Day 40

DIY

DIY Questions:

1.John was looking at specific Software Development Life Cycle (SDLC) models to use in one of his projects. Later, he discovered that the project might require making some changes at some later stage of the software development life cycle. Which one of the SDLC models from his list does he need to rule out from consideration for this?

a)RAD Model b)Waterfall Model c)V-Model d)Agile Model

Ans: **b)Waterfall Model**

2.Mike and his team finally got their first project from a particular client. The project is simple, where requirements for the project are fixed by the clients, and there is no scope for the requirements to get updated in the future. As it is Mike and his team’s first project, he wants to build it in a detailed step-by-step sequential manner, so he finally selected a software development life cycle model that fits his needs. Now Mike and his team have been following the phases of that life cycle model they had selected and have started developing the internal system structure/outline for all specified system requirements. Which phase is currently Mike and his in?

a) Integration Testing Phase b) Architectural Design Phase c) Module Design Phase d) Process Modelling Phase

Ans: **c) Module Design Phase**

3.You are developing software for supply chain management. Your choice of SDLC method is Waterfall. A co-worker decides to switch to the Agile methodology. Why is this switch a mistake need to rule out from consideration for this?

a)Delivering the application in parts renders it useless as the application must have 100% functionality to be approved.b)All necessary and required tasks will be run and tested simultaneously, delaying the product delivery time.c)The lack of thorough testing will lead to the productionof products with lowered levels of quality.d)The initial designs and features might be unnecessary at the time of project completion, which will be considered a monetary loss.

Ans: **a)Delivering the application in parts renders it useless as the application must have 100% functionality to be approved.**

4.You are developing software to map the spread of an epidemic and want to list the nearest healthcare centers and help facilities. The structure and type of assistance required are constantly changing, leading to continuous changes in your software design?What is the ideal SDLC model you must adapt to?

a)Agileb)Iterative c)Waterfalld)Scum

Ans: **b)Iterative**

5.While developing a webpage, the model you choose to develop the required software necessitates the purchase of hardware in the early stages of designing the software. What model have you adopted?

a)Waterfallb)Incrementalc)Kanband)Iterative

Ans: **a)Waterfall**

6.You are developing a web application using the iterative model. The following represent the analysis and design at different stages.Design(0), Analysis(0), Design(1), Analysis(1), Design(2), Analysis(2).Which option accurately represents the flow of the software?

a) Design(0)->Analysis(1)

Design(1)->Analysis(2)

b) Design(1)->Analysis(0)

Design(2)->Analysis(1)

c) Analysis(0)->Design(1)

Analysis(1)->Design(2)

d) Analysis(0)>Design(0)

Analysis(1)->Design(1)

Ans: **c) Analysis(0)->Design(1)**

**Analysis(1)->Design(2)**

7.What is a Feasibility Study?

Ans:

a feasibility study refers to the initial phase of assessing the practicality and viability of a software development project. It is conducted before the project officially begins and aims to determine if the project is worth pursuing further based on various factors.

The feasibility study in SDLC typically examines three key aspects:

Technical Feasibility: This aspect evaluates whether the project can be successfully implemented from a technical perspective. It assesses factors such as the availability of required technology, infrastructure, software, hardware, and development tools. The study determines if the necessary resources and expertise are available to carry out the project and achieve its objectives.

Economic Feasibility: The economic feasibility analysis focuses on assessing the financial viability of the software development project. It involves estimating the project's costs and potential benefits, such as increased efficiency, cost savings, revenue generation, or competitive advantage. This analysis helps stakeholders determine if the expected benefits outweigh the project's costs and if the return on investment (ROI) is favorable.

Operational Feasibility: The operational feasibility aspect examines whether the software project aligns with the organization's existing systems, processes, and resources. It considers factors such as compatibility, integration, and impact on day-to-day operations. The study assesses whether the software can be implemented and adopted smoothly within the organization, taking into account any necessary training, support, or changes in workflows.

The feasibility study in SDLC serves as a decision-making tool, enabling stakeholders to make informed choices about proceeding with the project. Based on the findings of the study, stakeholders can decide to continue with the project, modify its scope, or even cancel it if the study reveals significant risks or infeasibility.

The feasibility study report in SDLC documents the analysis, conclusions, and recommendations derived from the assessment. It provides a solid foundation for subsequent phases of the SDLC, such as requirements gathering, design, development, testing, and implementation, by ensuring that the project is well-founded and aligns with the organization's goals and resources.

8.Explain Level Design and Low-Level Design?

Ans:

In the context of software development, Level Design and Low-Level Design are two distinct stages within the Software Development Life Cycle (SDLC). They focus on different aspects of the design process and serve different purposes. Let's delve into each of them:

Level Design:

Level Design refers to the high-level conceptualization and planning of a software system. It involves defining the overall structure, components, and interactions of the system at a broad level. The main objectives of level design include understanding the requirements, identifying major system components, and establishing the relationships between these components. The outcome of the level design stage is a blueprint or architectural design that serves as a guide for the subsequent phases of development.

Key activities in Level Design typically include:

Requirement gathering and analysis: Understanding the functional and non-functional requirements of the system.

System architecture design: Identifying the major components, their responsibilities, and the interactions between them.

Data design: Designing the structure and organization of the data that the system will handle.

Interface design: Defining the interfaces between different system components and external systems or users.

Technology selection: Choosing the appropriate technologies, frameworks, and platforms to build the system.

High-level planning: Estimating the overall timeline, resources, and risks associated with the project.

Low-Level Design:

Low-Level Design (LLD) is a subsequent stage that follows the level design phase. It focuses on translating the high-level design into detailed specifications that can be used for implementation. In low-level design, the system's components are broken down into smaller modules or units, and the internal structure and behavior of each module are defined. LLD provides a granular understanding of how each component will be implemented, including algorithms, data structures, and interfaces.

Key activities in Low-Level Design typically include:

Module decomposition: Breaking down the system into smaller modules or units.

Algorithm design: Defining the algorithms and logic for each module, specifying how they will achieve their intended functionality.

Data structure design: Designing the data structures required for module implementation, including variables, arrays, lists, etc.

Interface specification: Defining the inputs, outputs, and communication protocols for each module, including function signatures, parameters, and return values.

Detailed planning: Estimating resource requirements, determining dependencies between modules, and creating a detailed implementation plan.

Overall, Level Design focuses on the broader structure and architecture of the system, while Low-Level Design delves into the specifics of each module or unit. Both stages are crucial for successful software development as they provide a clear roadmap for implementation and ensure that the final product meets the desired requirements.