INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

Examinations Control Office

Examination	B TECH VI SEMESTER END EXAMINATIONS REGULAR JUNE 2025 REG UG20		
Month & Year	1-Jun	**Date**	23/06/2025
Course Name	SOFTWARE QUALITY ASSURANCE AND TESTING		
Course Code	ACIC02	**E-Code**	3432

Instructions to Evaluators

- Evaluators should spend at least 3-5 minutes on one answer booklet during the evaluation.
- Evaluators should cross check that marks are allotted for all the attempted questions.
- The marks should be assigned fairly according to the mark distribution specified in the scheme of evaluation.
- For questions that were attempted incorrectly, evaluators are required to award zero marks.
- The evaluator must give a proper justification in case of any mistakes identified in the marks provided.

START WRITING FROM HERE

Q.No.

1(a) Software testing refers to checking the software and ensure that it meets the requirements of the users. It refers to testing of the software to ensure that it works for its intended purpose. Software testing is one of the most important aspect in the software developing life cycle. It ensures that the system works for its intended purpose and the users can experience a error free and bug free interface. Therefore the primary objective of software testing — is to ensure the reliability, security, customer satisfaction etc. The software testing ensures that the software works without any errors coming in the way. Without software testing, if we deploy our software to the customers, it may lead to many problems that arises like non-functional software, software that is full of errors, redundancy and lower customer satisfaction. Therefore it is important to address all these problems before deployment of our product and that is where software testing comes into play. Software testing's primarily objective, therefore is to

Q.No.

solve these problems. Thus, testing will contribute to software reliability and customer satisfaction. Testing the software ensures that our software has gone through a number of test cases and thus we can simulate the real-time environments through testing. Testing will ensure that the software has been simulated in various test cases and that it works as intended and gives the desired output after execution. This ensures that, after the software has been released, the users can experience a robust and therefore, a reliable experience. This leads to the software reliability. So after robust testing, we can ensure that our software is error- and bug-free for the most part and thus it doesn't lead to any faulty systems and thus failure which may lead to downtime. Therefore, testing ensures that we can capture any such bugs early-on and therefore ensure a smooth and reliable experience for the customer. Therefore, the software testing contributes to

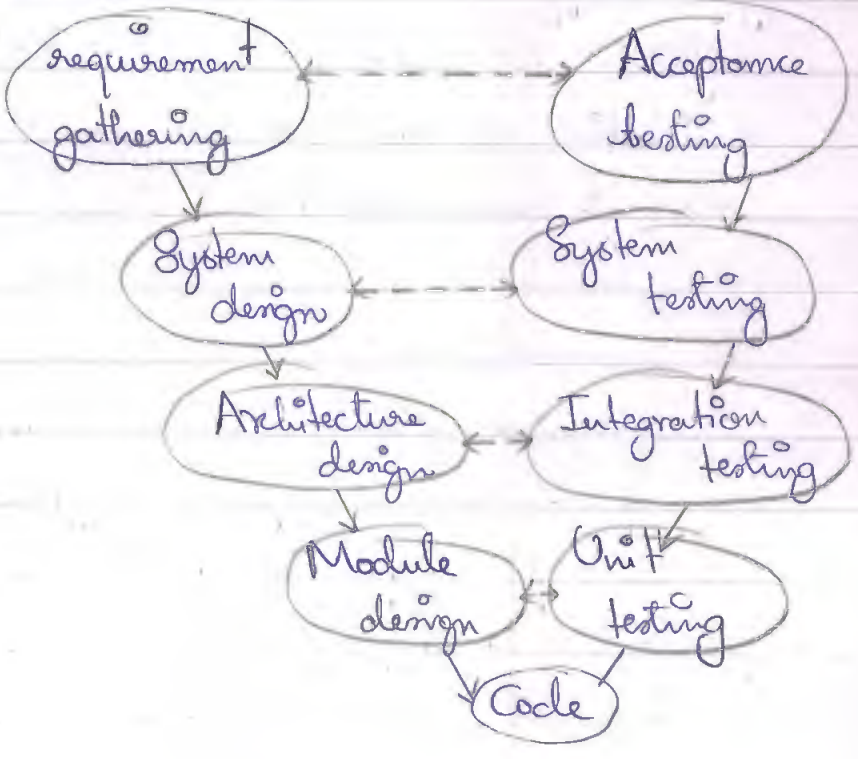
Q.No.

Software reliability and customer satisfaction.

1(b) Verification and Validation (V&V) are the two models/methods in software testing that used to check if our software is working properly or not.

Verification is the method in software testing that ensures if the different components of the software are working properly or not. It is the process in which the software is verified against a set guidelines and ensures that it is adhering to them. It verifies the software and its input and check if there are any anamolies.

V-model:



Q.No.

Verification ensures that the software is upto the standards set up the users. Validation in software testing, on the other hand will validate the input and/or output against the set guidelines and ensures that it is adhering to the guidelines unlike verification, which just check the software. Validation is used to check different types of test cases against the software and checks if it is validated or not. Verification on the other hand is used to verify our input against the output that we expect and just verify the results and checks if our software is working properly. We use validation to test the different test cases, and it ensures that maximum coverage of the software. Therefore, validation is one of important steps in software testing. Similarly, verification is used to check if one particular are of our software is working properly or not. These are thus the fundamental difference between Verification and Validation in Software testing. Test automation

Q.No.

in modern software testing plays a key role. Automated testing ensures that the tasks that are recursive in nature, testing multiple test cases etc are performed without any human intervention. It ensures that the time consumed is reduced drastically. Automated testing is also highly efficient and automates the tasks and it also results in high accuracy since there is no scope for any human errors. It improves the testing processes dramatically.

2(a) Testing a function refers to performing different tests on any give function independently where the functions of a specific part of the software is tested individually, properly. It ensures that the function is working as intended and it performs its functions. Testing a function independently needs to be done first to ensure that it is working properly by itself before the integrating with other components.

Testing a function in content is the concept that refers to the performing different tests on the given function in the specific

Q.No.

context that is provided. After ensuring that the function is performing on its own individually, we integrate the function with other components or test it in any environment to check if it is performing the same as expected is the providing of context for testing a functions. The testing of the function is subject to many different environments and testing methods thus giving rise to the concept of testing a function in context.

Incremental Integration: Incremental integration approach is the approach in which we slowly integrate/combine the individual components of the software and testing them as we go. We start with a simple component and we test it, then we integrate another component and we test this new combined one and check if it works and we continue. This is the incremental integration approach.

Top-down Integration: Top down integration is the method in which we iterated from the top and slowly move down to

Q.No.

the bottom by testing the elements from the top. We start at the top and itegrate downwards and we come down.

Bottom-Up Integration : Bottom-up integration is the integration approach in which we start at the bottom of the software components and we integrate the different elements as we move upwards. and testing them thus it is known as bottom-up integration.

Sandwich integration : Sandwich integration is the approach in which we integrate different components of the software that are random leaving the adjacent and ensure that the components are working properly. This approach is known as Sandwich integration.

Big-Bang Integration : Big Bang integration method is the method in which we test all the components of the software at once and thus integration of all of the software components is known as big-bang integration.

2(b) Boundary Value Analysis (BVA) is a type of ~~anal~~ analysis that is primarily used in the black-box testing method where we

Q.No.

make use of the edge cases to test the software. It helps in enhancing the test coverage in system testing. Boundary value analysis defines that we test the different edge test cases against our software and note the results. These results are thus used for the analysis of the software when boundary values are used and thus predict the behaviours of our software. Decision tables are the tables that are drawn with a set of parameters and then the resultant effects of the parameters on the action of our software or hardware to check whether they result in fail or pass of the different parameters and the test cases. Decision table notes the results against various parameters and we can ensure that we covers the maximum area of testing. Both Boundary Value Analysis (BVA) and decision tables improve test coverage in the system testing by ensuring that the edge cases of the software are tested and making sure that we don't miss any test cases where software may

Q.No.

not function. Boundary value analysis ensures that all edge cases are covered and decision table will make it easy to check the test cases and ensure that maximum test coverage is done in system testing. Software reliability refers to the reliable and robust nature of the software where the users can trust the software and can depend on it thus producing the reliability of software. There are several factors influencing reliability, they are — We need to ensure that the test coverage in testing is maximum to ensure that test cases are not missed and increasing the reliability. The software should also meet the expectations of the users thus improving its reliability. We also need to ensure the quality of the software. These are some of the factors influencing reliability.

3(a) System test design is one of the most important steps in the software testing lifecycle. System test design refers to the planning of the test along with the steps and procedures to follow, the steps also helps with the

Q.No.

design of the scope of the system testing where we need to clearly mark the objectives of the testing and thus we need to design the test such that it adheres with the guidelines and objectives and it meets the expectations of the users. There are several factors that influence the system test design. Firstly, we need to define the test objectives clearly and we need to identify and define the user requirements. After the objectives have been designed, we need to define the scope of the testing and design a plan to ensure that the user expectations will be met. The available resources and the management of the resources also influence the system design because they dictate how the flow of testing will go and therefore, we need to keep them in mind while designing the system test as they greatly influence it. These are some of the factors that influence the system test design. Test case design effectiveness can be measured using the metrics. There

Q.No.

are different metrics that can be used to measure the effectiveness of test case design. Test cases can be measured by simply performing them against the software and then defining if they are effective or not. These metrics are made to test the various different ways that are used to design the different test cases.

3(b) Finite State Machine (FSM) models are used to generate the test cases. Test cases are designed to test the software and its effectiveness. Therefore, we need to generate effective test cases and one the models used to generate test cases are finite state machine. Finite state model (FSM) is a machine that have a finite set of states. The machine is always in one state or an another. So we generate the test cases, by giving the input to finite state machine and thus it will change the input of the state. The state of the finite state machine isn't fixed and therefore as we change the input for the machine, the state of the machine also change. This

Q.No.

change of state are driven by events. These events trigger the change in the state of the machine. For example, there exist a login page and then the state of the login page is in some finite state, then when you enter the credentials, then the system will validate the input when you click login, here, it is the event and it is wrong, it displays error and here, error is another finite state. Similarly, if credentials are right, then it moves to home page and here it is another state. Therefore, the change of the states help in the generation of testcases by using finite state machine (FSM).

The transition tour method is the method where the transition to all the states of our software occurs atleast once. Forexample, login page is in a state, we enter credentials, wrong means, it changes to error state and right means, it goes to home page and here all the transitions have occured atleast once including the error state.

Q.No.

7(a). Root cause Analysis (RCA) is the analysis where where if any problem arises instead of only fixing the issue, we dig deep and find the reason the error arose and thus we can take proactive steps and ensure that we fix the problem at the root level and such issue will not occur again. We use root cause analysis to analyse and understand the problem that is caused. In testing, fixing a problem is costly, therefore we need to act proactively to identify errors beforehand and thus fix them before any issue arises. Root cause analysis will therefore is very beneficial and it helps in identifying and diagnosing errors so that the same errors will not arise in the future and thus saving time and money. It helps in identifying the defects in the software along with why the defect has been caused. Defect prevention is the technique in which we aim to identify the defects even before they arise and then diagnosing the errors. Defects in software can be costly so we need more testing methods so that we prevent the

Q.No.

defects i.e defect prevention. Root cause Analysis is used heavily for the defect prevention. Using the root cause analysis, once any errors has raised, instead of only solving the problem, we act more proactively and dig down to find the cause due to which the error has occured. After finding out the error and its cause, we analyse the cause of the errors and we solve the cause by identifying it and then diagnosing the cause. This ensures that the same defects aren't occured in the future and therefore we can't prevent any similar defects in the future, thus root cause analysis (RCA) is used in the defect prevention.

7(b) (i) Software fault tolerance: Software fault tolerance refers to term where if any error has occured in the software, then the software will not be affected as much and will not lead to the downtime. It refers to the capacity of the software to with respect to the

Q.No.

handling of errors and it working even through the software has some errors. We need to ensure that the fault tolerance is high and therefore it doesn't lead to downtime even with small numbers of errors. This is about the software fault tolerance.

(ii) Safety assurance : Safety assurance is the term that is used to define that our software is safe to use by the users. It assures that the software is reliable and therefore, it is safe to use. Similar to the quality assurance, the safety assurance encompass many things like ensuring that our software doesn't cause any errors in different environments ensuring interoperability, it is also safe from any malicious content and can be trusted upon by the user. In short, safety assurance ensures the safe usage of the software to the users.

(iii) Failure Containment : Failure containment refers to the containment or containing any failure that has occured in the software in its place and ensuring that the

Q.No.	
	failure doesn't spread out to other areas of our software. We must use robust technique, first, to identify any failure in the system and then making sure that this failure doesn't affect the functionality of other parts of the system. We need to isolate the area where the failure has occured and we need to contain it and fix it without letting it spread to other areas or affecting the functionality of other components in the software.
5(a)	McCall's quality factors are introduced by McCall to assess the quality of the software if they meet his certain criteria. There are three different quality factors. These categories are distinctly divided based on their purposes and the function of the software that they represent. They are Reliability – ensures that the software is reliable and the users can trust and depend on our software. Efficiency – It is described as the metrics that measures the efficiency of our software

Q.No.

overall and the efficiency of the functions.

Correctedness – ensures that our software is correct and it redeses the objectives & functions

Usability – describes whether the software is usable or not from the user's perspective and ensures that the software can be used with ease

Interoperatability – ensures that our software can be used in any environments and it doesn't lead to any errors or failures when we use it together with other elements.

Portability – defines that our software should be easily portalde from one platform to another and should work without any errors in different platforms.

Transportability – ensures that our software can be transported easily and doesn't lead to bugs in the process.

Compatability – ensures that our software is compatible with different types of environments and does work flawlessly.

These factors relate to specific quality criteria because all of these factors ensure that the quality of our software is assured. They work together and thus help in the

Q.No.

assurance of software quality.

5(b) Software quality Assurance (SQA) function in agile and DevOps environments: Agile and DevOps environments are the environments where there is continuous Integration and Continuous deployment (CI/CD) at work. It also uses the iterative approach therefore is continuous in these environments. In such environments, we thus also need to act iteratively such that we maintain the software quality assurance. We need to iteratively perform software testing. In Agile environments, there is a cycle of iterative development, therefore, we need to test the software at every step of the process iteratively ensuring that we perform testing and thus ensuring that they meet the specified requirements of the user. The testing will happen after each iteration to maintain the consistency of the software development. It also leads to the catching of errors in the early stage of the development

Q.No.

process so that no issue arises later in the development lifecycle. Thus, after each iteration the software quality is assured. Similarly, in devops environments, which has continuous integration and development, we also need to use the automation tools for the testing. We make sure that the automated tools will perform testing after each continuous integration automatically thus ensuring the software is tested and free of errors before deployment. This ensures the quality of software. Therefore, these methods will help to ensure the quality i.e this is how software quality assurance (SQA) function in agile and DevOps environments.

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.	

Q.No.

Q.No.	

Q.No.	

Q.No.	

ROUGH WORK

Content written here will not be considered for valuation

Requirements

System design

Architecture design

Acceptance testing

System testing

Integration