Q.1 Attempt any FIVE of the following : [10]

Q.1(a) Define software Quality Assurance and software Quality Control. [2]

Ans.:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Quality Assurance</th>
<th>Quality Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>Process oriented activities.</td>
<td>Product oriented activities.</td>
</tr>
<tr>
<td>(ii)</td>
<td>Quality assurance is the process of managing for quality.</td>
<td>Quality control is used to verify the quality of the output.</td>
</tr>
<tr>
<td>(iii)</td>
<td>They measure the process, identify the deficiencies/weakness and suggest improvements.</td>
<td>They measure the product, identify the deficiencies/weakness and suggest improvements.</td>
</tr>
<tr>
<td>(iv)</td>
<td>Relates to all products that will ever be created by a process</td>
<td>Relates to specific product</td>
</tr>
<tr>
<td>(v)</td>
<td>Activities of Quality assurance are process definition and implementation. Audits and Training</td>
<td>Activities of Quality control are reviews and testing.</td>
</tr>
<tr>
<td>(vi)</td>
<td>Verification is an example of quality assurance</td>
<td>Validation/Software testing is an example of quality control.</td>
</tr>
<tr>
<td>(vii)</td>
<td>Preventive activities.</td>
<td>It is a corrective process.</td>
</tr>
<tr>
<td>(viii)</td>
<td>Quality assurance is a proactive process.</td>
<td>Quality control is a reactive process.</td>
</tr>
<tr>
<td>(ix)</td>
<td>Quality assurance is a managerial tool.</td>
<td>Quality control is a corrective tool.</td>
</tr>
</tbody>
</table>

Q.1(b) Difference between white box and block box testing. [2]

Ans.: The Differences Between Black Box Testing and White Box Testing are listed below.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Black Box Testing</th>
<th>White Box Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester.</td>
<td>White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.</td>
</tr>
<tr>
<td>Levels Applicable To</td>
<td>Mainly applicable to higher levels of testing : Acceptance Testing System Testing</td>
<td>Mainly applicable to lower levels of testing : Unit Testing Integration Testing</td>
</tr>
<tr>
<td>Responsibility</td>
<td>Generally, independent Software Testers</td>
<td>Generally, Software Developers</td>
</tr>
<tr>
<td>Programming Knowledge</td>
<td>Not Required</td>
<td>Required</td>
</tr>
<tr>
<td>Implementation Knowledge</td>
<td>Not Required</td>
<td>Required</td>
</tr>
<tr>
<td>Basis for Test Cases</td>
<td>Requirement Specifications</td>
<td>Detail Design</td>
</tr>
</tbody>
</table>
Q.1(c) Enlist any four the benefits of Test Plan. [2]

Ans.: (i) Better coverage
   If written well, a test plan ensures that all aspects of a product are tested and covered.

(ii) More organized results allow for more efficient bug tracking
   Results from test plans are organized according to the section, and they build in which they were found. It is therefore much easier to see at a glance which areas are problematic.

(iii) Generates more bugs
   Test plans often help testers generate more bugs than exploratory testing. They are able to test all sections and all use cases. A benefit of test plans is that exploratory testing generates good and valuable bugs.

(iv) Avoids duplicates
   Test plans help to avoid duplicates. Test plans recognize and organize everything. If a tester finds a bug on version B, he can open the test plan for the previous version and see if the issue was logged. If it was, the tester can then discuss with the bug's original reporter to add information to the bug or to get a better idea of it. This promotes teamwork, and makes bug tracking much more efficient.

Q.1(d) Explain load testing. [2]

Ans.: Load testing: It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.

Q.1(e) Enlist different types of defect classification. [2]

Ans.: Classification
   Software Defects/Bugs are normally classified as per:
   - Severity/Impact
   - Probability/Visibility
   - Priority/Urgency
   - Related Dimension of Quality
   - Related Module/Component
   - Phase Detected
   - Phase Injected

Q.1(f) Write any four limitations of Manual Testing. [2]

Ans.: Limitations of Manual Testing are as given below:
   (i) Manual testing is slow and costly.
   (ii) It is very labor intensive; it takes a long time to complete tests.
   (iii) Manual tests don’t scale well. As the complexity of the software increases the complexity of the testing problem grows exponentially. This leads to an increase in total time devoted to testing as well as total cost of testing.
   (iv) Manual testing is not consistent or repeatable. Variations in how the tests are performed are inevitable, for various reasons. One tester may approach and perform a certain test differently from another, resulting in different results on the same test, because the tests are not being performed identically.
   (v) Lack of training is the common problem, although not unique to manual software testing.
   (vi) GUI objects size difference and color combinations are not easy to find in manual testing.
   (vii) Not suitable for large scale projects and time bound projects.
   Batch testing is not possible, for each and every test execution Human user interaction is mandatory.
   (viii) Comparing large amount of data is impractical.
   (ix) Processing change requests during software maintenance takes more time.
Q.1(g) Define following terms-Failure, Error, Defect and Bug. [2]

Ans.: The various terms related to software failure with respect to the area of application are listed as
Defect, Variance, Fault, Failure, Problem, Inconsistency, Error, Feature, Incident, Bug, and Anomaly.

Failure: the inability of a system or component to perform its required functions within specified performance requirements.

- Fault: An incorrect step, process, or data definition in a computer program.
- Error: A human action that produces an incorrect result.
- An error can be a grammatical error in one or more of the code lines, or a logical error in carrying out one or more of the client's requirements.
- Not all software errors become software faults. In some cases, the software error can cause improper functioning of the software. In many other cases, erroneous code lines will not affect the functionality of the software as a whole.
- A failure is said to occur whenever the external behaviour of a system does not conform to that prescribed in the system specification. A software fault becomes a software failure only when it is “activated”

Q.2 Attempt any THREE of the following: [12]

Q.2(a) With the help of neat diagram, describe unit testing. [4]

Ans.: {{**Note: Any other relevant diagram can be considered**}}

Unit Testing: Software product is made up of many units, each unit needed to be tested to find whether they have implemented the design correctly or not.

Additional Requirements: The module under consideration might be getting inputs from another module or the module is calling some another module. Some interface modules have to be simulated if required like drivers and stubs.

Drivers: The module where the required inputs for the module under test are simulated for the purpose of module or unit testing is known as a Driver module. The driver module may print or interpret the result produced by the module under test.

Stubs: The module under testing may also call some other module which is not ready at the time of testing. There is need of dummy modules required to simulate for testing, instead of actual modules. These are called stubs.
Unit testing procedure: Unit testing is normally considered as an adjunct to the coding step. The design of unit test can be performed before coding begins or after source code has been generated. Guidance for establishing test cases for finding out undiscovered errors can be taken by the review of design information. Each test case need to be associated with set of expected results with it.

In many applications a driver is known as "main program" that takes input from test case data, also passes that data to the component that need to be tested and prints concerned results. Stubs are useful for replacing modules which are subordinate of the component that we are going to test. A dummy sub program or stub can make use of subordinate modules interface. It generally does very less data manipulation, also provides verification of entry and used to return control to the module i.e. currently undergoing the testing.

Q.2(b) Differentiate between Drivers and Stub (any four points). [4]

**Ans.:** Difference between driver and stub

- **Stubs**
  1. Used in Top-Down Integration Testing.
  2. Code that emulates the called function.
  3. Stimulates the activity of missing & not developed modules.
  4. Created when lower level modules are not developed and higher level modules are tested.

- **Drivers**
  1. Used in Bottom-Up Integration Testing.
  2. Code that simulates a calling function.
  3. Drivers pass test cases to another code & invoke modules under testing.
  4. Created when high-level modules are not yet developed and lower level modules are tested.

Q.2(c) Explain Equivalence partitioning with respect to equivalence classes. [4]

**Ans.:** Equivalence partitioning is a software technique that involves identifying a small set of representative input values that produce as much different output condition as possible. This reduces the number of permutation & combination of input, output values used for testing, thereby increasing the coverage and reducing the effort involved in testing.

The set of input values that generate one single expected output is called a partition. When the behavior of the software is the same for a set of values, then the set is termed as equivalence class or partition.

Example: An insurance company that has the following premium rates based on the age group.
A life insurance company has a base premium of $0.50 for all ages. Based on the age group, an additional monthly premium has to pay that is as listed in the table below. For example, a person aged 34 has to pay a premium = $0.50 + $1.65 = $2.15

<table>
<thead>
<tr>
<th>Age group</th>
<th>Additional Premium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under 35</td>
<td>$1.65</td>
</tr>
<tr>
<td>35-59</td>
<td>$2.87</td>
</tr>
<tr>
<td>60+</td>
<td>$6.00</td>
</tr>
</tbody>
</table>

Based on the equivalence partitioning technique, the equivalence partitions that are based on age are given below:
- Below 35 years of age (valid input)
- Between 35 and 59 years of age (valid input)
- Above 6 years of age (valid input)
- Negative age (invalid input)
- Age as 0 (invalid input)
- Age as any three-digit number (valid input)

Q.2(d) State the Advantages and Disadvantages of using testing tools.

Ans.: Advantages of using automated tools for testing are as follows:

(i) Automated Software Testing Saves Time and Money
- Software tests have to be repeated often during development cycles to ensure quality.
  Every time source code is modified, software tests should be repeated.
- For each release of the software, it may be tested on all supported operating systems and hardware configurations.
- Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests.
- Automated software testing can reduce the time to run repetitive tests from days to hours.
- A time savings that translates directly into cost savings.

(ii) Testing Improves Accuracy
- Even the most conscientious tester will make mistakes during monotonous manual testing.
- Automated tests perform the same steps precisely every time they are executed and never forget to record detailed results.

(iii) Increase Test Coverage
- Automated software testing can increase the depth and scope of tests to help improve software quality.
- Lengthy tests that are often avoided during manual testing can be run unattended.
- They can even be run on multiple computers with different configurations.
- Automated software testing can look inside an application and see memory contents, data tables, file contents, and internal program states to determine if the product is behaving as expected.
- Automated software tests can easily execute thousands of different complex test cases during every test run providing coverage that is impossible with manual tests.
- Testers freed from repetitive manual tests have more time to create new automated software tests and deal with complex features.
(iv) Automation Does What Manual Testing Cannot
- Even the largest software departments cannot perform a controlled web application test with thousands of users.
- Automated testing can simulate tens, hundreds or thousands of virtual users interacting with network or web software and applications.

Disadvantages of using automated tools for testing are as follows:
(i) High investments required in package purchasing and training.
(ii) High package development investment costs.
(iii) High manpower requirements for test preparation.
(iv) Considerable testing areas left uncovered.

Q.3 Attempt any THREE of the following: [12]
Q.3(a) What do you mean by Software Metrics? List any three types of metrics. [4]
Ans.: Software metrics: Metrics are necessary to provide measurements of such qualities. Metrics can also be used to gauge the size and complexity of software and hence are employed in project management and cost estimation.

OR

A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute. Metrics can be defined as "STANDARDS OF MEASUREMENT". Software Metrics are used to measure the quality of the project. Simply, Metric is a unit used for describing an attribute. Metric is a scale for measurement.

Types of software metrics are:
(i) Project metrics
(ii) Progress metrics
(iii) Productivity metrics

Q.3(b) Explain Client-Server testing. [4]
Ans.:

In Client-server testing there are several clients communicating with the server.
- Multiple users can access the system at a time and they can communicate with the server.
- Configuration of client is known to the server with certainty.
- Client and server are connected by real connection.
- Testing approaches of client server system:

1. Component Testing: One need to define the approach and test plan for testing client and server individually. When server is tested there is need of a client simulator, whereas testing client a server simulator, and to test network both simulators are used at a time.
2. **Integration testing**: After successful testing of server, client and network, they are brought together to form system testing.

3. **Performance testing**: System performance is tested when number of clients is communicating with server at a time. Volume testing and stress testing may be used for testing, to test under maximum load as well as normal load expected. Various interactions may be used for stress testing.

4. **Concurrency Testing**: It is very important testing for client-server architecture. It may be possible that multiple users may be accessing same record at a time, and concurrency testing is required to understand the behavior of a system in this situation.

5. **Disaster Recovery Business continuity testing**: When the client server are communicating with each other, there exit a possibility of breaking of the communication due to various reasons or failure of either client or server or link connecting them. The requirement specifications must describe the possible expectations in case of any failure.

6. **Testing for extended periods**: In case of client server applications generally server is never shutdown unless there is some agreed Service Level Agreement (SLA) where server may be shut down for maintenance. It may be expected that server is running 24X7 for extended period. One needs to conduct testing over an extended period to understand if service level of network and server deteriorates over time due to some reasons like memory leakage.

7. **Compatibility Testing**: Client server may be put in different environments when the users are using them in production. Servers may be in different hardware, software, or operating system environment than the recommended. Other testing such as security testing and compliance testing may be involved if needed, as per testing and type of system.

Q.3(c) **Draw Defect Management Process. State the working of each phase.**

**Ans.:**

![Defect Management Process Diagram]

1. **Defect Prevention**: Implementation of techniques, methodology and standard processes to reduce the risk of defects.

2. **Deliverable Baseline**: Deliverables are considered to be ready for further developments. i.e. the deliverables meet exit criteria.

3. **Defect Discovery**: To find the defect through the process of verification and validation.

4. **Defect Resolution**: Defect is corrected or corrective action is taken and notification is given to tester.

5. **Process Improvement**: To identify ways to improve the process to prevent further future occurrences of similar defects, i.e. Corrective and preventive action is taken for processes improvement.

**Management Reporting**: Reporting is about status of application and processes.
Q.3(d) State any four points of comparison between Static analysis tools and Dynamic analysis tools.

**Ans.**:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Static analysis tool</th>
<th>Dynamic analysis tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>These tools are used by developers as part of the development and component testing process</td>
<td>These tools require the code to be in a &quot;running state&quot;</td>
</tr>
<tr>
<td>(ii)</td>
<td>code is not executed or run but tool itself is executed</td>
<td>They analyse rather than testing</td>
</tr>
<tr>
<td>(iii)</td>
<td>It is extension of compiler technology</td>
<td>They also help to understand background processes</td>
</tr>
<tr>
<td>(iv)</td>
<td>It also perform static analysis of requirement or analysis of website</td>
<td>These tool used by developed in component integration testing, middle ware, testing robustness and security.</td>
</tr>
<tr>
<td>(v)</td>
<td>Helps to understand the structure of the code and can also be useful to enforce coding standards.</td>
<td>Also performs web site testing to check whether each link does actually link to something else, it can find dead links.</td>
</tr>
<tr>
<td>(vi)</td>
<td>Features /characteristics of static testing tools are:</td>
<td>Features /characteristics of static testing tools are:</td>
</tr>
<tr>
<td></td>
<td>• Checks cyclomatic Complexity</td>
<td>• Detect memory leak</td>
</tr>
<tr>
<td></td>
<td>• Enforces coding standards</td>
<td>• Identify pointer arithmetic errors, null pointer</td>
</tr>
<tr>
<td></td>
<td>• Analyse structures and Dependencies</td>
<td>• Identify time dependence.</td>
</tr>
<tr>
<td></td>
<td>• Helpful in understanding coding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Identify defects in code.</td>
<td></td>
</tr>
<tr>
<td>(vii)</td>
<td>Examples. Flow analyzer, path tests, coverage analyzers, Interface analyzers</td>
<td>Examples. Test driver, Test beds, Emulators, Mutation analyzers</td>
</tr>
</tbody>
</table>

Q.4 Attempt any THREE of the following:

Q.4(a) Describe the Integration Testing.

**Ans.**:

Testing that occurs at the lowest level is called unit testing or module testing. As the units are tested and the low-level bugs are found and fixed, they are integrated and integration testing is performed against groups of modules. This process of incremental testing continues, putting together more and more pieces of the software until the entire product or at least a major portion of it is tested at once in a process called system testing.

With this testing strategy, it’s much easier to isolate bugs. When a problem is found at the unit level, the problem must be in that unit. If a bug is found when multiple units are integrated, it must be related to how the modules interact. Of course, there are exceptions to this, but by and large, testing and debugging is much more efficient than testing everything at once.

Types of Integration testing:

(i) **Top down Testing**: In this approach testing is conducted from main module to sub module. If the sub module is not developed a temporary program called STUB is used for simulate the sub module.
Advantages
- Advantageous if major flaws occur toward the top of the program.
- Once the I/O functions are added, representation of test cases is easier.
- Early skeletal Program allows demonstrations and boosts morale.

Disadvantages:
- Stub modules must be produced
- Stub Modules are often more complicated than they first appear to be.
- Before the I/O functions are added, representation of test cases in stubs can be difficult.

(ii) Bottom up testing: In this approach testing is conducted from sub module to main module, if the main module is not developed a temporary program called DRIVERS is used to simulate the main module.

Advantages:
- Advantageous if major flaws occur toward the bottom of the program.
- Test conditions are easier to create.
- Observation of test results is easier.

Disadvantages:
- Driver Modules must be produced.
- The program as an entity does not exist until the last module is added.

(iii) Bi-Directional Integration
- Bi-directional Integration is a kind of integration testing process that combines top-down and bottom-up testing.
- With an experience in delivering Bi-directional testing projects custom software development services provide the best quality of the deliverables right from the development of software process.
- Bi-directional Integration testing is a vertical incremental testing strategy that tests the bottom layers and top layers and tests the integrated system in the computer software development process.
- Using stubs, it tests the user interface in isolation as well as tests the very lowest level functions using drivers. Bi-directional Integration testing combines bottom-up and top-down testing
- Bottom-up testing is a process where lower level modules are integrated and then tested.
- This process is repeated until the component of the top of the hierarchy is analyzed. It helps custom software development services find bugs easily without any problems.
- Top down testing is a process where the top integrated modules are tested and the procedure is continued till the end of the related module.
- Top down testing helps developers find the missing branch link easily.

(iv) Incremental Integration
- After unit testing is completed, developer performs integration testing.
- It is the process of verifying the interfaces and interaction between modules.
- While integrating, there are lots of techniques used by developers and one of them is the incremental approach.
- In Incremental integration testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects.
- This approach is known as incremental integration testing.
- To the contrary, big bang is one other integration testing technique, where all the modules are integrated in one shot.
(v) Non-Incremental Integration
- The non-incremental approach is also known as "Big-Bang" Testing.
- Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system.
- When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

Q.4(b) Prepare and write six test cases for simple calculator application. [4]
Ans.:

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Test Case ID</th>
<th>Test Case Objectives</th>
<th>Pre-requisites</th>
<th>Steps</th>
<th>Input data</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TC_1</td>
<td>To add two integer and display the result on ten digit calculator</td>
<td>Calculator is switched on</td>
<td>1. Key in a valid integer from -9999999999 to +9999999999&lt;br&gt;2. Key in operator +&lt;br&gt;3. Key in second operand, a valid integer from -9999999999&lt;br&gt;To +9999999999</td>
<td>135 + 100</td>
<td>235 (addition, above ten digits will be expressed in exponential form)</td>
<td>235</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>TC_2</td>
<td>To subtract two integer and display the result on ten digit calculator</td>
<td>Calculator is switched on</td>
<td>1. Key in a valid integer from -9999999999 to +9999999999&lt;br&gt;2. Key in operator -&lt;br&gt;3. Key in second operand, a valid integer from -9999999999&lt;br&gt;To +9999999999</td>
<td>135 - 100</td>
<td>35 (subtraction, above ten digits will be expressed in exponential form)</td>
<td>35</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Q. 4(c) Explain the Test Management with Test Infrastructure management and Test People Management.

Ans.: (i) Test case database(TCDB)

<table>
<thead>
<tr>
<th>Entity</th>
<th>Purpose</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case</td>
<td>Records all the “static” information about the test</td>
<td>• Test case ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test case name (filename)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test case owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Associated files for the test case</td>
</tr>
<tr>
<td>Test case-product cross-reference</td>
<td>Provides a mapping between the tests and the corresponding product features: enables identification of tests for a given feature</td>
<td>• Test case ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Modulate ID</td>
</tr>
<tr>
<td>Test case run history</td>
<td>Gives the history of when a test was run and what was the result; provides inputs on selection of tests for regression runs</td>
<td>• Test case ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Run date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Time taken</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Run status (success/failure)</td>
</tr>
<tr>
<td>Test case - Defect cross-reference</td>
<td>Gives details of test cases introduced to test certain specific defects detected in the product; provides inputs on the selection of tests for regression runs</td>
<td>• Test case ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defect reference# (points to a record in the defect repository)</td>
</tr>
</tbody>
</table>

### Example Test Case

<table>
<thead>
<tr>
<th>No.</th>
<th>TC_3</th>
<th>Purpose</th>
<th>Calculated on</th>
<th>1. Key in a valid integer from 1 to 9999999999</th>
<th>2. Key in operator x 100</th>
<th>3. Key in second operand a valid integer from 1 to 9999999999</th>
<th>40000 (multiplier above ten digits will be expressed in exponential form)</th>
<th>40000</th>
<th>Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>TC_3</td>
<td>To multiply two integers and display the result on ten-digit calculator</td>
<td>Calculated on</td>
<td>1. Key in a valid integer from -9999999999 to +9999999999</td>
<td>2. Key in operator x 100</td>
<td>3. Key in second operand a valid integer from -9999999999 To +9999999999</td>
<td>40000 (multiplier above ten digits will be expressed in exponential form)</td>
<td>40000</td>
<td>Pass</td>
</tr>
</tbody>
</table>
(ii) **Defect Repository**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Purpose</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect details</td>
<td>Records all the “static” information about the tests</td>
<td>• Defect ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defect priority/severity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defect description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Any relevant version information (for example, OS version)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Customers who encountered the problem (could be reported by the internal testing team also)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data and time of defect occurrence</td>
</tr>
<tr>
<td>Defect test details</td>
<td>Provides details of test cases for a given defect.</td>
<td>• Defect ID</td>
</tr>
<tr>
<td></td>
<td>Cross references the TCDB</td>
<td>• Test case ID</td>
</tr>
<tr>
<td>Fix details</td>
<td>Provides details of fixes for a given defect; cross references the configuration management repository</td>
<td>• Defect ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fix details (file changed, fix release information)</td>
</tr>
<tr>
<td>Communication</td>
<td>Captures all the details of the communication that transpired for this defect among the various stakeholders these could include communication between the testing team and development team, customer communication, and so on. Provides insights into effectiveness of communication</td>
<td>• Test case ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defect reference#</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Details of communication</td>
</tr>
</tbody>
</table>

**Q.4(d) Give comparison between Alpha testing and Beta testing. (any four points) [4]**

**Ans.:** 1. Alpha Testing is conducted by a team of highly skilled testers at development site whereas Beta Testing is always conducted in Real Time environment by customers or end users at their own site.

2. Alpha testing requires lab environment or testing environment, whereas Beta testing doesn’t require any lab environment or testing environment.

3. Since Alpha Testing is done onsite therefore developers as well as business analyst are involved with the testing team whereas in Beta Testing developers and business analysts are not at all involved.

4. Beta testers can be naive or proficient end users of software product but alpha testers are always high skilled professional testers.

5. Alpha Testing involves both black box testing as well as white box testing, Beta Testing is always a black box testing or functional testing.

6. Alpha Testing is done before the launch of software product into the market whereas Beta Testing is done at the time of software product marketing.
7. Alpha Testing is conducted in the presence of developers and in the absence of end users whereas for Beta Testing this is exactly reversed.

8. Alpha testing is to ensure the quality of the product before moving to Beta testing Beta testing also concentrates on quality of the product, but gathers users input on the product and ensures that the product is ready for real time users.

9. Reliability and security testing are not performed in-depth Alpha Testing Reliability, Security, Robustness are checked during Beta Testing.

Q.4(e) Describe V-model with labelled diagram. State its any two advantages and disadvantages. Also write where it is applicable.

Ans.: V-model means verification and validation model. It is sequential path of execution of processes. Each phase must be completed before the next phase begins. Under V-model, the corresponding testing phase of the development phase is planned in parallel. So there is verification on one side of V & validation phase on the other side of V.

Verification Phase:
1. Overall Business Requirement: In this first phase of the development cycle, the product requirements are understood from customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

2. Software Requirement: Once the product requirements are clearly known, the system can be designed. The system design comprises of understanding & detailing the complete hardware, software & communication set up for the product under development. System test plan is designed based on system design. Doing this at earlier stage leaves more time for actual test execution later.

3. High level design: High level specification are understood & designed in this phase. Usually more than one technical approach is proposed & based on the technical & financial feasibility, the final decision is taken. System design is broken down further into modules taking up different functionality.

4. Low level design: In this phase the detailed integral design for all the system modules is specified. It is important that the design is compatible with the other modules in the system & other external system. Components tests can be designed at this stage based on the internal module design.

5. Coding: The actual coding of the system modules designed in the design phase is taken up in the coding phase. The base suitable programming language is decided base on requirements. Coding is done based on the coding guidelines & standards.

Validation:
- **Unit Testing**: Unit testing designed in coding are executed on the code during this validation phase. This helps to eliminate bugs at an early stage.
- **Components testing**: This is associated with module design helps to eliminate defects in individual modules.
- **Integration Testing**: It is associated with high level design phase & it is performed to test the coexistence & communication of the internal modules within the system.
- **System Testing**: It is associated with system design phase. It checks the entire system functionality & the communication of the system under development with external systems. Most of the software & hardware compatibility issues can be uncovered using system test execution.
- **Acceptance Testing**: It is associated with overall & involves testing the product in user environment. These tests uncover the compatibility issues with the other systems available in the user environment. It also uncovers the nonfunctional issues such as load & performance defects in the actual user environment.
When to use the V-model:

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

Q.5 Attempt any TWO of the following:

Q.5(a) Design test cases for the data filed from Admission form of your institute (Data filed are Name, SSC percentage, Adhar no, Address, mobile no)

Ans.: Test cases for admission form (Data filed are Name, SSC percentage, Adhar no, Address, mobile no)

<table>
<thead>
<tr>
<th>TC Id</th>
<th>Test Case</th>
<th>Tester</th>
<th>Date</th>
<th>Duration</th>
<th>Actual O/p</th>
<th>Expected O/p</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC 1</td>
<td>Blank name</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;name cannot be blank&quot;</td>
<td>Display message &quot;name cannot be blank&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC</td>
<td>SSC percentage as characters</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “Invalid percentage”</td>
<td>Display message “Invalid percentage”</td>
<td>pass</td>
</tr>
<tr>
<td>TC3</td>
<td>SSC percentage &gt; 100%</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “Invalid percentage”</td>
<td>Display message “Invalid percentage”</td>
<td>pass</td>
</tr>
<tr>
<td>TC4</td>
<td>Blank Aadhar number</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “aadhar number cannot be blank”</td>
<td>Display message “aadhar number cannot be blank”</td>
<td>pass</td>
</tr>
<tr>
<td>TC5</td>
<td>Aadhar no as characters</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “Invalid aadhar no”</td>
<td>Display message “Invalid aadhar no”</td>
<td>pass</td>
</tr>
<tr>
<td>TC6</td>
<td>Blank Mobile no</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “Pls enter mobile no”</td>
<td>Display message “Pls enter mobile no”</td>
<td>pass</td>
</tr>
<tr>
<td>TC7</td>
<td>Mobile no as characters</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “Invalid mobile no”</td>
<td>Display message “Invalid mobile no”</td>
<td>pass</td>
</tr>
<tr>
<td>TC8</td>
<td>All valid entries</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message “data submitted successfully”</td>
<td>Display message “data submitted successfully”</td>
<td>pass</td>
</tr>
</tbody>
</table>

Q.5(b) Explain defect life cycle to identify status of defect with proper labelled diagram.

Ans.:

![Defect Life Cycle Diagram](image)
The different states of bug life cycle are as shown in the above diagram:

- **New**: When the bug is posted for the first time, its state will be "NEW". This means that the bug is not yet approved.

- **Open**: After a tester has posted a bug, the lead of the tester approves that the bug is genuine and he changes the state as "OPEN".

- **Assign**: Once the lead changes the state as "OPEN", he assigns the bug to corresponding developer or developer team. The state of the bug now is changed to "ASSIGN".

- **Test/Retest**: Once the developer fixes the bug, he has to assign the bug to the testing team for next round of testing. Before he releases the software with bug fixed, he changes the state of bug to "TEST". It specifies that the bug has been fixed and is released to testing team. At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

- **Deferred**: The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

- **Rejected**: If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to "REJECTED".

- **Verified**: Once the bug is fixed and the status is changed to "TEST", the tester tests the bug. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "VERIFIED".

- **Reopened**: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "REOPENED". The bug traverses the life cycle once again.

- **Closed**: Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to "CLOSED". This state means that the bug is fixed, tested and approved.

- **Fixed**: When developer makes necessary code changes and verifies the changes then he/she can make bug status as "Fixed" and the bug is passed to testing team.

- **Pending retest**: After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.

**Optional**

- **Duplicate**: If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to "duplicate".

- **Not a bug**: The state given as "Not a bug" if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.
Q.5(c) Describe use of load testing and stress testing to test online result display facility of MSBTE website.

Ans.: Stress Testing: In stress testing of MSBTE online result display, the resources used will be less than the requirement. For e.g. Provide less RAM for the server, or decrease the bandwidth of the internet connection, or provide less hits for page. If the system has limited resources available, the response of the online result system may deteriorate due to non-availability of the resources. It tries to break the page, site or connection under test by overwhelming its resources in order to find the circumstances under which it will crash. It is also a type of load testing. It is designed to determine the behavior of the software under abnormal situations. In stress testing test cases are designed to execute the system in such a way that abnormal conditions.

Load Testing: When a MSBTE online result display facility is tested with a load that causes it to allocate its resources in maximum amounts. The idea is to create an environment more demanding than the application would experience under normal workloads. Eg. Apply more number of hits on the result section, try displaying multiple results in multiple browsers, etc. Load is varied from minimum to the maximum level the system can sustain without running out of resources. Load is being increased transactions may suffer excessive delays. Load testing involves simulating real-life user load for the target application. It helps to determine how application behaves when multiple students hits it simultaneously to check the results. Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system.

Q.6 Attempt any TWO of the following:

Q.6(a) Design test cases for MSBTE Online Exam form filling.(any valid six test cases)

Ans.: Test cases for exam form filling

<table>
<thead>
<tr>
<th>TC Id</th>
<th>Test Case</th>
<th>Tester</th>
<th>Date</th>
<th>Duration</th>
<th>Actual O/p</th>
<th>Expected O/p</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC 1</td>
<td>Blank user id</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;User id cannot be blank&quot;</td>
<td>Display message &quot;User id cannot be blank&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 2</td>
<td>Wrong user id</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Wrong User id/Password&quot;</td>
<td>Display message &quot;Wrong User id/Password&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 3</td>
<td>Blank Password</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Password cannot be blank&quot;</td>
<td>Display message &quot;Password cannot be blank&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 4</td>
<td>Wrong password</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Wrong password&quot;</td>
<td>Display message &quot;Wrong password&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 5</td>
<td>No of wrong attempts =3</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Block user&quot;</td>
<td>Display message &quot;Block user&quot;</td>
<td>pass</td>
</tr>
</tbody>
</table>
### Q.6(b) Prepare a Test Plan along with the Test Cases for the MS Word option 'Save As'. Test Cases should be at least six.

**Ans.**

Test cases for "Save As"

<table>
<thead>
<tr>
<th>TC Id</th>
<th>Test Case</th>
<th>Tester</th>
<th>Date</th>
<th>Duration</th>
<th>Actual O/p</th>
<th>Expected O/p</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC 1</td>
<td>Click &quot;Save As&quot; option</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display Save as dialog box</td>
<td>Display Save as dialog box</td>
<td>pass</td>
</tr>
<tr>
<td>TC 2</td>
<td>Blank file name</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Pls enter file name&quot;</td>
<td>Display message &quot;Pls enter file name&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 3</td>
<td>Special character in file name</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 4</td>
<td>Same file name</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;File already exist&quot;</td>
<td>Display message &quot;File already exist&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 5</td>
<td>File name length is more</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 6</td>
<td>Wrong extension</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>Display message &quot;Invalid file name&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 7</td>
<td>Valid file name</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Save file</td>
<td>Save file</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Q. 6(c) Design any three test cases for railway reservation form and prepare defect report of it.

Ans.: Test cases for railway reservation form

<table>
<thead>
<tr>
<th>TC Id</th>
<th>Test Case</th>
<th>Test</th>
<th>Date</th>
<th>Duration</th>
<th>Actual O/p</th>
<th>Expected O/p</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC 1</td>
<td>Blank user id</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;User id cannot be blank&quot;</td>
<td>Display message &quot;User id cannot be blank&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 3</td>
<td>Blank Password</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Password cannot be blank&quot;</td>
<td>Display message &quot;Password cannot be blank&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 4</td>
<td>Password Length less than 8 characters</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Password must be at least of 8 characters&quot;</td>
<td>Display message &quot;Password must be at least of 8 characters&quot;</td>
<td>pass</td>
</tr>
<tr>
<td>TC 5</td>
<td>Re-type password not matching</td>
<td>SNN</td>
<td>23-SEP-19</td>
<td>1 day</td>
<td>Display message &quot;Passwords not matching&quot;</td>
<td>Display message &quot;Passwords not matching&quot;</td>
<td>pass</td>
</tr>
</tbody>
</table>