Roll no: 1

Student name: Nitesh Mulam

Email id: niteshmulam30@gmail.com

Q1 a: Compare FTR and Walkthrough

A Formal Technical Review (FTR) is a structured software quality control activity. conducted by software engineers to uncover errors in function, logic, or implementation. It is characterized by a formal meeting with a specific agenda, a review leader, and a recorder who produces a formal summary report. In contrast, a Walkthrough is an informal peer review process where the author of the work product guides the team through the logic or code to share knowledge and identify bugs early. While FTRs focus on strict adherence to standards, Walkthroughs are often used for early stage brainstorming.

Q1 c: Explain the LOC

Lines of Code (LOC) is a direct measure of software size used in software engineering to

estimate project effort, cost, and productivity.

As a size-oriented metric, it is calculated by counting the total number of lines in a delivered program, typically excluding comments and blank lines. While LOC is easy to calculate and automate, it is highly dependent on the programming language used and can penalize shorter, more efficient code. It is often used as an input for estimation models like Cocomo to predict the man months required for development

Q2 b: Explain the different techniques in white box testing.

1) White box testing, also known as structural testing, examines the internal logic and structure of the code.

2) Control Flow Testing is a strategy that uses the program's control flow as a model to design test cases for every decision point.

3) Basis Path Testing allows the test case designer to derive a logical complexity measure of a procedural design for defining execution

paths.

4) Cyclomatic Complexity provides a quantitative measure of the logical complexity of a program, defining the number of independent paths.

5) Condition Testing is a test case design method that exercises the logical conditions contained in a program module.

6) Data Flow Testing selects test paths of a program according to the locations of definitions and uses of variables.

Q2 c: Explain steps in version and change control.

Baseline Creation: Establishing a reference point in the software lifecycle that is formally reviewed and agreed upon.

Change Request: The process begins when a stakeholder submits a formal request for a modification to a configuration item.

Impact Analysis: Evaluating the request to determine how the change will affect the software, schedule, and cost

Change Review Board (CRB): An authority that reviews the request and decides whether to approve, deny, or defer the change.

Check-out: Developers take code from a central repository to prevent others from editing the same version simultaneously.

Modification: The developer implements the approved change in a controlled local environment.

Quality Audit: Testing and reviewing the modified code to ensure it meets requirements without introducing new ideas

Q3 b: Explain cohesion and Coupling. Explain different types with detailed example

Core Principle: Effective modular design aims for High Cohesion and Low Coupling.

Cohesion Definition: A measure of the functional strength of a module-how closely elements within it relate.

Functional Cohesion: The highest level, where a module performs exactly one task, like

"Calculate Square Root".

Sequential Cohesion: Elements are grouped because the output of one process input to another.

Temporal Cohesion: Elements grouped together only because they are performed at the same time.

Coupling Definition: A measure of the degree of interdependence between two separate modules.

Data Coupling: The best form, where modules communicate by passing simple data pieces as parameters.

Control Coupling: Occurs when one module directs the logic of another by passing "flags".

Common Coupling: Two modules share the same global data area, which can lead to integrity issues.

Content Coupling: The worst form, where one module directly modifies internal data.

Q3 c: Explain the Spiral model of software development.

The Spiral Model is an evolutionary software process model that is highly effective for large-scale, complex, and high-risk projects. It combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

Risk Driven Approach: The primary distinction of the Spiral model is that it uses risk analysis to guide the development process at every stage..

* Quadrant I (Planning): This phase involves the determination of objectives, alternatives, and constraints for the current phase of the project.

* Quadrant 2 (Risk Analysis): Technical and management risks are identified and resolved in this quadrant, often through the use of prototyping or simulation.

* Quadrant 3 (Engineering): This stage involves the actual development of the next level product, including activities such as coding,

testing, and integration.

* Quadrant 4 (Evaluation): This phase involves customer assessment of the work product and planning for the next "circuit" or iteration of the spiral.

*Iterative Evolution: The project moves through these four quadrants in multiple loops, with the cumulative cost increasing as the spiral progresses through each iteration.

Q4 a: Explain the general format of SRS for Hospital Management system

Introduction: Defines the purpose, scope, and specific objectives of the HMS.

General Description: Overview of product perspective and user classes like Doctors and Patients.

Functional Requirements - Registration:

Details for patient admission and electronic record management. Functional Requirements

Appointments: Logic for scheduling, canceling, and tracking consultations.

Functional Requirements - Billing:

Requirements for processing payments and insurance claims.

Non-Functional Requirements

Security: Ensuring patient data privacy through role-based access.

Non-Functional Requirements

Availability: Requirement for the system to be accessible 24/7.

External Interface Requirements: Interaction with hardware like scanners and other software APels.

Database Requirements: Description of the data schema needed to store medical histories and records.

Appendices: Includes a glossary of terms and specific diagrams like Fbs for workflow.

Q4 b: Explain software Re-engineering in detail.

Definition: Analyzing and attering a system to reconstitute it in a new form to improve maintainability.

Inventory Analysis: Assessing the portfolio to

identify candidates for re-engineering based on business valve.

Document Restructuring: Updating documentation for a legacy system to reflect its current state.

Reverse Engineering: Analyzing a program to identify components and interrelationships for high-level abstractions.

Code Restructuring: Modifying source code to make it more readable without changing functionality.

Data Restructuring: Redesigning data structures or schemas to improve performance or security.

Forward Engineering: Using knowledge from reverse engineering to build the system with modern tech.