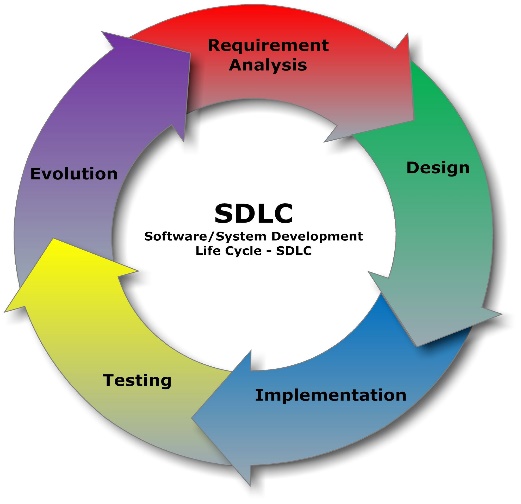
**Day-2**: **Assignment 1,2&3**

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

**Infographic**: Software Development Life Cycle (SDLC) Overview



**1.** **Requirements**:

Description: Gathering and documenting what the users and stakeholders need from the software.

Importance: Ensures a clear understanding of project goals and sets the foundation for all future phases.

Interconnections: Forms the basis for the Design phase.

**2.** **Design**:

Description: Crafting detailed architecture and design plans based on the requirements.

Importance: Provides a roadmap for developers, ensuring the system's structure is well-planned.

Interconnections: Guided by Requirements and critical for the Implementation phase.

**3.** **Implementation**:

Description: Writing the actual code to build the software as per the design specifications.

Importance: Converts designs into a functional software product.

Interconnections: Follows the Design phase and precedes Testing.

**4.** **Testing**:

Description: Testing the software to identify and fix defects and ensure it meets the requirements.

Importance: Ensures the software is of high quality and functions correctly.

Interconnections: Utilizes the Implementation output and must be completed before Deployment.

**5.** **Deployment**:

Description: Releasing the final product to users.

Importance: The software becomes available for actual use, and user feedback is gathered.

Interconnections: Follows successful Testing and marks the transition to maintenance.

**Assignment 2:**

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

**Case Study**: Implementation of SDLC Phases in the Development of an Autonomous Vehicle System

**Project Overview**

In this case study, we analyse the implementation of the Software Development Life Cycle (SDLC) phases in the development of an autonomous vehicle system by a leading automotive company, Auto Drive Inc. The project aims to develop a fully autonomous vehicle that can navigate urban environments safely and efficiently.

SDLC Phases

1. **Requirement** **Gathering**

->To collect and define the detailed requirements of the autonomous vehicle system from various stakeholders, including customers, regulatory bodies, and internal teams.

Stakeholder Interviews and Surveys: Conducted extensive interviews and surveys with potential users, transportation authorities, and automotive engineers to understand the expectations and regulations.

Use Case Scenarios: Developed detailed use case scenarios to cover various aspects of autonomous driving, including urban navigation, highway driving, and emergency handling.

Requirements Documentation: Created comprehensive documentation outlining functional and non-functional requirements, such as safety standards, performance metrics, and user interface specifications.

**Outcome**:

- >A well-defined set of requirements that served as a foundation for the subsequent phases.

- >Identification of critical features such as obstacle detection, lane-keeping, and pedestrian recognition.

2. **Design**

-> To translate the requirements into a structured design that serves as a blueprint for the implementation phase.

System Architecture Design: Developed a high-level architecture that includes sensor integration, data processing modules, decision-making algorithms, and vehicle control systems.

Component Design: Designed individual components such as the perception system (for obstacle detection), the planning system (for path planning), and the control system.

User Interface Design: Created mock-ups and prototypes of the user interface for interacting with the autonomous vehicle.

**Outcome**:

- >A detailed design specification that guides the implementation team.

- >Identification of key technologies and tools required, such as LIDAR, computer vision algorithms, and real-time operating systems.

3. **Implementation**

-> To build the autonomous vehicle system according to the design specifications.

Modular Development: Divided the project into smaller modules, each developed by specialized teams focusing on perception, planning, and control.

Agile Methodology: Adopted an agile development approach with iterative sprints, allowing for continuous feedback and incremental improvements.

Integration and Testing: Conducted regular integration sessions to ensure that individual modules work together seamlessly.

**Outcome**:

- >Development of a working prototype of the autonomous vehicle.

- >Successful integration of various components, resulting in a cohesive system.

4. **Testing**

-> To validate the functionality, performance, and safety of the autonomous vehicle system.

Unit Testing: Performed rigorous unit tests on individual components to ensure they meet the specified requirements.

System Testing: Conducted comprehensive system tests in simulated environments to validate the overall functionality and performance.

Field Testing: Executed extensive field tests in controlled environments and real-world conditions to assess the vehicle's behaviour in various scenarios.

**Outcome**:

->Identification and resolution of critical bugs and performance issues.

->Validation of the system's ability to handle complex driving situations safely.

5. **Deployment**

->To deploy the autonomous vehicle system for commercial use.

Pilot Programs: Launched pilot programs in selected cities to gather real-world data and user feedback.

Regulatory Approvals: Worked closely with regulatory bodies to ensure compliance with safety standards and obtain necessary approvals.

User Training: Provided training programs for users to familiarize them with the autonomous vehicle's features and operation.

**Outcome**:

->Successful deployment of the autonomous vehicle system in multiple cities.

->Positive user feedback and increased public acceptance of autonomous driving technology.

6. **Maintenance**

-> To ensure the continuous operation and improvement of the autonomous vehicle system.

Monitoring and Support: Established a monitoring system to track the performance and health of deployed vehicles.

Software Updates: Regularly released software updates to fix bugs, enhance features, and improve safety.

User Feedback Integration: Continuously gathered and incorporated user feedback to refine the system.

**Outcome**:

->Improved reliability and performance of the autonomous vehicle system over time.

->Ongoing enhancement of features and user experience.

Evaluation of SDLC Phases Contribution to Project Outcomes

1. Requirement Gathering: Ensured a clear understanding of stakeholder needs, leading to a well-defined project scope and objectives.

2. Design: Provided a solid blueprint that guided the development process, reducing ambiguity and aligning the team with the project goals.

3. Implementation: Facilitated the creation of a functional prototype through modular development and agile practices, enabling iterative improvements.

4. Testing: Validated the system's functionality and safety, ensuring that it met the required standards before deployment.

5. Deployment: Enabled a smooth transition from development to real-world use, with pilot programs and regulatory compliance paving the way for broader adoption.

6. Maintenance: Ensured the system's longevity and relevance through continuous monitoring, updates, and user engagement.

**Assignment 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.

Comparison of SDLC Models for Engineering Projects

**1. Waterfall Model**

Overview: The Waterfall model is a linear sequential approach where each phase must be completed before the next begins. It is one of the earliest SDLC models.

Phases:

1. Requirement Gathering

2. System Design

3. Implementation

4. Integration and Testing

5. Deployment

6. Maintenance

Advantages:

- >Simplicity and Ease of Use: Clear structure and easy to understand.

- >Defined Stages: Each phase has specific deliverables and review processes.

- >Documentation: Extensive documentation is generated, which can be useful for future reference and maintenance.

Disadvantages:

- >Inflexibility: Changes are difficult to implement once the process has moved to the next stage.

- >Late Testing: Issues may not be detected until later stages, which can be costly to fix.

- >Poor for Complex Projects: Not ideal for projects where requirements might evolve.

Applicability:

-> Suitable for projects with well-defined requirements that are unlikely to change.

- >Best for simple or smaller-scale engineering projects with clear objectives.

**2. Agile Model**

Overview: Agile is an iterative and incremental model that focuses on flexibility, customer feedback, and rapid delivery. It promotes adaptive planning and evolutionary development.

Phases:

- >Agile does not have fixed phases but works in iterative cycles called sprints, typically lasting 2-4 weeks.

Advantages:

- >Flexibility and Adaptability: Easily accommodates changes in requirements.

- >Customer Involvement: Regular feedback from stakeholders ensures the product meets their needs.

- >Continuous Improvement: Regular retrospectives and iterative development improve the process and product over time.

Disadvantages:

- >Less Predictability: Due to its flexible nature, it can be challenging to predict the final outcome early in the project.

- >Requires Experienced Team: Teams need to be skilled in agile practices and self-organization.

- >Documentation Can Suffer: Focus on rapid development can sometimes lead to insufficient documentation.

Applicability:

- >Suitable for projects with dynamic or evolving requirements.

- >Ideal for complex engineering projects that benefit from continuous feedback and iterative development.

**3. Spiral Model**

Overview: The Spiral model combines iterative development with the systematic aspects of the Waterfall model. It emphasizes risk assessment and management.

Phases:

1. Planning

2. Risk Analysis

3. Engineering and Development

4. Evaluation

Advantages:

- >Risk Management: Proactive identification and mitigation of risks.

- >Iterative Development: Allows for incremental refinement through each spiral iteration.

- >Flexibility: Can incorporate changes and feedback at multiple stages.

Disadvantages:

- >Complexity: More complex to manage compared to other models.

- >Cost: Can be expensive due to its risk analysis and management processes.

- >Documentation Intensive: Requires comprehensive documentation at each spiral.

Applicability:

- >Suitable for large, complex, and high-risk engineering projects.

- >Ideal for projects where risk management is crucial.

**4. V-Model (Verification and Validation Model)**

Overview: The V-Model is an extension of the Waterfall model that emphasizes validation and verification at each development stage.

Phases:

1. Requirement Analysis

2. System Design

3. Architecture Design

4. Module Design

5. Coding

6. Unit Testing

7. Integration Testing

8. System Testing

9. Acceptance Testing

Advantages:

->Structured Approach: Clear and structured phases with specific deliverables.

- >Early Testing: Emphasizes testing from the beginning, reducing the likelihood of issues in later stages.

- >High Quality: Ensures thorough verification and validation processes.

Disadvantages:

- >Inflexibility: Like the Waterfall model, changes are difficult to accommodate once development has begun.

- >High Cost of Change: Early stages require detailed planning and design, making changes costly.

- >Documentation Overhead: Generates a lot of documentation, which can be time-consuming.

Applicability:

- >Suitable for projects where high quality and thorough testing are critical.

- >Ideal for safety-critical engineering projects such as aerospace, automotive, and medical device development.