Debakshmi Pan

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.

1. Requirements

- Objective: Define what the software should do.
- Key Activities:
    - Gather requirements from stakeholders.
    - Document functional and non-functional requirements.
    - Analyze feasibility.
- Importance: Ensures a clear understanding of what is needed, preventing scope creep and miscommunication.

2. Design

- Objective: Plan the software structure.
- Key Activities:
    - Create architectural design.
    - Design database schema.
    - Develop detailed technical specifications.
- Importance: Provides a blueprint for development, ensuring all components will work together seamlessly.

3. Implementation

- Objective: Write the code and build the software.
- Key Activities:
    - Develop and integrate modules.
    - Follow coding standards.
    - Perform initial debugging.
- Importance: Translates designs into a functional software product.

4. Testing

- Objective: Ensure the software is defect-free.
- Key Activities:
    - Conduct unit, integration, and system testing.
    - Perform user acceptance testing (UAT).
    - Identify and fix bugs.

- Importance: Verifies that the software meets requirements and works as intended, ensuring quality and reliability.

5. Deployment

- Objective: Release the software to users.
- Key Activities:
    - Plan and execute deployment.
    - Provide user training and documentation.
    - Monitor and maintain post-deployment.
- Importance: Delivers the product to users and ensures it operates smoothly in the live environment.

Interconnection of Phases

- Sequential Flow: Each phase builds upon the previous one, ensuring a structured development process.
- Feedback Loops: Continuous feedback between phases for adjustments and improvements.
- Iterative Process: Revisiting and refining phases as needed to enhance the software quality.

<u>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.</u>

Case Study: Implementing SDLC in a Smart Home Automation System

Project Overview

A tech company developed a Smart Home Automation System to control home appliances via a mobile app.

1. Requirement Gathering

- What They Did:
    - Talked to future users to understand their needs.

    - Wrote down what the system should do.
- Why It Matters:
    - Clear requirements prevented misunderstandings and ensured the project met user needs.

2. Design

- What They Did:
    - Planned how the system would work.
    - Created diagrams and models of the system.
- Why It Matters:
    - A good plan made the development process smoother and helped identify potential problems early.

3. Implementation

- What They Did:
    - Wrote the code for the system in small, manageable parts.
    - Regularly reviewed and tested the code.
- Why It Matters:
    - Ensured that the system was built correctly and all parts worked together.

4. Testing

- What They Did:
    - Tested each part of the system individually and together.
    - Had real users test the system before final release.
- Why It Matters:
    - Found and fixed bugs, ensuring the system worked well for users.

5. Deployment

- What They Did:
    - Gradually released the system to users.
    - Provided instructions and help for new users.
- Why It Matters:
    - Smooth release ensured users could start using the system easily and without problems.

6. Maintenance

- What They Did:
    - Monitored the system and made improvements.
    - Fixed any new issues that came up.
- Why It Matters:
    - Kept the system running well and improved it over time based on user feedback.

Conclusion

Following the SDLC phases helped the tech company build a high-quality Smart Home Automation System that met user needs, was reliable, and could be easily improved in the future. Each phase was crucial for the project's success.

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.

Waterfall Model:

- What is it?
  - Imagine building a house step by step, starting from laying the foundation, building walls, adding rooms, and finally, decorating. The Waterfall model is like following a strict plan, where each step is completed before moving to the next.
- How does it work?
  - Requirements: List what the house needs - how many rooms, what type of roof, etc.
  - Design: Draw detailed plans for each room and how they fit together.
  - Implementation: Build the house according to the plans.
  - Testing: Check everything works as it should - lights turn on, doors close properly.
  - Deployment: Move into the house once everything is finished.
- Pros:
  - Easy to Understand: Follows a clear order, like ticking off items on a checklist.
  - Clear Milestones: Each step has a clear finish point before moving to the next.
- Cons:
  - No Going Back: Once you've built a wall, it's hard to change without starting over.
  - Late Surprises: Problems might pop up late, like finding a leaky pipe after you've finished painting.

Agile Model:

- What is it?
    - Imagine building a house with Lego bricks. You start with a simple structure and keep adding or changing parts until it's just right. The Agile model is like building software in small chunks, making changes as you go based on feedback.
- How does it work?
    - Backlog: Make a list of what the house should have - a kitchen, a living room, etc.
    - Sprint Planning: Decide which parts of the house to build first.
    - Development: Build those parts quickly.
    - Review: Show what you've built to your friends and see if they like it.
    - Retrospective: Talk about what worked well and what could be better for the next part.
- Pros:
    - Flexible: You can make changes as you go, even after you've started building.
    - Always Improving: Your house gets better with each change you make based on feedback.
- Cons:
    - Need Involvement: You have to talk to others a lot to know what changes to make.
    - Less Paperwork: You might not have detailed plans like in the Waterfall model, so it's easier to forget things.

Spiral Model:

- What is it?
    - Imagine building a car and testing different parts as you go to make sure everything works smoothly. The Spiral model is like taking small steps, checking for problems, and then moving on, kind of like a safety net.
- How does it work?
    - Planning: Think about what kind of car you want to build and what could go wrong.
    - Risk Analysis: Look at potential problems and how to avoid them.
    - Development & Testing: Build a small part of the car and test it to see if it works.
    - Evaluation: Check if the part works well and if there are any new risks.
    - Next Iteration: Decide whether to keep going or change something before moving on.
- Pros:
    - Safe Steps: You can find and fix problems early before they become big issues.
    - Change-Friendly: You can make changes at any step without starting over.
- Cons:

  - Takes Time: Checking for problems at every step can make the process slower.
  - More Complicated: You need to think about risks and potential problems a lot.

V-Model:

- What is it?
  - Imagine building a paper airplane and checking how well it flies at every step. The V-Model is like making sure everything works perfectly by testing it thoroughly at each stage before moving on.

- How does it work?
  - Requirements Analysis: Think about what your airplane should do - how far it should fly, etc.
  - Architectural Design: Plan how you'll fold the paper to make the airplane.
  - Module Design: Decide how big each part of the airplane should be.
  - Implementation: Fold the paper according to your plans.
  - Unit Testing: Test each part of the airplane separately - how well the wings stay on, etc.
  - Integration Testing: Test how well all the parts fit together.
  - System Testing: Test the airplane as a whole - how far it flies, how straight it goes.
  - Acceptance Testing: Check if the airplane meets all your requirements - if it flies far enough, etc.
- Pros:
  - Thorough Testing: Makes sure everything works perfectly before moving on.
  - Clear Process: Each step has its own tests, so you know when it's done.
- Cons:
  - No Going Back: If you find a problem late, it can be hard to fix without starting over.
  - Takes Time: Testing everything at each step can make the process slower.