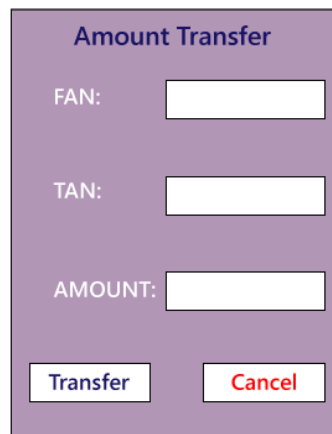


STQA CAT 2 ANSWERS

1. a) What is Integration test? Explain with an example

- Integration testing is the second level of the software testing process comes after unit testing.
- In this testing, units or individual components of the software are tested in a group.
- The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.
- Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**.
- **Example :**
Let us see one sample example of a banking application, as we can see in the below image of amount transfer.



The image shows a 'Amount Transfer' form with a purple background. It contains three input fields labeled 'FAN:', 'TAN:', and 'AMOUNT:'. Below these fields are two buttons: 'Transfer' and 'Cancel'.

- First, we will login as a user **P** to amount transfer and send Rs200 amount, the confirmation message should be displayed on the screen as **amount transfer successfully**. Now logout as P and login as user **Q** and go to amount balance page and check for a balance in that account = Present balance + Received Balance. Therefore, the integration test is successful.
- Also, we check if the amount of balance has reduced by Rs200 in P user account.
- Click on the transaction, in P and Q, the message should be displayed regarding the data and time of the amount transfer.
- Perform **positive and negative integration testing**.

Here **positive** testing implies that if the total balance is Rs15, 000 and we are transferring Rs1500 and checking if the amount transfer works fine. If it does, then the test would be a pass.

And **negative testing** means, if the total balance is Rs15, 000 and we are transferring Rs20, 000 and check if amount transfer occurs or not, if it does not occur, the test is a pass. If it happens, then there is a bug in the code, and we will send it to the development team for fixing that bug.

- How to do Integration Testing?
 1. Prepare the Integration Tests Plan
 2. Design the Test Scenarios, Cases, and Scripts.
 3. Executing the test Cases followed by reporting the defects.
 4. Tracking & re-testing the defects.
 5. Steps 3 and 4 are repeated until the completion of Integration is successful.

Q) What are different types of integration testing?

Types of Integration Testing

- Big Bang Approach :
- Incremental Approach: which is further divided into the following
 - Top Down Approach
 - Bottom Up Approach
 - Sandwich Approach – Combination of Top Down and Bottom Up

Big Bang Testing

Big Bang Testing is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

Advantages:

- Convenient for small systems.

Disadvantages:

- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
- Since the Integration testing can commence only after “all” the modules are designed, the testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

Incremental Testing

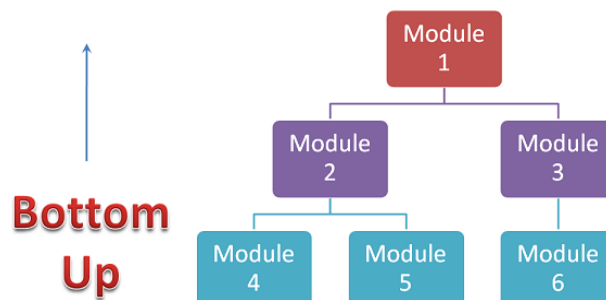
In the **Incremental Testing** approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application. Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

Incremental Approach, in turn, is carried out by two different Methods:

- Bottom Up
- Top Down

Bottom-up Integration Testing

Bottom-up Integration Testing is a strategy in which the lower level modules are tested first. These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested. Once the lower level modules are tested and integrated, then the next level of modules are formed.



Advantages:

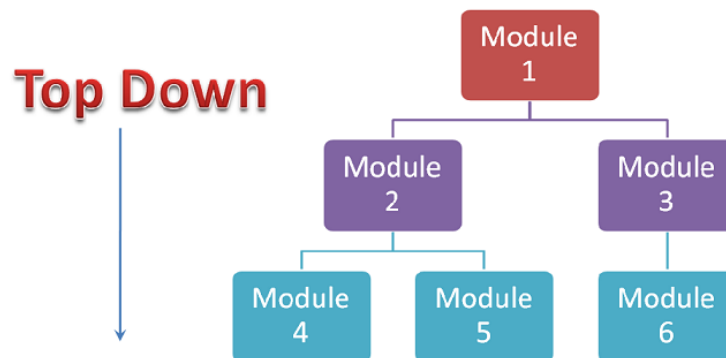
- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- An early prototype is not possible

🚦 Top-down Integration Testing

Top Down Integration Testing is a method in which integration testing takes place from top to bottom following the control flow of software system. The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality. Stubs are used for testing if some modules are not ready.



Advantages:

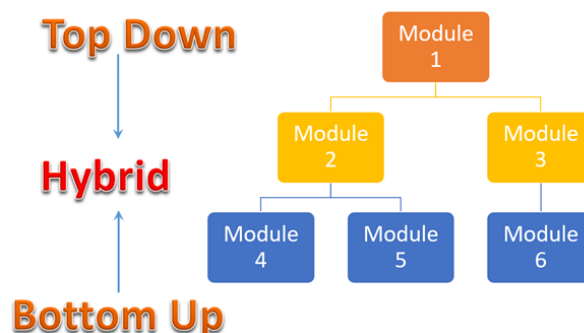
- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.

Disadvantages:

- Needs many Stubs.
- Modules at a lower level are tested inadequately.

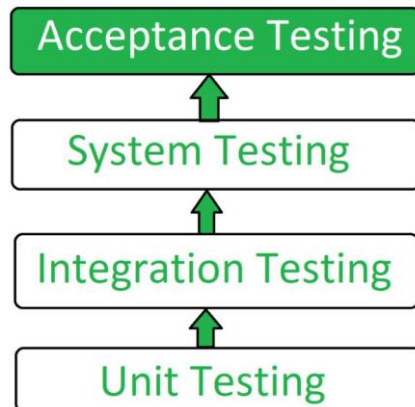
🚦 Sandwich Testing

Sandwich Testing is a strategy in which top level modules are tested with lower level modules at the same time lower modules are integrated with top modules and tested as a system. It is a combination of Top-down and Bottom-up approaches therefore it is called **Hybrid Integration Testing**. It makes use of both stubs as well as drivers.



1. b) Explain Acceptance Testing.

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.



Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

Acceptance Criteria.

1. Product acceptance
2. Procedure acceptance
3. Service level agreements

Selection test cases for Acceptance Testing

- ❖ End-to-end functionality verification
- ❖ Domain tests
- ❖ User scenario tests
- ❖ Basic sanity tests
- ❖ New functionality
- ❖ A few non-functionality
- ❖ Tests pertaining to legal obligations and service level agreements
- ❖ Acceptance test data

Types of Acceptance testing

- **Benchmarking:** a predetermined set of test cases corresponding to typical usage conditions is executed against the system
- **Pilot Testing:** users employ the software as a small-scale experiment or in a controlled environment
- **Alpha-Testing:** pre-release closed / in-house user testing
- **Beta-Testing:** pre-release public user testing
- **Parallel Testing:** old and new software are used together and the old software is gradually phased out.

Use of Acceptance Testing:

- To find the defects missed during the functional testing phase.
- How well the product is developed.
- A product is what actually the customers need.
- Feedback help in improving the product performance and user experience.
- Minimize or eliminate the issues arising from the production.

2. a) Give the difference between Alpha testing and Beta testing.

Alpha Testing	Beta Testing (Field Testing)
1. It is always performed by the developers at the software development site.	1. It is always performed by the customers at their own site.
2. Sometimes it is also performed by Independent Testing Team.	2. It is not performed by Independent Testing Team.
3. Alpha Testing is not open to the market and public	3. Beta Testing is always open to the market and public.
4. It is conducted for the software application and project.	4. It is usually conducted for software product.
5. It is always performed in Virtual Environment .	5. It is performed in Real Time Environment .
6. It is always performed within the organization.	6. It is always performed outside the organization.
7. It is the form of Acceptance Testing.	7. It is also the form of Acceptance Testing.
8. Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.	8. Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
11. It is always performed at the developer's premises in the absence of the users.	11. It is always performed at the user's premises in the absence of the development team.

2. b) What is Usability Testing and Accessibility testing?

Usability Testing

- Also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is.
- A small set of target end-users, use software application to expose usability defects.
- Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.
- This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.
- In **Usability Testing**, the **user-friendliness** can be described with the help of the following characteristics:
 - Easy to understand
 - Easy to access
 - Look and feel
 - Faster to Access
 - Effective Navigation
 - Good Error Handling

Accessibility Testing

- Accessibility Testing is defined as a type of Software Testing performed to ensure that the application being tested is usable by people with disabilities like hearing, color blindness, old age, and other disadvantaged groups.
- It is a subset of [Usability Testing](#).
- People with disabilities use assistive technology, which helps them in operating a software product. Examples of such software are:
 - **Speech recognition software** – Converts the spoken word to text, which serves as input to the computer.
 - **Screen reader software** – Used to read out the text that is displayed on the screen.
 - **Screen Magnification Software**– Used to enlarge the monitor and make reading easy for vision-impaired users.
 - **Special keyboard** made for users for easy typing who have motor control difficulties
- Accessibility Testing can be performed in 2 ways:
 1. **Manual:**

There are various tools available in the market to test the accessibility of a software application but may be the available tools are highly costly and/or are less skilled as per requirements. Therefore, manual testing is performed to check the accessibility of the software product. For example:

(a) Check the brightness of the software product. Check whether it is adjustable or not. Check is it good for a person with less eye sight.

(b) Check the sound performance of the software. Check whether it is properly usable by a deaf person.
 2. **Automated:**

Automation is widely used in different testing techniques. In the automated process, there are several automated tools for the accessibility testing. These tools include:

 - **(a) WebAnywhere** – It is a screen reader tool.
 - **(b) Hera** – It is used to check the style of the software application.

3. a) What are the levels of testing in software testing?

Level1: Unit Testing

- **Unit testing** is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.
- The first level of testing involves **analysing each unit or an individual component** of the software application.
- Unit testing is also the first level of **functional testing**. The primary purpose of executing unit testing is to validate unit components with their performance.
- A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.
- Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.
- This kind of testing is performed by developers.

Level2: Integration Testing

- The second level of software testing is the **integration testing**. The integration testing process comes after **unit testing**.
- It is mainly used to test the **data flow from one module or component to other modules**.
- In integration testing, the **test engineer** tests the units or separate components or modules of the software in a group.
- The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.
- When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.
- We only go for the integration testing when the functional testing has been completed successfully on each application module.
- In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.
- This kind of testing is performed by testers.

Level3: System Testing

- The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.
- It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system**.
- To check the end-to-end flow of an application or the software as a user is known as **System testing**.
- In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system.
- In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements.
- It evaluates both functional and non-functional need for the testing.

Level4: Acceptance Testing

- The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

- The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.
- In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done**.
- The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.
- Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

Levels of Testing

Unit Test

Test Individual Component

Integration
Test

Test Integrated Component

System Test

Test the entire System

Acceptance
Test

Test the final System

3. b) Explain Exploratory testing and give the execution steps for it.

- **Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers check system on the fly.
- They may note down ideas about what to test before test execution.
- The focus of exploratory testing is more on testing as a “thinking” activity.
- Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.
- Following is a step by step process on How to perform Exploratory Testing which is also called session based test management (SBTM Cycle):

Step 1) Create a Bug Taxonomy (classification)

- Categorize common types of faults found in the past projects
- Analyze the root cause analysis of the problems or faults
- Find the risks and develop ideas to test the application.

Step 2) Test Charter

- Test Charter should suggest
 1. what to test
 2. how it can be tested
 3. What needs to be looked
- Test ideas are the starting point of exploration testing
- Test charter helps determine how the end user could use the system

Step 3) Time Box

- This method includes a pair of testers working together not less than 90 minutes
- There should not be any interrupted time in those 90 minutes session
- Timebox can be extended or reduced by 45 minutes
- This session encourages testers to react on the response from the system and prepare for the correct outcome

Step 4) Review Results

- Evaluation of the defects
- Learning from the testing
- Analysis of coverage areas

Step 5) Debriefing

- Compilation of the output results
- Compare the results with the charter
- Check whether any additional testing is needed

4. a) What is a test data in software testing?

- **Test Data in Software Testing** is the input given to a software program during test execution.
- It represents data that affects or affected by software execution while testing.
- Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.
- Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.
- In Black Box Testing the code is not visible to the tester.
- Test Data can be Generated –
 - Manually
 - Mass copy of data from production to testing environment
 - Mass copy of test data from legacy client systems
 - Automated Test Data Generation Tools
- Your functional test cases can have test data meeting following criteria –
 - **No data:** Check system response when no data is submitted
 - **Valid data:** Check system response when Valid test data is submitted
 - **Invalid data:** Check system response when *InValid* test data is submitted
 - **Illegal data format:** Check system response when test data is in an invalid format
 - **Boundary Condition Dataset:** Test data meeting boundary value conditions
 - **Equivalence Partition Data Set:** Test data qualifying your equivalence partitions.
 - **Decision Table Data Set:** Test data qualifying your decision table testing strategy
 - **State Transition Test Data Set:** Test data meeting your state transition testing strategy
 - **Use Case Test Data:** Test Data in-sync with your use cases.

4. b) What is Test Data Generation? Why test data should be created before test execution?

- Everybody knows that testing is a process that produces and consumes large amounts of data.
- Data used in testing describes the initial conditions for a test and represents the medium through which the tester influences the software.
- It is a crucial part of most [Functional Tests](#).
- Depending on your testing environment you may need to CREATE Test Data (Most of the times) or at least identify a suitable test data for your test cases (is the test data is already created).
- Typically test data is created in-sync with the test case it is intended to be used for.
- Test Data can be Generated –
 - Manually
 - Mass copy of data from production to testing environment
 - Mass copy of test data from legacy client systems
 - Automated Test Data Generation Tools
- Typically sample data should be generated before you begin test execution because it is difficult to handle test data management otherwise.
- Since in **many testing environments creating test data takes multiple pre-steps or very time-consuming test environment configurations.** .
- Also If test data generation is done **while** you are in test execution phase you may exceed your testing deadline.

5. a) What are the approaches for Test Data Generation in Software Testing.

- **Manual Test data generation:** In this approach, the test data is manually entered by testers as per the test case requirements. It is a time taking the process and also prone to errors.
- **Automated Test Data generation:** This is done with the help of data generation tools. The main advantage of this approach is its speed and accuracy. However, it comes at a higher cost than manual test data generation.
- **Back-end data injection:** This is done through SQL queries. This approach can also update the existing data in the database. It is speedy & efficient but should be implemented very carefully so that the existing database does not get corrupted.
- **Using Third Party Tools:** There are tools available in the market that first understand your test scenarios and then generate or inject data accordingly to provide wide test coverage. These tools are accurate as they are customized as per the business needs. But, they are quite costly.

OR

1) Manual Test data generation:

In this technique, all the datasets are generated manually by the tester with respect to all the required test case through experience and anticipations.

Pros:

Easy to implement, no additional tools are needed to be deployed.

Increase the confidence of the tester.

Cons:

Accuracy of data sets generated by this scheme mostly doubtful.

Time-consuming process.

2) Automated Test Data Generation:

The major feature of this testing that makes it more efficient than the above technique is the speed, automated data generation technique produces data as in an expedited manner through analyzing large volume of data in a small-time interval. In this scheme, we use automated tools, there are many available in the market.

Pros:

The data sets generated by this scheme are highly accurate.

Data generation speed is very fast.

Cons:

The one demerit of this method is that it is a costlier method to implement.

The second one is that these tools take time to understand the system.

3) Back end data injection Approach:

This method is done with the help of using SQL queries. Here a tester writes the relevant query and injects it into the database in order to populate the required data sets with respect to the test cases. This is also an easier method which generates a large amount of data in just a few minutes. We can update the database in this scheme if some new datasets are found through other resources like sample XML documents etc could be updated for future use if required.

Pros:

It is less time-consuming technique.

Less expertise required as compared to the above technique as you only need to write a correct query to populate data required.

Cons:

If you write any invalid query or incorrect it may populate illogical dataset or may cause the failure of your database system so keep attention while injecting any query into database.

4) Third-party tool:

A number of tools are available in the market that is processed or provided by the out premises tools. These tools first understand the scenarios of your system under testing and then generates dataset as per the requirement. These tools are customizable as per your need of the business. These tools provide wide coverage and accuracy in generating datasets.

Pros:

These tools are accurate because they first understand the entire system and then generated the datasets accordingly.

Cons:

Costlier technique to implement because the price of such a tool is high as compared to other technique.

Less coverage in case of heterogeneous testing environment because these tools aren't generic in nature.

5. b) Explain the test plan with proper scenario.

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

Making Test Plan document has multiple benefits

- Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
- Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
- Important aspects like test estimation, test scope, [Test Strategy](#) are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

6. a) What is the Scope of Automation?

- Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite.
- The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports.
- Software Test Automation demands considerable investments of money and resources.
- The scope of Automation testing highly depends on the needs of the particular business.
- For some, it is to identify the right tools for Automation testing, while for others, it means determining the team size to execute Automated testing.
- But in common, the Automation testing scope is about understanding what your business testing requirements are and how you can automate them using the right set of tools.
- You can automate Functional, Performance, Regression, and other automation testing types to reduce your manual workload.
- The scope of automation is the area of your Application Under Test which will be automated.
- Following points help determine scope:
 - The features that are important for the business
 - Scenarios which have a large amount of data
 - Common functionalities across applications
 - Technical feasibility
 - The extent to which business components are reused
 - The complexity of test cases
 - Ability to use the same test cases for cross-browser testing

6. b) Enlist the Criteria for Selecting a Testing Tool.

The Categories for selecting Test Tools are,

1. Meeting requirements;
2. Technology expectations;
3. Training/skills;
4. Management aspects.

Explanation:

1. Meeting requirements : There are plenty of tools available in the market but rarely do they meet all the requirements of a given product or a given organization. Evaluating different tools for different requirements involve significant effort, money, and time. Given of the plethora of choice available, huge delay is involved in selecting and implementing test tools.
2. Technology expectations : Test tools in general may not allow test developers to extends/modify the functionality of the framework. So extending the functionality requires going back to the tool vendor and involves additional cost and effort. A good number of test tools require their libraries to be linked with product binaries.
3. Training/skills : While test tools require plenty of training, very few vendors provide the training to the required level. Organization level training is needed to deploy the test tools, as the user of the test suite are not only the test team but also the development team and other areas like configuration management.
4. Management aspects: A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

OR

Guidelines for selecting a tool:

1. The tool must match its intended use. Wrong selection of a tool can lead to problems like lower efficiency and effectiveness of testing may be lost.
2. Different phases of a life cycle have different quality-factor requirements. Tools required at each stage may differ significantly.
3. Matching a tool with the skills of testers is also essential. If the testers do not have proper training and skill then they may not be able to work effectively.
4. Select affordable tools. Cost and benefits of various tools must be compared before making final decision.
5. Backdoor entry of tools must be prevented. Unauthorized entry results into failure of tool and creates a negative environment for new tool introduction.

7. a) Explain how the Progress metrics is calculated.

7. b) Define Object Oriented testing and explain its related issues.

8. a) What is object oriented Integration?

8. b) What is Web testing?

→ Web testing are a few testing techniques to test web applications or websites for finding errors and bugs.

② A web application must be tested properly before it goes to the end-users.

③ Testing a web application does not only means finding common bugs or errors but also testing the quality related issues associated with application.

④ Basically there are four type of web-based testing that are available and they are:

- a) Static website testing
- b) Dynamic website testing
- c) E-commerce website Testing
- d) Mobile-based web testing

⑤ In web-based testing various areas have to be tested for finding potential errors and bugs, and steps for testing a web app are given below:

- a) App Functionality
- b) Usability
- c) Browser Compatibility
- d) Security
- e) Load issues
- f) Storage & database.

⑥ Example:

There are various examples of considerations that

need to be checked while testing a web application, some of them are:

- a) Do all pages are having valid internal and external links or URLs?
- b) Whether the website is working as per the system compatibility?
- c) What types of security does the website needed (if unsecured)?
- d) As per user interfaces - does the size of displays are the optimal and the best fit for the website?

9. a) What is Syntax testing and write in detail about its formats and test cases.

Syntax Testing, a black box testing technique, involves testing the System inputs and it is usually automated because syntax testing produces a large number of tests. Internal and external inputs have to conform the below formats:

- Format of the input data from users.
- File formats.
- Database schemas.

Syntax testing is performed to verify and validate the both internal and external data input to the system, against the specified format, file format, database schema, protocol and other similar things. Generally, syntax tests are automated, as they involve the production of large number of tests.

9. b) Illustrate logic based testing.

10. a) Write short note on :
1. Transition testing
 2. State testing
-

State Transition Testing is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test(AUT). State transition testing helps to analyze behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behavior.

It is the model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before, is a finite state system.

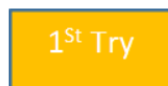
State Transition Testing Technique is helpful where you need to **test different system transitions**.

- This can be used when a tester is testing the application for a finite set of input values.
- When the tester is trying to test sequence of events that occur in the application under test. I.e., this will allow the tester to test the application behavior for a sequence of input values.
- When the system under test has a dependency on the events/values in the past.

Four Parts Of State Transition Diagram

There are 4 main components of the State Transition Model as below

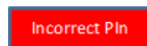
1) **States** that the software might get



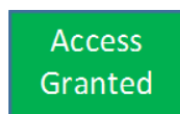
2) **Transition** from one state to another



3) **Events** that origin a transition like closing a file or withdrawing money



4) **Actions** that result from a transition (an error message or being given the cash.)



State transition diagram

