testing
        - Integration testing
        - Performance testing
- Outcome: Identification and resolution of defects, ensuring a stable and reliable software product.

5. Deployment Phase:

- Importance: Deployment involves releasing the software to users and maintaining it in a production environment.
    - Key Activities:
        - Deployment planning
        - User training
        - Rollout strategy
- Outcome: Successful delivery of the software to end-users, followed by ongoing support and maintenance.

Conclusion: The SDLC phases are interconnected and iterative, with each phase building upon the previous one. By following this structured approach, organizations can develop software that meets user requirements, is of high quality, and can be deployed successfully.

2]. Develop a case study analyzing the implementation of SDLC phases in a real-world engineering project library management system. Evaluate how Requirement Gathering, Design, Implementation, Testing, Deployment, and Maintenance contribute to project outcomes

Case Study: Implementing SDLC Phases in a Library Management System

Introduction: In this case study, we'll examine the implementation of Software Development Life Cycle (SDLC) phases in the development of a Library Management System (LMS). The LMS aims to automate library operations, including cataloguing, borrowing, and returning books, to enhance efficiency and user experience.

1. Requirement Gathering: During the requirement gathering phase, the development team collaborated with library staff to understand their needs and pain points. Interviews, surveys, and workshops were conducted to gather user requirements. Key features identified included book cataloguing, member management, borrowing and returning books, and reporting capabilities. By involving stakeholders early on, the team ensured that the LMS would meet user expectations and address specific library workflow challenges.

2. Design: In the design phase, the development team translated the gathered requirements into a detailed system architecture and user interface design. They designed a modular system with separate components for the frontend interface, backend database, and business logic. User-friendly interfaces were created for librarians to perform tasks such as adding new books, managing member accounts, and generating reports. The design phase laid the foundation for the development process, ensuring clarity and alignment with project objectives.

3. Implementation: In the implementation phase, the development team began coding the LMS based on the design specifications. They followed coding best practices and utilized appropriate technologies to build a scalable and maintainable system. The frontend was developed using HTML, CSS, and JavaScript, while the backend was implemented using a combination of Python and Django framework for rapid development. Continuous integration and version control systems were employed to streamline the development process and ensure code quality.

4. Testing: Testing was a critical phase in ensuring the quality and reliability of the LMS. The development team conducted various types of testing, including unit testing, integration testing, and user acceptance testing (UAT). Unit tests were written to validate individual components, while integration tests verified the interaction between different modules. UAT involved library staff testing the system in a simulated environment to ensure that it met their requirements and was intuitive to use. Defects and issues identified during testing were promptly addressed to maintain the integrity of the system.

5. Deployment: Once testing was successfully completed, the LMS was deployed to the production environment. Deployment planning involved coordinating with IT staff to ensure that the necessary infrastructure and resources were in place. User training sessions were conducted to familiarize library staff with the new system and address any questions or concerns. A phased rollout strategy was implemented to minimize disruption to library operations and facilitate a smooth transition to the new system.

6. Maintenance: The maintenance phase involved on-going support and updates to the LMS to address bugs, introduce new features, and adapt to changing requirements. A helpdesk system was established to provide technical assistance to library staff and address any issues that arose post-deployment. Regular software updates were released to improve system performance, enhance security, and incorporate user feedback. Maintenance activities were crucial in ensuring the long-term usability and effectiveness of the LMS.

3] Research and compare 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.

In engineering projects, selecting the appropriate Software Development Life Cycle (SDLC) model is crucial for successful project execution. This assignment compares four commonly used SDLC models: Waterfall, Agile, Spiral, and VModel. Each model has its own set of advantages, disadvantages, and suitability for different engineering contexts.

1. Waterfall Model:

Advantages:

- Clear milestones and deliverables.
- Well-suited for projects with stable requirements
- Sequential and easy to understand
- Emphasizes documentation and thorough planning.

Disadvantages:

- High risk of late-stage failures if requirements are misunderstood or change.
- Testing is deferred until the end, potentially leading to high rework costs.
- Limited flexibility for changes after the initial planning stage.

Applicability:

- Best suited for projects with well-defined and stable requirements, such as simple software or hardware projects with low complexity and minimal expected changes.

2. Agile Model:

Advantages:

- Continuous feedback loop between developers and stakeholders.
- Reduced risk of project failure due to early detection of issues.
- Iterative and incremental, allowing for flexibility and adaptation to changing requirements.
- Early and frequent delivery of working software.

Disadvantages:

- Requires active involvement and commitment from stakeholders throughout the project.
- May lack documentation, which can lead to knowledge loss over time.
- Not suitable for projects with strict regulatory or compliance requirements.

Applicability:

- Well-suited for dynamic projects with evolving requirements, such as software development, where rapid deployment and flexibility are essential

3. Spiral Model:

Advantages:

- Allows for iterative development with each loop addressing identified risks.

- Incorporates risk management throughout the project lifecycle.
- Well-suited for large-scale and complex projects where risks are high.

Disadvantages:

- Complex and costly due to the need for risk analysis and mitigation.
- Can lead to scope creep if risks are not managed effectively.
- Requires experienced and skilled project management

Applicability:

- Suitable for projects with high levels of uncertainty and complexity, such as software projects involving cutting-edge technologies or critical systems development.

4. V-Model:

Advantages:

- Emphasizes verification and validation activities throughout the development process.
- Clearly defines relationships between development stages and corresponding testing phases.
- Ensures early detection and correction of defects.

Disadvantages:

- Sequential nature may not accommodate changes well.

- Heavy emphasis on documentation may lead to slower progress.
- Testing activities may be delayed until late stages, increasing the risk of defects.

Applicability:

- Suitable for projects with strict regulatory or compliance requirements, such as safety-critical systems or projects where traceability is essential.