# Assignment No : 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: The Software Development Life Cycle (SDLC):

Title: **The Software Development Life Cycle (SDLC) - Phases and Importance**

**Introduction :**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

**Phase of SDLC:**

1. **Planning and Requirement Analysis**
2. **Defining Requirements**
3. **Designing the Product Architecture**
4. **Building or Developing the Product**
5. **Testing the Product**
6. **Deployment in the Market and Maintenance**

**Sequential phases :**

1. **Requirements:**

**Description:** This phase involves gathering and analyzing the requirements from stakeholders to understand what they want from the software.

**Importance**

Ensures clear understanding of user needs.

Forms the foundation for the next phases.

**Output:** Requirements Specification Document.

1. **Design:**

**Description:** In this phase, the software's architecture and design are created based on the requirements.

**Importance:**

Provides a blueprint for developers.

Ensures that all components are properly defined.

**Output:** Design Documents, including system architecture and data models.

1. **Implementation:**

**Description:** This is the phase where the actual coding of the software takes place.

**Importance:**

Translates design into a functional system.

Includes coding and internal testing.

**Output:** Source Code, Build.

1. **Testing:**

**Description:** In this phase, the software is tested to find and fix bugs and to ensure that it meets the requirements.

**Importance:**

Ensures software quality and functionality.

Identifies and resolves defects.

**Output:** Test Plans, Test Cases, Test Reports.

1. **Deployment:**

**Description:** The software is deployed to the production environment where it will be used by the end-users.

**Importance:**

Delivers the final product to users.

Ensures proper functioning in the real environment.

**Output:** Deployed Software, Deployment Plans.

**Interconnections:**

**Requirements → Design:** Clear requirements guide the design process.

**Design → Implementation:** A well-defined design ensures smooth implementation.

**Implementation → Testing:** Proper coding leads to effective testing.

**Testing → Deployment:** Thorough testing ensures a successful deployment.

**Continuous Feedback Loop:** Each phase provides feedback to the previous ones for continuous improvement.

**Conclusion:**

Understanding and following the SDLC phases is crucial for developing high-quality software that meets user expectations and functions correctly. Each phase builds upon the previous one and contributes to the success of the project. The interconnected nature of these phases ensures a comprehensive approach to software development, minimizing risks and maximizing efficiency.

**Infographic one pager layout** :

Phases of SDLC (Left Side)

1.Requirement

2.Design

3.Implementation

4.Testing

5.Deployment

2.Interconnections (Right Side)

Flowchart or arrows connecting each phase

And The Feedback loop illustration

**Title and Subtitle:** Centrally positioned at the top for immediate recognition.

**Phases of SDLC (Left Side):** Each phase (Requirements, Design, Implementation, Testing, Deployment) is placed in a rounded box with color coding. Descriptions of each phase include:

**Brief Description**

**Importance**

**Output**

**Interconnections (Right Side):** Descriptions of how each phase connects to the next, emphasizing the continuous feedback loop.

**Footer:** A concluding summary that reinforces the importance and interconnectedness of the SDLC phases.

# Assignment 2

Develop a case study analyzing 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 has the context menu .

Case Study: **Implementing SDLC Phases in a Blogging Platform Development Project**

Requirement Gathering:

The development team interviews bloggers and content creators to understand their needs for a new blogging platform. They identify requirements such as user authentication, blog post creation, comment management, and basic analytics.

Design:

Based on the gathered requirements, the team designs the architecture of the blogging platform, including the database structure, user interface layouts, and navigation flow. They create mockups or prototypes to visualize the design.

Implementation:

Developers begin coding the blogging platform, building features such as user registration, blog post creation, comment sections, and user profile pages. They use programming languages and frameworks suited to web development, such as HTML, CSS, JavaScript, and a backend language like Python or Node.js.

Testing:

The platform undergoes testing to ensure functionality and usability. Testers check that users can register, create blog posts, leave comments, and navigate the site smoothly. They also test for compatibility with different web browsers and devices.

Deployment:

Once testing is complete, the blogging platform is deployed to a web server and made accessible to users. This involves configuring the server, setting up domain name and hosting, and ensuring proper security measures are in place.

Maintenance:

After deployment, the development team provides ongoing maintenance and support for the blogging platform. They address any bugs reported by users, implement new features based on feedback, and perform regular updates to keep the platform secure and up-to-date.

Evaluation:

Each phase of the SDLC contributes to the successful development and deployment of the blogging platform. Requirement gathering ensures alignment with user needs, design creates a blueprint for implementation, implementation brings the design to life, testing ensures quality and usability, deployment makes the platform available to users, and maintenance ensures its continued functionality and relevance over time.

# 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.

**Project Scenario: Development of a Smart Home Automation System**

**Waterfall Model:**

Advantages:

With well-defined requirements and a clear scope, the Waterfall model can work well for this project. Each phase, such as requirement analysis, design, implementation, testing, deployment, and maintenance, can be carefully planned and executed.

Disadvantages:

However, if there are changes in requirements or technology during the development process, it can be challenging to incorporate them due to the model's rigid nature. Additionally, testing of the entire system may only occur towards the end, increasing the risk of finding critical issues late in the project.

**Agile Model:**

Advantages:

Given the dynamic nature of technology and user needs in smart home systems, Agile would be suitable. The incremental and iterative approach allows for continuous feedback from users, ensuring that the system meets their evolving requirements.

Disadvantages:

However, managing frequent iterations and ensuring consistent customer involvement may be challenging, especially if stakeholders are not readily available for feedback. Additionally, the focus on working software over comprehensive documentation may pose challenges in documenting system architecture and requirements.

**Spiral Model:**

Advantages:

The Spiral model's emphasis on risk management makes it a good fit for projects like smart home automation, where there are uncertainties in technology integration and compatibility. Each iteration allows for risk identification and mitigation, ensuring a more robust and reliable system.

Disadvantages:

However, the Spiral model can be resource-intensive and complex to manage, requiring a skilled team and significant upfront planning. It may also lead to delays if risks are not effectively managed or if there are frequent changes in requirements.

**V-Model:**

Advantages:

The V-Model's focus on parallel development and testing phases makes it suitable for projects where thorough testing is crucial, such as smart home automation systems. Each stage of development is associated with corresponding testing activities, ensuring comprehensive test coverage.

Disadvantages:

Yet, the V-Model's sequential nature may not be well-suited for projects with evolving requirements or where rapid adaptation is necessary. Changes late in the development cycle can be costly and time-consuming to implement.

# Assignment 4

Write an assignment to write a methodology using Test Driven ,Behavior driven and Feature driven model to create your own chat application ,somewhat similar to what’s app.

Test Driven methodology:

Test-driven development (TDD) is a method of coding in which you first write a test and it fails, then write the code to pass the test of development, and clean up the code. This process recycled for one new feature or change. In other methods in which you write either all the code or all the tests first, TDD will combine and write tests and code together into one.

Behavior driven methodology:

Behavior-driven development (BDD) is an Agile software development methodology in which an application is documented and designed around the behavior a user expects to experience when interacting with it.

Features driven methodology: It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive of feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

**Create own new chatbox : Build a Simple Group Chat Application using TDD, BDD, and FDD**

Objective: Create a basic group chat application that allows multiple users to chat together in real-time, similar to WhatsApp groups. Use TDD, BDD, and FDD methodologies to ensure a robust and maintainable application.

Requirements:

1. Multiple users can join a group chat

2. Users can send and receive messages in real-time

3. Display a list of online users in the group

4. Simple user authentication (username and password)

Methodology:

1. TDD: Write unit tests for each feature, implement, and refactor

2. BDD: Define scenarios and behaviors for each feature, write step definitions, and run scenarios

3. FDD: Identify key features, prioritize, and develop incrementally

Features to Implement:

1. Group Chat Creation

2. User Join/Leave Group

3. Real-time Message Display

4. Online User List

Deliverables:

1. A working group chat application

2. Unit tests and BDD scenarios

3. Feature roadmap and development timeline

4. Brief report explaining the methodologies used