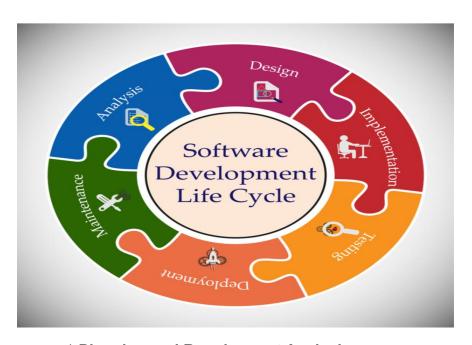
Assignment 2: Develop a case study analyzing the implementation of SDLC phases in a real-world engineering project. Evaluate how Requirement Gathering, Design, Implementation, Testing, Deployment, and Maintenance contribute to project outcomes.

==Software Development Life Cycle (SDLC)



1. Planning and Requirement Analysis:

- o Understand client needs and define project scope.
- o Identify quality assurance requirements and assess project risks.

2. System Design:

- o Create detailed specifications based on requirements.
- Develop blueprints, flowcharts, and other design artifacts.

3. Implementation (Coding):

- o Write actual source code based on the design.
- Integrate different modules and components.

4. Testing:

- o Verify system functionality and identify defects.
- o Conduct unit, integration, and acceptance testing.

5. **Deployment**:

- o Roll out the system to users.
- o Install, configure, and ensure data migration.

6. Maintenance and Support:

o Regularly update, fix bugs, and assist users.

C

Case Study: Building an E-Commerce Platform

1. Requirement Gathering:

- Objective: Understand client needs and define project scope.
- **Implementation**: Conduct interviews, surveys, and workshops with stakeholders. Gather functional and non-functional requirements.
- **Impact**: Clear requirements prevent scope creep and ensure alignment with business goals.

1. Assign Roles:

- o Identify project stakeholders (internal and external).
- o Conduct interviews, workshops, and surveys.
- Document functional and non-functional requirements.

2. Visualize with Diagrams:

o Use tools like Microsoft Visio, Lucidchart, or draw.io.².

3. Analyze and Prioritize:

- o Categorize requirements.
- Prioritize based on project goals and constraints.

4. Validate and Refine:

- o Review with stakeholders.
- o Ensure clarity and alignment.

0

REQUIREMENTS GATHERING PROCESS



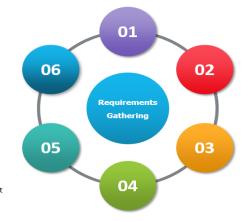
- · Better Management of Projects.
- Confidence in Project Management Capability .

02-Project Sponsors

- Projects Easier to Start-up & Establish
- Better Project Delivery.

03-Top Management

- Better Management of Projects
- Satisfy "Modernizing Government".
- "House Style" For Projects Management



04-Likely Participants

- · Project Management Skills.
- Personal & Career Development.
- Interesting ,Practical , Relevant Training.

05-IT Unit

Fits With Prince

06-H.R Unite

- Good Training.
- Coherence With Other Training.
- Contributes to "Leading Edges" Portfolio.

POWERSLIDES ______ WWW.POWERSLIDES.CO

2., Design

- **Objective**: Create a blueprint for the system.
- Implementation: Develop system architecture, database schema, and user interfaces.
- **Impact**: A well-designed system minimizes risks, enhances scalability, and improves user experience.

1. Objective:

Transform all requirements into detailed specifications covering all aspects of the system.

Plan and assess security risks.

System Design Document: Accurately describes the system design, serving as input for development.

2. Goals:

Transform requirements into complete, detailed system design specifications.

3. Implementation (Coding):

• Objective: Write code based on design specifications.

- **Implementation**: Developers write and test code, integrating various components.
- Impact: Quality code ensures system reliability and maintainability.

Software Development Life Cycle (SDLC) is a critical step where the project begins to take shape. **Objective**:

- o Developers start writing the actual source code for the software.
- o This phase translates the design specifications into a functional product.

2.

- 3. **Key Activities**:
 - o Coding: Writing code based on the design.
 - o **Integration**: Combining different modules and components.
 - **Testing**: Ensuring code correctness and functionality.

0

- 4. Importance:
 - o Implementation brings the project to life.
 - o It's where the actual reorganization happens.

4. Testing:

- Objective: Verify system functionality and identify defects.
- Implementation: Conduct unit, integration, and user acceptance testing.
- **Impact**: Rigorous testing reduces post-deployment issues and enhances user satisfaction.

1. Unit Testing:

Objective: Verify individual components (units) of the software.

Implementation: Developers write test cases for specific functions or modules.

2. Integration Testing:

Objective: Test interactions between different modules.

Implementation: Combine units and test their interactions

3. **System Testing**:

Objective: Validate the entire system against requirements.

: Test end-to-end functionality.

4. Acceptance Testing:

Objective: Verify if the system meets business requirements.

Implementation: Conducted by users or stakeholders.

5. Deployment:

- Objective: Roll out the system to users.
- Implementation: Install software, configure servers, and ensure data migration.
- **Impact**: Successful deployment leads to system availability and user adoption.

1. **Production Installation**:

- o Deploy the software to the production environment.
- o Install, configure, and ensure data migration.

2. Customer Acceptance:

- o Verify successful software execution, completeness, and correctness.
- o Essential for a smooth launch of the software.

6. Maintenance and Support:

- Objective: Ensure ongoing system performance.
- Implementation: Regular updates, bug fixes, and user support.
- **Impact**: Effective maintenance sustains system health and user trust.

Importance of the Maintenance Phase:

- Bug Fixing: Identifying and fixing software defects (bugs) missed during testing or reported by users.
- Enhancements: Adding new features or functionality as user needs evolve.
- Adaptation to Changing Environments: Ensuring compatibility with new hardware, operating systems, and external factors.
- **Performance Optimization**: Monitoring and optimizing software performance.
- Security Updates: Applying patches and addressing vulnerabilities.

Types of Maintenance:

Corrective Maintenance: Fixing bugs discovered after software deployment.

Adaptive Maintenance: Updating software to remain compatible with environmental changes.

Perfective Maintenance: Enhancing software based on user feedback or business needs.

Best Practices:

- **Regular Monitoring**: Continuously monitor software performance.
- Version Control: Maintain different versions for easy rollback.
- Security Audits: Regularly assess security vulnerabilities.
- User Feedback: Actively listen to user feedback for improvements.
- **Documentation**: Keep documentation up-to-date.

•