/10/23

[illegible]
9606 comps B

SE assignment II

Q.1) Risk assessment in the context of software engineering projects is the process of identifying, analyzing, & evaluating potential risks & uncertainties that could impact the successful completion or outcome of a software development project.

This include identifying potential threats, vulnerabilities & uncertainties related to project requirements, resources, technology, stakeholders & external factors:

Risk assessment is essential in software projects as follows:-

1) Proactive Problem Identification:- Risk assessment allows project teams to identify potential issues & challenges early in the project lifestyle cycle. This approach enables teams to take steps to mitigate or address these risks before they escalate into more significant problems

2) Resource Allocation: It helps in allocating resources (time, budget & personnel) appropriately. By identifying potential risks, project managers can allocate resources to address or mitigate these risks, ensuring that project remains on track

3) Prioritization of Efforts:- Not all risks are of equal importance. Risk assessment helps in prioritizing efforts to focus on the most critical & impactful risks. This ensures that resources are utilized efficiently

4) Stakeholders Communication & Expectations: It enables effective communication with stakeholders about potential challenges & uncertainties. This transparency builds trust & helps manage expectations regarding project timeline

5) Quality Assurance:- Some risks may be related to the quality of software being developed. By identifying & addressing these risks, the project team can ensure that final product meets the required quality standards

(Q.2) Software Configuration Management (SCM) is a set of processes, tools, & techniques used to manage & control changes in a software project, its primary goal is to maintain the integrity & consistency of software products throughout their development life cycle

1) Version Control:- SCM helps track & manage different versions of software artifacts (code, documents, etc) This ensures that developers are working on the correct & latest versions, reducing the risks of errors or inconsistencies

2) Change Management:- It provides a structured approach to handle changes in software components. This prevents unauthorized or unplanned modifications, which can lead to bugs or system failures

3) Configuration Identification:- SCM defines what constitutes a configuration item (CI). This includes all elements that make up the software, such as source code, documentation, libraries & configuration files. This clarity ensures that all necessary components are accounted for and properly managed.

4) Build & Release Management:- SCM oversees the process of creating builds from source code. It ensures that builds are reproducible, consistent, & properly documented. This helps in reliably delivering software to various environment.

5) Build & Release Dependency Management: SCM keeps track of dependencies btwn different software components. This ensures that changes to one component do not break functionality in dependent components.

6) Risk Mitigation: By enforcing processes for code review, testing & documentation, SCM helps mitigate the risks of introducing defects or inconsistencies into software.

7) Regulatory Compliance: For projects subject to regulatory requirements (such as medical or financial software), SCM helps in ensuring that processes are followed, changes are documented & audits can be conducted as needed.

8) Formal Technical Reviews (FTR) plays a crucial role in assuring software quality & reliability. They do so by employing a structured evaluation process, offering the following benefits:

1) Early Error Identification & Resolution:- FTR meticulously examines software artifacts, pinpointing defects & inconsistencies in the initial stages, preventing them from permeating the final product.

(2) Enhanced Requirement Understanding: Through FTR, stakeholders gain a deeper grasp of requirements, minimizing misinterpretation & subsequent implementation errors.

(3) Adherence of Standards: FTR ensures compliance with established norms & practices, maintaining consistency & adherence to organizational or industry standards.

(4) Proactive Risk Management: By scrutinizing artifacts, FTR detects & addresses potential risks early, averting critical issues later in the project.

(5) Requirement Traceability: FTR establishes a clear link to requirements & development artifacts, ensuring comprehensive coverage & functionality alignment.

(6) Validation of Design Choices: FTR serves as a platform for validating design decisions, confirming the soundness of chosen architectural & design approaches.

(7) Uniformity & Consistency: FTR enforces coding conventions, naming conventions & style guidelines, promoting a uniform & cohesive codebase.

(A) Formal walkthrough for a software project involves a systematic & structured review of project artifacts. Here is process for conducting a formal walkthrough

(1) Preparation & planning:

(a) Select Participants: Identify the key stakeholders who should be part of the walkthrough, including developers, designers, testers & relevant subject matter experts.

(b) Schedule the walkthrough: Set a date & time for the walkthrough, ensuring that all essential participants can attend

a. Presenter's Introduction: The designated presenter (often the developer or designer) provides a brief overview of the artifact being reviewed, its purpose, and any key design or implementation decisions.

b. Walkthrough of Artifacts: The presenter guides the participants through the content, explaining the design choices, code structure, or any relevant details. They may highlight critical areas and discuss how they contribute to meeting the project requirements.

Participant Engagement:

a. Ask Questions: Participants are encouraged to ask questions, seek clarifications, and provide feedback on the presented material.

b. Discuss Design Choices: Engage in discussions regarding design decisions, trade-offs, and potential improvements. Evaluate if the design aligns with project objectives and requirements.

c. Identify Issues: Participants should actively look for defects, inconsistencies, or

c. Share Materials in Advance: Distribute the relevant documents or artifacts (e.g., code, design documents, requirements) to participants ahead of time, allowing them to review and prepare.

d. Define Objectives and Scope: Clearly articulate the goals and scope of the walkthrough, specifying what aspects of the project will be reviewed.

Introduction and Overview:

a. Opening Remarks: Start the walkthrough with an introduction, where the moderator outlines the purpose of the session, its goals and the expected outcomes.

b. Review Objectives and Scope: Reiterate the specific objectives and scope of the walkthrough to ensure everyone is aligned.

Presentation by Presenter:

a. Presenter's Introduction: The designated presenter (often the developer or designer) provides a brief overview of the artifact

Considering software reliability is crucial when analyzing potential risks in a project for several reasons:

1) User Confidence and Trust: Reliable software instills confidence in users. They are more likely to trust and continue using a system that consistently performs as expected.

2) Business Reputation: Software failures or frequent glitches can damage a company's reputation. Reliability issues can lead to negative reviews, customer dissatisfaction, and potential loss of business.

3 financial Implications: Unreliable software can lead to financial losses due to downtime, lost sales, and potential legal or contractual penalties for failing to meet service level agreements (SLAs).

4. Compliance and Legal Consequences: In certain industries, like healthcare or finance, software reliability is critical for meeting regulatory compliance. Failing to comply with

6. Maintenance Costs: Unreliable software often requires more frequent maintenance and support. This can lead to higher operational costs and divert resources from other critical tasks.

7. User Experience and Satisfaction: Reliability directly impacts user experience. Frequent crashes, errors, or slow performance can frustrate users and lead to reduced satisfaction.