Course Name – Software Engineering
Course Code – BCA50114

A. Multiple Choice Type Questions

1. What is software?

a) A set of related web pages
b) A set of instructions given to computer
c) An electronic device
d) A physical component of computer

2. Which of the following is not a type of software?

a) System Software
b) Application Software
c) Embedded Software
d) Cable Software

3. What is the main characteristic of good software?

a) High cost
b) Low performance
c) Efficiency
d) Frequent errors

4. Which term defines the process of developing software in a systematic way?

a) Web Engineering
b) Software Engineering
c) Hardware Engineering
d) Programming

5. Which of the following is an attribute of good software?

a) Inflexibility
b) Maintainability
c) Complexity
d) Redundancy

6. What is a key challenge faced by software engineers?

a) Hardware design
b) System assembly
c) Coping with legacy systems
d) Wiring circuits

7. What type of software supports the running of application software?

a) System Software
b) Utility Software
c) Compiler
d) Application Software

8. Which of these is not an example of system software?

a) Operating System
b) Compiler
c) Word Processor
d) Device Driver

9. What is software cost usually associated with?

a) Hardware resources
b) Documentation only
c) Development and maintenance
d) Marketing

10. Who is known as the father of software engineering?

a) Niklaus Wirth
b) Alan Turing
c) Watts Humphrey
d) Barry Boehm

11. Software that controls and manages the hardware components is called?

a) Compiler
b) System Software
c) Utility Software
d) Application Software

12. Which software attribute refers to how well the software works under stated conditions?

a) Portability
b) Reliability
c) Flexibility
d) Efficiency

13. Which of the following is a goal of software engineering?

a) Creating programs without planning
b) Increased cost of development
c) Producing reliable and efficient software
d) Avoiding software documentation

14. The process of defining, developing and maintaining software is called?

a) Software Engineering
b) Programming
c) Debugging
d) Testing

15. Which is not a characteristic of software?

a) Engineered
b) Does not wear out
c) Manufactured
d) Custom-built

16. Which of these is a key activity in software engineering?

a) Hardware assembly
b) Algorithmic soldering
c) Requirement Analysis
d) Circuit wiring

17. Software with minimal human interaction for its functions is called?

a) Manual Software
b) Automated Software
c) Dynamic Software
d) Interactive Software

18. Systems engineering involves?

a) Only hardware
b) Only software
c) Both hardware and software
d) Circuit design

19. What do we call the software that is installed on the system to support other software?

a) System Software
b) Utility Software
c) Operating System
d) Driver Software

20. Which is a software attribute related to time and resource usage?

a) Usability
b) Efficiency
c) Portability
d) Maintainability

21. Identify the reason software engineering is considered essential in modern development practices.

a) To understand hardware designs
b) To reduce software costs and improve quality
c) To build mobile phones
d) To create game hardware

22. Interpret how software engineering principles contribute to managing large and complex projects.

a) It helps in structured planning and tracking
b) It increases errors
c) It decreases communication
d) It adds cost

23. Recognize the challenge in software engineering that is specifically related to scalability issues.

a) Handling large and growing systems
b) Choosing fonts
c) Using outdated hardware
d) Programming in BASIC

24. Infer the significance of systems engineering knowledge in the software development lifecycle.

a) It integrates software with other system components
b) It replaces hardware design
c) It eliminates testing
d) It builds GUIs

25. Classify the given example as system software.

a) Operating System
b) MS Word
c) Tally
d) Photoshop

26. Predict why software does not physically deteriorate like hardware over time.

a) It has no physical components
b) It is biodegradable
c) It is electronic
d) It is fixed

27. Identify how documentation contributes to the maintainability of a software system.

a) Helps in maintenance and future development
b) Reduces file size
c) Increases loading time
d) Improves graphic resolution

28. Indicate the benefit of modularity in improving software design.

a) It supports independent development and maintenance
b) It complicates design
c) It reduces usability
d) It increases cost

29. Select the reason why efficiency is a critical factor in evaluating software quality.

a) It affects system performance
b) It makes software colorful
c) It increases size
d) It prevents viruses

30. Recognize which attribute of software contributes to its continued correct operation over time.

a) Reliability
b) Flexibility
c) Usability
d) Compatibility

31. Infer why accurate cost estimation remains a persistent issue in software projects.

a) Many variables and unpredictable changes
b) Software is cheap
c) All tools are free
d) Hardware is complex

32. Classify the type of software that provides the foundation for application programs to run.

a) System Software
b) Multimedia Software
c) Utility Software
d) Embedded Software

33. Review the components typically included in software engineering cost calculations.

a) Development and maintenance
b) Machine cost
c) Paper cost
d) Cabling cost

34. Interpret the importance of capturing user requirements during software development.

a) To develop appropriate and usable software
b) To increase prices
c) To confuse users
d) To avoid UI design

35. Indicate how abstraction simplifies the complexity of software design.

a) It hides complexity
b) It shows all errors
c) It increases coupling
d) It merges systems

36. Identify how software differs from hardware in terms of functionality and behavior.

a) Software is logical and intangible
b) Software is made of silicon
c) Software requires cooling
d) Software is built with resistors

37. Identify the main focus of system engineering in the context of software development.

a) Integration of hardware and software
b) Hardware design only
c) GUI testing
d) Font design

38. Interpret what the \'key challenge\' in software engineering typically refers to.

a) Managing complexity, cost, and quality
b) Creating animation
c) Game testing
d) Wiring circuits

39. Infer why understanding the characteristics of software is important for developers.

a) To choose the best development approach
b) To write poems
c) To understand keyboard layout
d) To fix printers

40. Recognize why software systems are considered \'engineered\' rather than just coded.

a) They follow systematic development processes
b) They are always physical
c) They are manually operated
d) They run without planning

41. Interpret the primary purpose of the Waterfall model in structured development.

a) To write code without planning
b) To implement rapid changes
c) To follow a sequential development process
d) To develop in parallel phases

42. Identify a core characteristic that defines Agile methods.

a) Emphasis on documentation
b) Iterative development with customer feedback
c) Linear process flow
d) Requires CASE tools

43. Recognize the concept of process iteration within software process models.

a) Repeating a process until hardware is complete
b) Delivering the product without review
c) Repeating phases to improve the product
d) Skipping steps in development

44. Classify the model that integrates design and prototyping through multiple stages.

a) Waterfall Model
b) Incremental Model
c) Spiral Model
d) RAD Model

45. Interpret the main goal of Rapid Application Development (RAD) methodology.

a) To build software without user input
b) To develop software in long cycles
c) To quickly develop prototypes and gather user feedback
d) To replace Agile methodology

46. Infer the benefit of using component-based software engineering in modern projects.

a) Reduces cost and improves maintainability
b) Slows down development
c) Increases coupling
d) Avoids reuse of modules

47. Recognize how CASE tools contribute to different phases of software development.

a) Physical debugging
b) Managing hardware
c) Supporting software process activities
d) Replacing project managers

48. Compare Waterfall and Agile models with respect to flexibility.

a) Both have similar flexibility
b) Agile is more flexible than Waterfall
c) Waterfall allows more changes
d) Agile avoids user feedback

49. Identify the primary objective of Extreme Programming (XP) practices.

a) To follow rigid planning
b) To eliminate customer interaction
c) To improve software quality through frequent releases
d) To delay development process

50. Paraphrase the idea behind incremental delivery in iterative development models.

a) Delivering software at once
b) Breaking system into functional increments for delivery
c) Delaying product testing
d) Avoiding integration

51. Interpret the primary purpose of the Waterfall model in sequential process management.

a) To write code without planning
b) To implement rapid changes
c) To follow a sequential development process
d) To develop in parallel phases

52. Identify a distinguishing characteristic of Agile methods in handling requirements.

a) Emphasis on documentation
b) Iterative development with customer feedback
c) Linear process flow
d) Requires CASE tools

53. Recognize how iteration plays a role in refining software during development.

a) Repeating a process until hardware is complete
b) Delivering the product without review
c) Repeating phases to improve the product
d) Skipping steps in development

54. Classify the hybrid model that merges design with prototyping stages.

a) Waterfall Model
b) Incremental Model
c) Spiral Model
d) RAD Model

55. Interpret how RAD emphasizes quick development and frequent user feedback.

a) To build software without user input
b) To develop software in long cycles
c) To quickly develop prototypes and gather user feedback
d) To replace Agile methodology

56. Infer the benefit of component-based software engineering in modular development.

a) Reduces cost and improves maintainability
b) Slows down development
c) Increases coupling
d) Avoids reuse of modules

57. Illustrate how CASE tools support various phases of software development.

a) Physical debugging
b) Managing hardware
c) Supporting software process activities
d) Replacing project managers

58. Compare the flexibility between the Waterfall model and Agile methodologies.

a) Both have similar flexibility
b) Agile is more flexible than Waterfall
c) Waterfall allows more changes
d) Agile avoids user feedback

59. Identify the primary goal of Extreme Programming (XP) in software projects.

a) To follow rigid planning
b) To eliminate customer interaction
c) To improve software quality through frequent releases
d) To delay development process

60. Paraphrase the concept of incremental delivery within iterative models.

a) Delivering software at once
b) Breaking system into functional increments for delivery
c) Delaying product testing
d) Avoiding integration

61. Identify the Agile principle that supports adapting to changing customer requirements.

a) Ignoring changes late in development
b) Following strict plans
c) Welcoming changing requirements
d) Delivering all software at once

62. Choose the most suitable process model for a project with evolving or unclear requirements.

a) Waterfall Model

b) Spiral Model

c) V-Model

d) Structured Programming

63. Use CASE tools during requirement analysis to enhance which aspect of the development process.

a) Automatic hardware updates

b) Automated software process documentation

c) Manual coding only

d) No interaction with users

64. Identify the step involved in managing development risks in Spiral Model.

a) Skipping testing

b) Early risk analysis and prototyping

c) Late integration

d) One-time delivery

65. Select the most effective approach to reduce development time in GUI-based applications.

a) V-Model

b) RAD Model

c) Waterfall Model

d) Formal Methods

66. Apply the concept of non-functional requirements to specify system performance criteria.

a) System usability

b) Response time limits

c) Module cohesion

d) Code reuse metrics

67. Demonstrate how domain requirements influence system functionality in a banking application.

a) Financial regulations

b) Loan processing rules

c) Network latency

d) Interest rate tables

68. Use stakeholder viewpoints to identify conflicting user requirements in large-scale systems.

a) User personas

b) Project scope

c) User interface layout

d) Error logs

69. Apply interview techniques to elicit software requirements from non-technical users.

a) Surveys   b) Observation
c) Interviews   d) Surveys

70. Implement use-case modeling to describe user interaction in a hotel booking system.

a) Flowcharts   b) Use-case diagrams
c) Gantt charts   d) Component diagrams

71. Choose the appropriate scenario-based technique to clarify ambiguous requirements.

a) Interview transcripts   b) Use-case modeling
c) Storyboards   d) Regression testing

72. Apply a logical DFD to model the flow of information in an inventory management system.

a) Data storage structure   b) Information flow between processes
c) Programming logic   d) Hardware interfacing

73. Use a physical DFD to illustrate system components and their interactions with hardware.

a) Data encryption techniques   b) Hardware layout
c) System architecture   d) Cloud infrastructure

74. Implement an ER diagram to define entities and relationships in a student record system.

a) User feedback loops   b) Entity-relationship diagram
c) Function point analysis   d) UML activity diagrams

75. Apply a data dictionary to define attributes associated with system data flows.

a) System availability   b) Data flow frequency
c) Data dictionary definitions   d) Data replication tools

76. Use requirement validation to identify ambiguous or incomplete user statements.

a) User preferences   b) Ambiguous user needs
c) Software lifecycle   d) Code indentation

77. Implement requirement specification to document constraints in a warehouse control system.

a) Design prototypes
b) Access control lists
c) Test scripts
d) Deployment models

78. Apply the standard SRS format to organize requirements for a mobile banking app.

a) User interface design
b) Stakeholder analysis
c) Cost estimation models
d) Release planning

79. Use feasibility analysis to assess whether a proposed software solution meets budget constraints.

a) Database normalization
b) Market segmentation
c) Budget tracking
d) Functional decomposition

80. Apply process modeling techniques to represent workflow in a medical appointment system.

a) Team composition
b) Task sequencing
c) DFD partitioning
d) API management

81. Identify a correct example of functional requirements to an e-commerce system.

a) The system should load within 2 seconds
b) The system must allow users to add items to a cart
c) The interface should be attractive
d) The system should use less memory

82. Demonstrate how use-cases are utilized in the process of requirement analysis.

a) To represent data structures
b) To validate performance metrics
c) To capture user interactions with the system
d) To build class diagrams

83. Identify domain requirement analysis techniques in the context of a banking application.

a) Defining GUI elements
b) Recording withdrawal limits for accounts
c) Listing programming languages
d) Designing login buttons

84. Use a logical DFD to model an online ticket booking system and determine what it represents.

a) Hardware specifications
b) Physical storage media
c) Data flow between processes and data stores
d) Computer network setup

85. Choose the SRS structure to properly document user login requirements.

a) Include hardware manuals
b) Define user interface and authentication logic
c) Write compiler code
d) Document user complaints

86. Identify the concept of requirement elicitation through structured interviews.

a) Use fixed templates
b) Write pseudocode
c) Gather needs directly from stakeholders
d) Create test cases

87. Choose the most suitable tool for documenting data structures in a software project.

a) Data Dictionary
b) Flowchart
c) Pseudocode
d) Wireframe

88. Implement requirement validation to detect inconsistencies or conflicts in the specification.

a) Hardware type
b) Ambiguities and inconsistencies
c) Color scheme
d) Printer support

89. Select the appropriate application of a physical DFD in system analysis.

a) Describes physical files and devices
b) Describes relationships in ER diagram
c) Describes program logic
d) Describes color themes

90. Identify the first step involved in process modeling to a hospital management system and.

a) Test the database
b) Draw a context-level DFD
c) Create GUI
d) Buy a server

91. Use an ER diagram during requirement analysis and determine what the relationships represent.

a) Interfaces
b) Data flow
c) Associations between entities
d) Programming logic

92. Implement requirement specification for a library management system by identifying essential elements.

a) Design of shelves
b) Bookshelf material
c) Borrowing and returning rules
d) Page colors

93. Choose a relevant feasibility consideration during the requirement analysis phase.

a) UI theme selection
b) Evaluating budget and technology constraints
c) Final testing
d) Compiler debugging

94. recognize a valid functional requirement of an ATM system.

a) The system must respond quickly
b) The system must allow cash withdrawal
c) The system should be green
d) The interface should be appealing

95. Select an appropriate method for gathering software requirements from stakeholders.

a) Painting
b) Interview
c) Sketching
d) Zooming

96. Apply a data dictionary to monitor system elements and identify what it describes.

a) Layout colors
b) Structure and meaning of data items
c) Background music
d) Access times

97. Identify a valid use-case of a student management system and.

a) Manage library books
b) Register new student
c) Display background
d) Set password rules

98. Demonstrate how scenarios assist in eliciting detailed software requirements.

a) Design interface
b) Write test cases
c) Describe sequences of user-system interactions
d) Select fonts

99. Apply the principle of functional independence when documenting software requirements.

a) Combine all modules
b) Minimize interactions among modules

c) Repeat logic
d) Write single-line code

100. Use a requirement specification format for a mobile application and identify the section describing expected user actions.

a) Design Constraints
b) Functional Requirements
c) Glossary
d) References

101. identify the aspects typically analyzed white-box testing on a code module.

a) UI design
b) Code logic and paths
c) Screen resolution
d) Software license

102. Choose the appropriate method used to conduct validation testing in software systems.

a) Check if the software meets the customer requirements
b) Verify internal functions of the code
c) Install hardware
d) Check user feedback design

103. Demonstrate unit testing by identifying the specific components it targets.

a) Complete application
b) Subsystem interaction
c) Individual components or functions
d) Database performance

104. Apply equivalence partitioning to a test scenario and determine its primary goal.

a) To reduce the number of test cases
b) To eliminate all testing
c) To ignore edge cases
d) To increase data entry

105. Choose a suitable black-box testing technique for requirement-based testing.

a) Statement coverage
b) Path testing
c) Boundary value analysis
d) Branch testing

106. Use regression testing in a situation where existing features might be affected by new changes.

a) Testing new unrelated features
b) After fixing bugs or making changes
c) Before code compilation
d) To test documentation

107. Apply verification in software development and identify what gets verified in this process.

a) Meeting user expectations

b) Checking compliance with specifications

c) Performance speed

d) Graphics rendering

108. Choose the correct context in which alpha testing is typically performed.

a) Performed by users at their site

b) Performed by internal developers before release

c) Used for security testing only

d) Testing only for GUIs

109. Demonstrate how the Capability Maturity Model (CMM) supports software quality improvement.

a) Set quality goals for hardware

b) Define maturity levels in process improvement

c) Create marketing content

d) Build operating systems

110. Apply ISO 9000 principles to software development and recognize what standard they ensure.

a) Color coding standards

b) Product quality management system

c) Internet connectivity

d) Hardware selection

111. Choose the testing type that evaluates how the system interacts with external components.

a) Integration Testing

b) Unit Testing

c) Acceptance Testing

d) System Testing

112. Demonstrate boundary value testing for input values ranging from 1 to 100 and identify a valid test input.

a) 0

b) 1

c) 150

d) 200

113. Use black-box testing on a login form and determine what should be tested.

a) Internal code structure

b) Browser compatibility

c) Expected input-output behavior

d) RAM usage

114. Choose the testing technique that is most effective for achieving function coverage.

a) Path testing

b) Performance testing

c) Stress testing

d) Usability testing

115. Apply unit testing and identify the types of errors it is intended to catch.

a) Integration bugs
b) Syntax and logic errors in a module
c) Network issues
d) Data migration errors

116. Apply system testing and determine what it validates in the complete software application.

a) Each class
b) Whole system compliance with requirements
c) Library files
d) Input data speed

117. Choose a V-model testing activity that corresponds directly with the design phase.

a) Unit testing
b) Integration testing
c) Acceptance testing
d) System testing

118. Use defect tracking tools and identify the phase during which they are most commonly applied.

a) Requirement gathering
b) Testing phase
c) Feasibility study
d) Maintenance phase only

119. Choose an approach that correctly applies usability testing in software evaluation.

a) Assessing hardware compatibility
b) Evaluating user-friendliness of interface
c) Measuring CPU speed
d) Tracking bugs

120. Demonstrate the focus of integration testing during the software testing lifecycle.

a) Interface between components
b) Hardware-software compatibility
c) Database normalization
d) UI color themes

121. Apply quality assurance practices and determine their primary focus.

a) Final stage delivery
b) Preventing defects during development
c) Post-release tracking only
d) Backup policies

122. Choose the correct purpose of designing a test case during software testing.

a) Define budget
b) Provide testing procedure and expected output
c) Draw GUI design
d) Create class hierarchy

123. Demonstrate acceptance testing and identify what aspect of the software it verifies.

a) System matches internal specifications
b) System meets customer needs
c) Class variables are valid
d) Speed of internet

124. Use the Capability Maturity Model (CMM) to track the maturity of which development process aspect.

a) Team attendance
b) Process maturity level
c) CPU temperature
d) GUI design

125. Select the activity that directly relates to designing software test cases.

a) Identify input and expected output
b) Optimize database schema
c) Build network diagram
d) Develop animations

126. Interpret the purpose of project scheduling in managing software timelines.

a) To decorate the interface
b) To estimate memory usage
c) To define timelines and resources for tasks
d) To document the UI

127. Interpret the meaning of risk management in the context of software engineering.

a) Avoid all features
b) Design only the database
c) Identify, analyze, and plan for project risks
d) Check system fonts

128. Identify the key elements involved in effective software project planning.

a) Requirements and GUI
b) Size estimation, cost estimation, and scheduling
c) Code coloring
d) Menu creation

129. Recognize the role of Function Point Analysis in software measurement.

a) To decorate project charts
b) To count database tables
c) To estimate size based on functionality provided
d) To delete unused functions

130. Classify the main purpose of the COCOMO II model in cost estimation.

a) To test user satisfaction
b) To code user interface
c) To estimate cost, effort, and schedule
d) To style documentation

131. Infer the benefit of using earned value analysis in project performance tracking.

a) Determines UI colors
b) Measures project performance against baseline
c) Counts team members
d) Tracks installation time

132. Interpret the use of risk mitigation techniques within the RMMM strategy.

a) Ignore the risk
b) Create a prototype
c) Plan to reduce or avoid impact of risks
d) Install new hardware

133. Identify an activity typically associated with project management.

a) Requirement deletion
b) Project planning and tracking
c) Keyboard testing
d) Printing documentation

134. Review the purpose of defining tasks during software project scheduling.

a) To calculate CPU temperature
b) To determine roles and deliverables
c) To clean code
d) To draw flowchart

135. Distinguish between size estimation and cost estimation in software projects.

a) Size is about pixels, cost about time
b) Size is about tasks, cost is financial resources
c) Size is a tool, cost is a language
d) They are the same

136. Recognize the role of quality standards such as ISO 9000 in software processes.

a) For user training
b) To ensure software meets quality processes
c) For printing manuals
d) To reduce typing time

137. Identify a benefit of applying project management tools in software development.

a) Reduces team size
b) Improves code length
c) Helps in planning, scheduling, and tracking
d) Increases RAM

138. Interpret the significance of risk monitoring during the software lifecycle.

a) Track known risks and detect new ones
b) Buy software
c) Rename tasks
d) Make the UI bright

139. Review how Lines of Code (LOC) estimation assists project managers.

a) Defines GUI layout
b) Estimates project size by counting lines of code
c) Manages sound effects
d) Prints pages

140. Identify the full form and components of RMMM in risk planning.

a) Risk Management and Monitoring Model
b) Random Memory Mapping Module
c) Rapid Modeling Method
d) Requirement Mapping Mechanism

141. Classify the types of activities involved in software project scheduling.

a) Installing antivirus
b) Defining tasks, timelines and dependencies
c) Color coding
d) Data deletion

142. Recognize how people management influences project execution.

a) To measure cable length
b) To select button icons
c) To organize and motivate team members
d) To track cloud usage

143. Identify what a task set represents in the context of software scheduling.

a) List of unrelated bugs
b) Sequence of project activities
c) Hardware commands
d) Memory errors

144. Infer the importance of cost estimation in managing project resources.

a) It adjusts brightness
b) It helps allocate resources effectively
c) It measures hard disk speed
d) It prints flowcharts

145. Interpret how project tracking ensures alignment with goals and schedules.

a) Recording IP addresses
b) Monitoring project progress against plan
c) Detecting virus
d) Locking files

146. Identify what constitutes a milestone in software project management.

a) A specific point or event in the project timeline
b) A color scheme
c) A hardware module
d) A GUI element

147. Recognize one common reason that contributes to project failure.

a) Excellent scheduling
b) Strong team leadership
c) Poor requirement analysis
d) Frequent testing

148. Classify the key output generated through cost estimation methods.

a) Risk metrics
b) Effort and budget projections
c) CPU load
d) Screen brightness

149. Review how software metrics enhance project monitoring and control.

a) By making interface icons
b) By providing quantitative measurement of project attributes
c) By fixing server IPs
d) By organizing fonts

150. Interpret how resources are allocated effectively in project planning.

a) Assigning tools to hardware
b) Assigning people and tools to tasks
c) Resetting servers
d) Documenting icons

Short Type Questions

B.

1 Define software engineering.
2 List any four attributes of good software.
3 State two types of software with examples.
4 Identify the key challenges faced by software engineers.
5 Recall the importance of system engineering in software development.
6 Explain the characteristics of software.
7 Discuss how software engineering helps in reducing cost.
8 Illustrate the difference between software and hardware.
9 Summarize the benefits of using a systematic software process.
10 Classify different types of software engineering costs.

11 Explain the concept of software process model.

12 Summarize the characteristics of the Waterfall model.

13 Discuss the purpose of process iteration.

14 Identify the benefits of incremental delivery.

15 Classify different agile methodologies.

16 Apply RAD model to a time-constrained project.

17 Demonstrate component-based development in real applications.

18 Use prototyping to gather early feedback.

19 Illustrate the benefits of Agile over Waterfall.

20 Choose an appropriate model for unstable requirements.

21 Apply use-case modeling for a library system.

22 Use an ER diagram to design a hospital system.

23 Demonstrate logical DFD for an ATM system.

24 Apply functional and non-functional requirements to a banking system.

25 Illustrate the structure of a Software Requirements Specification (SRS).

26 Formulate a complete SRS for an online exam portal.

27 Design a data dictionary for a student management system.

28 Create use-cases for a hotel booking application.

29 Demonstrate unit testing on a login module.

30 Apply black-box testing for an e-commerce cart.

31 Use equivalence partitioning to test date input.

32 Illustrate how validation testing is performed.

33 Apply regression testing after bug fixing.

34 Analyze the difference between verification and validation.

35 Distinguish between alpha and beta testing.

36 Examine how quality standards affect testing.

37 Explain how ISO 9000 standards contribute to software quality improvement.

38 Describe the five levels of the Capability Maturity Model (CMM) and their importance in software process improvement.

39 Explain the need for proper software project management to avoid project failure.

40 Explain how the size of software is estimated or calculated during project planning.

41 Summarize the steps involved in the Risk Mitigation, Monitoring, and Management (RMMM) plan.

42 Explain how earned value analysis helps track the progress and performance of a software project.

43 Describe the components of a network scheduling diagram and its role in project planning.

44 Differentiate between Function Point Analysis and Lines of Code (LOC) based estimation methods.

45 Explain the difference between verification and validation in software testing with suitable examples.

46 Describe the significance of quality assurance in the software development life cycle.

47 Design a Gantt chart for a software project schedule.

48 Formulate a risk management plan for an e-learning platform.

49 Describe the purpose of regression testing and when it is applied in a project.

50 Develop a project schedule for a mobile app project.

Long Answer Type Questions

C.

1 List and describe key characteristics of software.

2 Define software engineering and explain its purpose.

3 Differentiate between software and hardware with examples.

4 Explain the key challenges in software engineering.

5 Explain the features and drawbacks of the Waterfall model.

6 Compare iterative and incremental development models.

7 Apply the Spiral model in high-risk project planning.

8 Demonstrate the use of Agile methodology in real-life project development.

9 Design a strategy to gather both functional and non-functional requirements from various stakeholders for a new e-commerce platform.

10 Create a clear method to separate user requirements from system requirements in a healthcare management system.

11 Construct a logical DFD that models the core processes of an online ticket booking system.

12 Formulate a systematic approach for using data dictionary entries to validate the consistency of requirement specifications.

13 Develop a feasible solution to represent data relationships using an ER diagram for a library management system.

14 Design a structured interviewing technique that maximizes stakeholder engagement for effective requirement elicitation.

15 Propose a structured format for documenting functional requirements that enables traceability and impact analysis of changes.

16 Create a multi-user requirements gathering approach using viewpoint-oriented analysis.

17 Develop a framework to assess the technical feasibility of a software project.

18 Build a data dictionary template suitable for documenting all entities and attributes in an inventory system.

19 Design a change management process that protects SRS integrity while adapting to evolving needs.

20 Build a process for validating requirements to avoid conflicts and ensure completeness in a financial application.

21 Develop a step-by-step method to evaluate the feasibility of a software project focusing on operational and schedule aspects.

22 Propose a method to validate software requirements using stakeholder walkthroughs and checklist-based reviews.

23 Create a complete quality management system covering quality control, assurance, and improvement for software projects.

24 Design a software maintenance plan that includes corrective, adaptive, perfective, and preventive maintenance with proper process models.

25 Plan a method for updating legacy systems using reuse and modern testing practices.

26 Analyze the relationship between cohesion and coupling in software design principles.

27 Compare abstraction and information hiding as complexity reduction techniques.

28 Differentiate between architectural patterns and design patterns based on their scope.

29 Examine how modularity contributes to achieving functional independence in software systems.

30 Classify the seven types of cohesion based on their strength and desirabilit

31 Analyze the impact of tight coupling on software maintainability.

32 Investigate the role of refinement in top-down design methodology.

33 Compare command-line interfaces with graphical user interfaces in terms of usability.

34 Categorize modern GUI elements based on their primary interaction functions.

35 Examine the design considerations for effective error message implementation.

36 Distinguish between procedural and object-oriented programming paradigms.

37 Analyze the criteria for selecting appropriate programming languages for specific projects.

38 Break down the key components that constitute good programming practices.

39 Compare different coding standards in terms of their benefits for team development.

40 Investigate the relationship between user interface design principles and GUI control selection.

41 Design a comprehensive testing framework for software quality assurance using verification and validation principles.

42 Develop a complete project plan for a restaurant POS system.

43 Develop a systematic quality management process based on ISO 9000 standards for software organizations.

44 Formulate an integrated cost estimation model using function point analysis techniques.

45 Construct a comprehensive risk management framework using RMMM methodology for software projects.

46 Generate a complete test case design methodology for achieving maximum defect detection coverage.

47 Synthesize a project scheduling approach using network scheduling techniques for software development.

48 Build a detailed CMM implementation plan for organizational process maturity improvement.

49 Compose an earned value analysis system for real-time project performance measurement.

50 Plan an integrated people management strategy for software project team effectiveness.