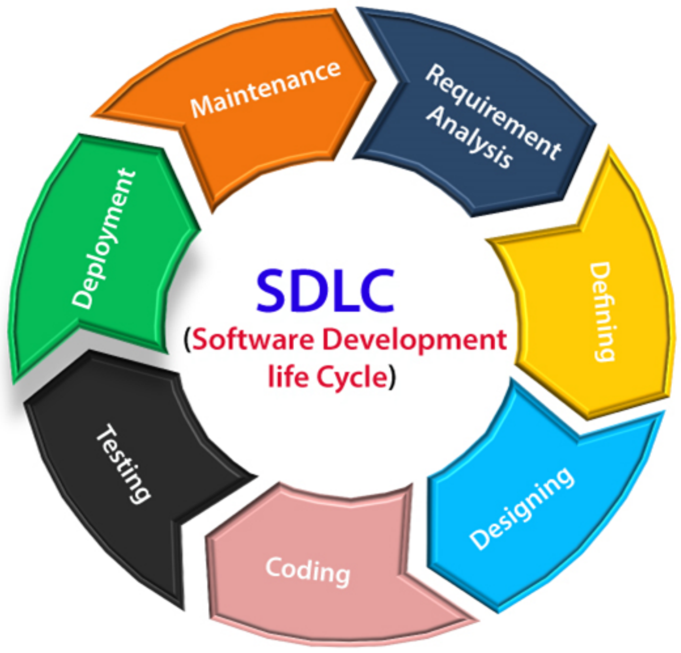
EXPERIMENT 2

Aim -

Case study for SDLC.

Theory -

Software Development Life Cycle (SDLC)
A software life cycle model (process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.
In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities into phases differently. Thus, no element in which the life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.



Stage1: Planning and requirement analysis
Requirement Analysis is the most important and necessary stage in SDLC.
The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identification of the risks associated with the projects is also done at this stage.
Business analysts and Project organizers set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, and what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

Stage2: Defining Requirements
Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted by the project stakeholders.
This is accomplished through the "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software
The next phase is about bringing down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

Stage4: Developing the project
In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing
After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
During this stage, unit testing, integration testing, system testing, and acceptance testing are done.

Stage6: Deployment
Once the software is certified, and no bugs or errors are stated, then it is deployed.
Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
After the software is deployed, then its maintenance begins.

Stage7: Maintenance
Once when the client starts using the developed systems, then the real issues come up and requirements are to be solved from time to time.
This procedure where the care is taken for the developed product is known as maintenance.

What is secure SDLC?
The Software Development Lifecycle (SDLC) is a structured process that enables high-quality software development, at a low cost, in the shortest possible time. Secure SDLC (SSDLC) integrates security into the process, resulting in the security requirements being gathered alongside functional requirements, risk analysis being undertaken during the design phase, and security testing happening in parallel with development, for example.
Importance of secure SDLC-
As Secure Software Development Lifecycle integrates security tightly into all phases of the lifecycle there are benefits throughout the lifecycle, making security everybody's responsibility and enabling software development that is secure from its inception. Some of the biggest benefits are as follows:

- **Reduced Costs**: Thanks to early identification of security concerns allowing the embedding of controls in parallel. No more patching post-deployment.
- **Security-First:** Secure SDLC builds security-focused cultures, creating a working environment where security comes first, and everyone's eyes are on it. Improvements happen across the organization.
- **Development Strategy:** Defining security criteria from the outset improves technology strategy, making all team members aware of the security criteria of the product, and ensuring developer security throughout the lifecycle.
- **Better Security:** Once Secure SDLC processes are embedded, security posture improves across the whole organization. Organizations that are security aware reduce their risk of cyberattack significantly.

Case Study: Software Development Life Cycle (SDLC) Implementation in Healthcare

Introduction:

This case study looks at how a healthcare organization used the Software Development Life Cycle (SDLC) to improve its Electronic Health Records (EHR) system. The hospital in question is a multi-specialty hospital with numerous departments and a large patient load. The organization recognized the need to improve the existing EHR system in order to streamline processes, boost efficiency, and provide better patient care.

Phase 1: Planning and Requirements Gathering

The project was launched by the hospital's management, who assembled a cross-functional team of IT specialists, healthcare professionals, administrators, and project managers. To identify pain points and gather requirements, the team met with stakeholders such as doctors, nurses, administrators, and patients.

Key objectives have been identified:

- Improve the usability of the EHR system by ensuring easy navigation and user-friendly interfaces for medical staff.

- Improve data security and privacy measures to meet HIPAA requirements.
- Improve interoperability to allow for seamless data exchange between departments and outside healthcare providers.
- Improve system performance and reduce downtime for continuous patient care.

Phase 2: System Design and Architecture
During the design phase, the requirements were translated into a comprehensive design plan for the enhanced EHR system. The team concentrated on the following factors:

- UX designers worked with medical staff to create intuitive and efficient interfaces that corresponded to clinical workflows.
- Data security experts devised measures to safeguard sensitive patient data and ensure compliance with data protection standards.
- Interoperability: Technical architects identified data exchange standards and protocols such as HL7 and FHIR.
- Hardware upgrades and redundancy measures were planned by infrastructure experts to minimize system downtime.

Phase 3: Development and Implementation
Coding and programming activities kicked off the development phase. To manage development iterations efficiently, the team used Agile methodologies. To automate testing and deployment processes, they set up a continuous integration and continuous deployment (CI/CD) pipeline.

Important steps to take during this phase:
- Modular components were created by developers to allow for parallel development and easier maintenance.
- To catch and address defects early, regular code reviews and testing cycles were performed.
- To ensure a smooth transition from the old system to the improved EHR, data migration strategies were implemented.

Phase 4: Testing and Quality Assurance
A thorough assessment was carried out to confirm the system's operational, safety, and operational effectiveness. Diverse testing methods, such as unit, integration, system, and user acceptance testing (UAT), were executed.
QA engineers oversaw functional and non-functional assessments to ensure adherence to specifications and benchmarks.
Medical personnel actively engaged in UAT, offering input and pinpointing usability concerns.
A comprehensive security evaluation, encompassing vulnerability checks and penetration testing, was undertaken to safeguard data integrity.

Phase 5: Deployment and Training

Following the successful conclusion of testing and quality assurance, the team readied for the deployment phase. The upgraded EHR system was implemented in gradual stages to mitigate any potential disruptions.

1. Thorough training initiatives were organized for medical personnel, administrators, and other users to facilitate a seamless transition to the updated system.
2. Comprehensive documentation and user manuals were supplied for reference purposes.

Phase 6: Maintenance and Support

Subsequent to deployment, the project transitioned to the maintenance and support phase. A specialized support team was put in place to address any emerging issues or requests.

1. Frequent system updates and patches were introduced to tackle bugs and enhance security measures.
2. Active efforts were made to gather user feedback to pinpoint potential areas for enhancement.

Conclusion -

Implementing the SDLC to enhance the EHR system significantly improved hospital efficiency and patient care. The upgraded system's user-friendliness, data security, and compatibility led to streamlined workflows, reduced errors, and greater patient satisfaction. The healthcare facility continued best practices, prioritizing regular updates to meet evolving industry needs.