POST LAB QUESTIONS FOR SOFTWARE ENGINEERING LAB FOR AY23-24 :

By. Dr. B.S. Daga

1. **Software Requirement Specification:**

   a) Evaluate the importance of a well-defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success. b) Analyse a given SRS document to identify any ambiguities or inconsistencies and propose improvements to enhance its clarity and completeness. c) Compare and contrast different techniques for requirement elicitation, such as interviews, surveys, and use case modelling, and determine their effectiveness in gathering user needs.

2. **Implementing Project using SCRUM method on JIRA Tool**:

a) Assess the effectiveness of the Scrum framework for managing software development projects compared to traditional project management methodologies. b) Analyse a Sprint Backlog in JIRA and identify any potential bottlenecks or issues that might hinder the team's progress during the sprint. c) Evaluate the role of the Scrum Master in handling conflicts within the development team and resolving impediments to maintain a smooth project flow.

3. **Implementing Project using KANBAN method on JIRA Tool**:

a) Compare and contrast the Kanban and Scrum methodologies in terms of flexibility, adaptability, and workflow management in different project scenarios. b) Analyse a Kanban board in JIRA and propose improvements to

optimize the team's efficiency and productivity. c) Evaluate the impact of Work In Progress (WIP) limits on a Kanban board and how it affects the team's throughput and cycle time.

4. **Calculating function points of the Project:**

a) Critically evaluate the Function Point Analysis method as a technique for software sizing and estimation, discussing its strengths and weaknesses. b) Apply the Function Point Analysis technique to a given software project and determine the function points based on complexity and functionalities. c) Propose strategies to manage and mitigate uncertainties in function point estimation and how they can impact project planning and resource allocation.

5. **Estimating project cost using COCOMO Model:**

a) Analyse the COCOMO model and its different modes (Organic, Semi-detached, Embedded) to determine the most suitable mode for a specific project type. b) Apply the COCOMO model to estimate the project cost and effort required for a given software development project. c) Evaluate the factors influencing COCOMO estimates, such as project size, personnel capabilities, and development tools, and their implications on project planning and scheduling.

6. **Data flow analysis of the Project:**

a) Evaluate the benefits of using Data Flow Diagrams (DFD) to analyze and visualize the data movement in a complex software system. b) Apply

data flow analysis techniques to a given project and identify potential data bottlenecks and security vulnerabilities. c) Propose improvements to the data flow architecture to enhance the system's efficiency and reduce potential risks.

7. **Design using Object-Oriented approach with emphasis on Cohesion and Coupling:**

a) Analyze a given software design and assess the level of cohesion and coupling, identifying potential areas for improvement. b) Apply Object-Oriented principles, such as encapsulation and inheritance, to design a class hierarchy for a specific problem domain. c) Evaluate the impact of cohesion and coupling on software maintenance, extensibility, and reusability in a real-world project scenario.

8. **Design test cases for performing black box testing:**

a) Create a set of black box test cases based on a given set of functional requirements, ensuring adequate coverage of different scenarios and boundary conditions. b) Evaluate the effectiveness of black box testing in uncovering defects and validating the software's functionality, comparing it with other testing techniques. c) Assess the challenges and limitations of black box testing in ensuring complete test coverage and discuss strategies to overcome them.

9. **Design test cases for performing white box testing:**

a) Generate white box test cases to achieve 100% statement coverage for a given code snippet. b) Compare and contrast white box testing with black

box testing, highlighting their respective strengths and weaknesses in different testing scenarios. c) Analyze the impact of white box testing on software quality, identifying its potential to uncover complex logic errors and security vulnerabilities.

10. **Version controlling & Risk Analysis of the project:**

a) Assess the importance of version control in collaborative software development and the benefits it offers in managing code changes and collaboration. b) Conduct a risk analysis of a software development project, identifying potential risks and their impact on project delivery and quality. c) Evaluate the role of version control in risk mitigation and how it enables better project management and collaboration among team members.