What is Software?

It is a set of instructions or programs used to execute a specific task.

What is Software Testing?

Software Testing can be defined as a **process** of verification (static testing) and validation (dynamic testing) to ensure that software meets business as well as technical requirements and desired quality.

Quality of Software should,

1) Meet Customer Requirement (Needs)

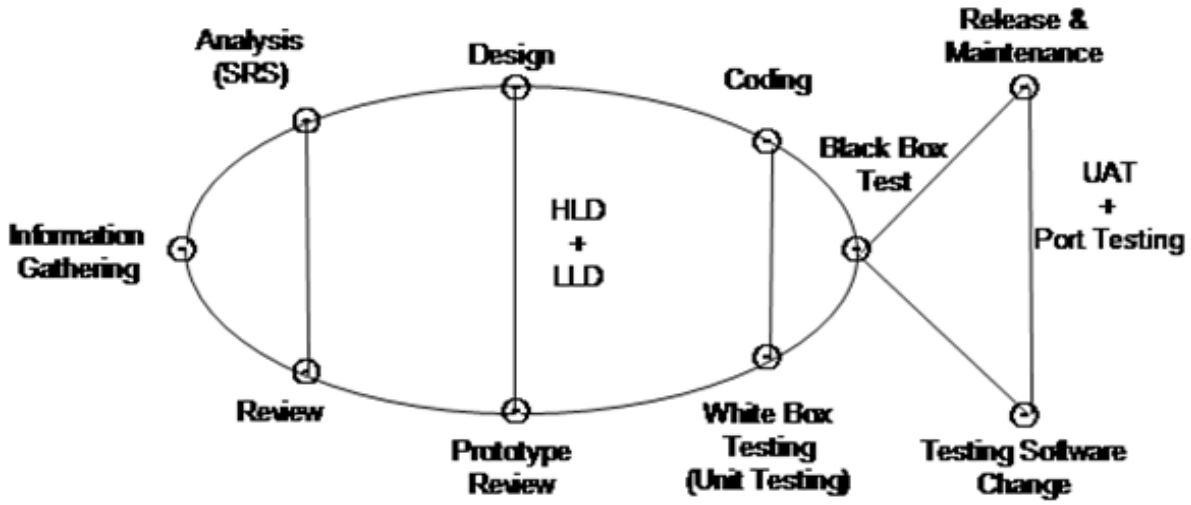
2) Meet Customer Expectations (Feeling, user experience, Performance etc.)

3) Cost of Software (affordable)

4) Timely Delivery (Time-to-Market)

5) Risk Management (Maintenance period)

Software Development Life Cycle (SDLC):



SDLC is a process which defines various stages involved in development of software for delivering high quality products.

The purpose of SDLC is to deliver high quality software as per customer requirement.

SDLC has defined phases like Requirement Gathering, Analysis, Design, Coding, Testing, Release & Maintenance.

1) Requirement Gathering

In this phase, Business Analyst is involved to gather relevant information from customers as per business needs. Following questions are asked by BA during this phase,

1. What is the business problem (problem statement)?
2. Why is this software required? How will it solve the problem?
3. Who are the targeted users? How will this impact them?

Business Analyst (BA) documents the requirement in BRS (Business Requirement Specification) format.

Business Analyst gets BRS reviewed & approved by the customer before passing to the next phase i.e. the analysis phase.

2) Analysis Phase

In this phase the Project Manager is involved. Project Manager receives BRS document as input from BA.

After receiving BRS document project manager does following activity-

i) Analyzing Requirements (Study)

ii) Feasibility Study (Check possibility)

iii) Decide Technology

iv) Effort estimation and planning

Project manager defines & documents the software requirements in the form of SRS (Software Requirement Specification)

SRS consists of all the software requirements to be designed and developed during the project life Cycle.

SRS can be defined as functional requirements to be developed as per customer needs and system requirements to be used.

The key people involved in the review of SRS are Project Manager, Business Analyst & Senior team members (Software Architect, Project lead, Test lead, etc).

The outcome of this phase is an SRS document which is reviewed by internal stakeholders as well as the client before moving to the next phase.

3) Design Phase

It has two steps High Level Design (HLD) & Low-Level Design (LLD).

High Level Design (HLD) - It defines architecture of software products to be developed & done by Software Architect.

Low Level Design (LLD) - It defines how each and every feature of software should work and how every component should work that is internal structure. After HLD & LLD, the prototype model is developed for client review before moving to the coding phase. Prototype is a blueprint of how the actual software will look like.

4) Coding Phase

In this phase software developers are involved.

This is a phase where we start coding for software.

Outcome of this phase is source code document & developed (working) software.

Developers also perform unit testing (using White box testing techniques) in this phase to verify internal working of the code.

5) Testing Phase

In this phase, a Software Tester is involved.

When the software is ready, it is sent to the testing team for complete testing.

The testing is done manually or using automated testing tools depending on the process defined in SDLC & ensure each and every feature of software is working fine as per customer requirements.

After completion of testing software is made ready for delivery or deployment for the customer.

6) Release and Maintenance

After Successful testing, software is delivered or deployed to customers for their reviews.

Deployment is done by the release team.

Once the customer starts using software then actual problems will come up & will need to be resolved from time to time as part of the maintenance phase.

Difference between Verification & Validation.

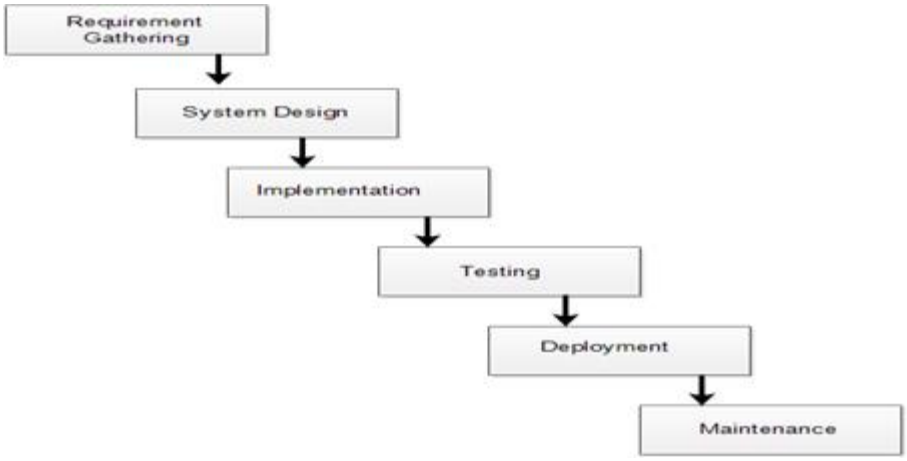
VERIFICATION	VALIDATION
Verification is a static practice of verifying documents, design, code and program.	Validation is a dynamic mechanism of validating and testing the actual product.
It does not involve executing the code.	It always involves executing the software.
It is human based checking of documents and files.	It is computer based execution of a program.
Verification uses methods like inspections, reviews, walkthroughs, and etc.	Validation uses methods like black box (functional) testing, grey box testing, and white box (structural) testing etc.

Verification is to check whether the software conforms to specifications.	Validation is to check whether software meets the customer expectations and requirements.
It can catch errors that validation cannot catch. It is low level exercise.	It can catch errors that verification cannot catch. It is High Level Exercise.
Verification is done by the development team to provide that the software is as per the specifications in the SRS document.	Validation is carried out with the involvement of the customer and testing team.
It, generally, comes first before validation.	It is generally done after verification.
Question Are we building the product right?	Question Are we building the right product?
Evaluation Items Plans, Requirement Specs, Design Specs, Code, Test Cases	Evaluation Items The actual product/software.

Interview questions:

1. What is SDLC?
2. Explain SDLC in detail.
3. What is the difference between verification and validation?
4. What is the difference between HLD and LLD?
5. What happens in the Information or Requirement Gathering phase?
6. What happens in the Analysis phase?
7. What happens in the Design phase?
8. What happens in the coding and testing phase?
9. What happens in the release and maintenance phase?

Waterfall Model



The Waterfall Model was the first process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and easy to use. In a waterfall model, each phase must be completed fully before the next phase can begin.

This type of software model is used for projects which are small & there are no uncertain requirements (i.e. requirements should be fixed).

At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.

In this model software testing starts only after development is completed i.e. phases do not overlap.

Advantages of waterfall model:

1. Simple and easy to understand and use.

2. Easy to manage due to the rigidity of the model.

3. Phases are processed and completed one at a time & they don't overlap.

4. This model works well for smaller projects where requirements are clearly defined & well understood.

Disadvantages of waterfall model:

1. Release takes longer and longer: Each release is taking more time, effort, and cost to get delivered to its customer.
2. Release schedule slip: Commitment to the customer is not met. If releases are not delivered on the promised time the plans are thrown in disarray and money and credibility is lost.
3. Stabilization at the end takes longer and longer
4. Planning takes too long and doesn't get right: Releases can take too long and because we didn't plan well enough at the start of the work. We didn't get our requirements firmed up and fully developed. To rectify it all more ideas are included and these are reworked. As a result a lot more time is spent in planning and release date is delayed.
5. Changes are hard to introduce mid release: The current process cannot accommodate change easily. Often something critical has to be included or a new feature is to be added. To incorporate this change we have to adjust all the work that

we have already done to accommodate it. This is difficult because it's hard to understand the ripple effect in the software.

6. Quality is deteriorating: Every time we might rush the development of the project, there is a possibility of compromising on the quality by missing defects and resulting in failure.

When to use the waterfall model:

1. Requirements are very well known, clear and fixed.
2. Product definition is stable.
3. Technology is understood.
4. There are no ambiguous requirements
5. Sufficient teams with required expertise are available.
6. The project is of short duration.

Interview questions:

1. What is a waterfall model?
2. Explain how the waterfall model works?
3. Advantages and disadvantages of the waterfall model?
4. When to use a waterfall model?

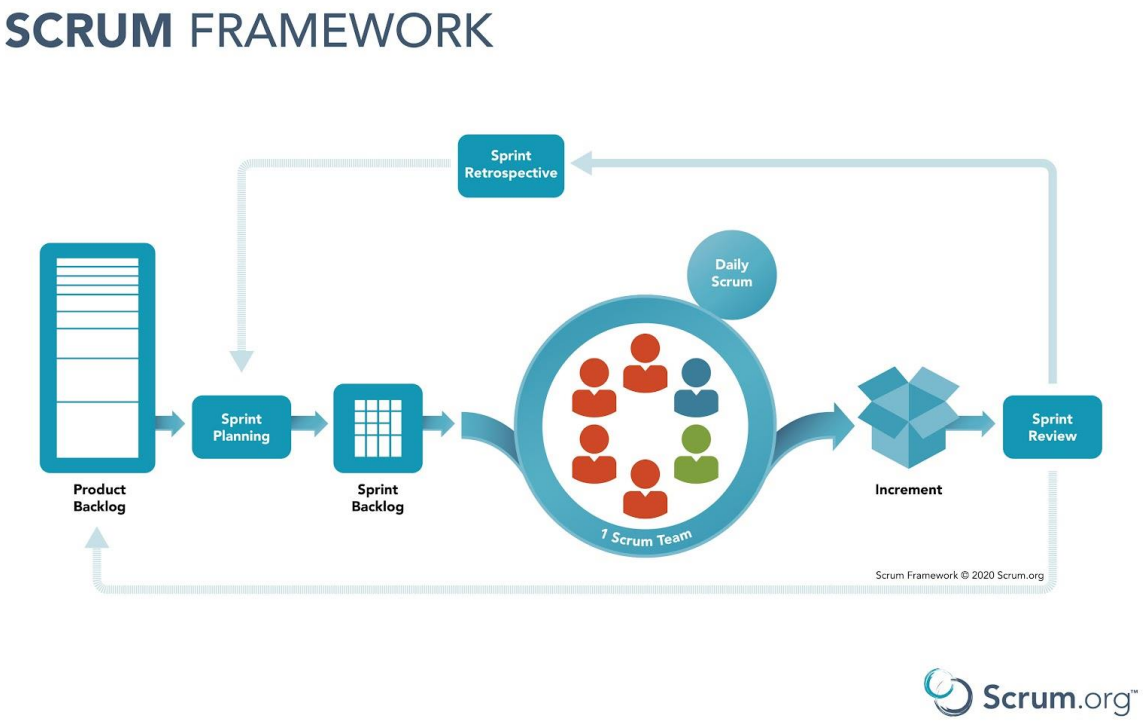
Agile Scrum Framework

https://agilemanifesto.org

https://agilemanifesto.org/principles.html

https://www.youtube.com/watch?v=9TycLR0TqFA

https://scrumguides.org/scrum-guide.html



Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.

Scrum is an agile process that allows iterative (repeatable) & incremental development by focusing on delivering the highest business value in the shortest time.

There are 3 roles (Product Owner, Scrum Master and Developers), 3 artifacts (increment, product backlog and sprint backlog) and 5 events (Sprint planning, Sprint, Daily Scrum, Sprint Review, Sprint Retrospective)

There are several agile methods like Scrum, Extreme Programming (XP), Kanban but Scrum is used mostly.

Agile Scrum Framework promotes iterative (repetitive) development throughout the life cycle of projects with close communication & collaboration.

It allows rapid delivery of working software (in 1 to 4-week time).

The Business team (i.e. Product Owner) sets the priority, teams are self-organized to determine the best way to deliver highest priority features as a result every 1 week to 4 weeks anyone can see real working software & decide to release it as it is or continue to enhance in next sprints.

Sprint is the heart of agile scrum methodology. Sprint is a container event which consists of other events like Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective.

Features are planned, designed, developed & tested during the sprint cycle.

During this process there is a 15-minute daily scrum held by developers and for the developers.

Product Backlog –

It is a kind of bucket of all requirements where all user stories (features) are maintained by the product owner. Product Backlog is created by the product owner who prioritizes it as per the business needs.

During sprint planning meetings brainstorming of user stories is done to understand & collectively decide which user story should be selected for the sprint backlog in consultation with the product owner

Epic/User Stories –

It is a feature which has to be developed by the Developers. In scrum methodology we don't have a user requirement document or detailed SRS instead, requirements are defined in short description in the product backlog in the form of user stories.

Example-

As a user, I want a Login Page in a net banking application which should have a password lock feature so that in case I enter an incorrect password three consecutive times then access should be restricted for 24 hours.

As a user, I want a Login page in a net banking application so that I can login and access my account.

As a user, I want to login into my LinkedIn account so that I can access my profile.

There are some characteristics for user stories as below,

1. User stories should be short, realistic, complete, negotiable & testable.
2. Sprint goal never changes in the middle of the sprint.

3. It is the responsibility of the scrum master & Development team to make sure the product owner has drafted the user stories correctly with a proper set of acceptance criteria.

Sprint Backlog –

Based on priority, user stories are taken from product backlog to sprint backlog.

The Scrum team brainstorms on it to determine feasibility in sprint planning.

Collectively the scrum team identifies the tasks to be developed in the sprint cycle. (UX design for login page, content, front end development, backend development, testing of login page functionality)

Sprint Cycle –

It has a predefined time frame in which work has to be completed and made ready for review.

Sprint duration can be between 1 to 4 weeks.

Scrum Master –

Scrum Master is the facilitator of the scrum team to ensure the scrum team remains productive and progressive.

In case of any obstacles scrum master follows up to resolve it for the team.

Scrum Master is the bridge between product owner and development team.

In case of any impediments & challenges the Scrum Master discusses it with the product owner & development team to enable them to find the solution.

Scrum Master makes sure the development team has a daily scrum meeting.

Burndown Chart (Metric)–

It is a graph which shows how much work is remaining on the basis planned vs. actual completion.

It is a tracking mechanism through which day to day tasks are tracked in line with sprint goals.

Story Point (Estimation metric) –

These are quantitative indications of complexity for user stories. Based on story points, user stories are estimated.

Story point is relative & not absolute (has to be realistic) to make sure our estimated efforts are smaller & precise, then user stories are determined.

User Stories will be evaluated & will have story point estimation.

Each & every user story is assigned a story point based on Fibonacci series, the higher the number the complex is user stories.

1 - 30 mins

2 - 1 hr

3 - 2 hrs

5 - 4 hrs

8 - 1 day

13 - 2 days

21 - 3 days

34 - 5 days

55 - 7 days

Sprint Velocity –

It helps in forecasting based on past sprint wise team efficiency. We take the average of past sprint based on their total story points worth of work completed.

Sprint	Story Points
1	120
2	110
3	130
--	
Total	360
Velocity (360 total storypoint/3 sprints)	120

Activity of Scrum Methodology (Explain)–

1. In Agile Scrum methodology, we start with a sprint planning meeting. It is a meeting where the entire development team, Scrum Master; Product Owner meets to brainstorm on product backlog items.

2. Based on discussion, the selection of user stories are estimated as per Fibonacci Series.

3. The team identifies the tasks that should be developed to implement user stories as per business value and priority.

Execution of Sprint–

This is where the actual work will be done by the development team to accomplish defined tasks & take user stories to done state.

Daily Stand-up-Meeting

During scrum cycle every day the scrum team meets for 15 minutes during daily scrum meeting team discusses on following points.

1. What did team member achieve yesterday?

2. What did team member planned to do today?

3. What are roadblock or impediments?

Sprint Review Meeting-

At the end of every sprint cycle the scrum team meets again & demonstrates the implemented user stories to the product owner and stakeholders.

The Product Owner may verify user stories as per acceptance criteria.

This meeting is attended by Scrum Master, Product Owner & development team as well as stakeholders.

In this meeting, the sprint increment will either be accepted or recommended for improvements.

Retrospective Meeting-

In this meeting, the scrum team meets & discusses to document the following points.

1. What went well during the sprint?

2. What didn’t work well during the sprint?

3. Lessons learned

Real Scenario –

1. We have a team of 9 “Developers” that will work on various aspects like -

UX (1), Front end (2), API development (1), Backend (2), Architect (1), Tester (2),

2. Sprint Duration is decided to 4 weeks. We have 1-month sprint starting from 26th November to 4th January

3. Product Owner will prioritize the list of user stories in the product backlog.

4. Team decides to meet on 26th Nov for a sprint planning meeting. Product owner takes 1 user story from the product backlog, describes it & leaves it to the team for brainstorming. The entire team discusses & communicates the direction to the product owner to have a clear understanding of user stories. In a similar way after user stories are taken the individual team member identifies individuals’ task for each story & calculate the estimated number of hours it will need to complete the task.

5. During planning meetings and final decisions of user stories are done.

The selected user stories are moved from product backlog to sprint backlog. For each user story, every team member declares their identified tasks & if required can have discussion on those tasks to size or resize it based on Fibonacci series.

The Scrum master or team enters their individual tasks for each story in a tool (e.g. JIRA). After all stories are completed Scrum master knows the initial velocity and finally starts the sprint.

6. Once the sprint gets started then the tasks are assigned to each team member working on those features.

7. Team meets daily for 15 minutes at a fixed time and place to maintain consistency (Daily Scrum).

8. Scrum master tracks the progress on a daily basis.

9. In case of any road blocks/challenges, scrum master takes the lead to resolve them.

10. On 4th Jan the team meets again for a sprint review meeting & demonstrates the user stories to product owners and stakeholders.

11. On 4th Jan team also meets for retrospective meeting

12. On 5th Jan the team meets for a planning meeting for the next sprint & this cycle continues until all user stories from product backlogs are cleared.

Advantages of Agile Scrum Methodology

1. Higher customer satisfaction due to rapid development model
2. Focus on people interaction rather than process and documentation.
3. Working software is delivered frequently
4. Flexible to adapt changes in requirements

Disadvantages of Agile Scrum Methodology;

1. Lack of focus on necessary design and documentation.
2. Project can go off track if the customer is not clear about the final output.
3. In case of large deliverables, it is difficult to estimate the required amount of effort at the beginning.
4. It requires expert people on the project.

Interview Questions:

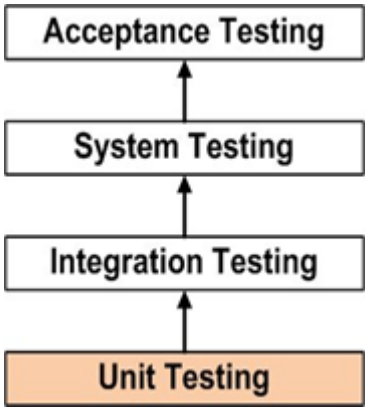
1. What is Agile Scrum Methodology?
2. What are the roles in Agile Scrum Methodology?
3. What are the artifacts and events in Agile Scrum?
4. Explain working of Agile Scrum Framework?
5. Advantage and Disadvantages of Agile Scrum?

Unit Testing –

A Unit is a smallest part of software like function, class, procedure or interface.

Unit testing is a method by which developer verifies working of the code as per design & requirement specifications to check whether the code is behaving as expected or not.

The objective of unit testing is to divide or isolate each part of the program and test individual part of the program. Developers will check a particular function by providing a set of input & by verifying corresponding output.



From above diagram unit testing is done before integration testing.

During Unit Testing developers use white box testing techniques to verify internal structure of code.

Following aspects are considered for unit testing;

1. Module Interface is tested to ensure data flows into & out of the program unit.

2. Local Data structure is tested to ensure data stored temporarily maintains its integrity.

3. Boundary Conditions are tested to ensure the module operates within established boundary limits.

4. All Independent paths are tested to ensure every program statement executes at least once.

5. All the error handling paths are tested.

Interview Questions:

1. What is unit testing?

2. Who performs unit testing? - Developer

3. When it is performed? – Before integration testing

4. What is tested during unit testing?- Developer verifies the internal structure of the program/code.

5. What is the scope of unit testing? What areas are covered?

6. How Unit Testing is performed? - Using White box testing techniques unit testing is performed.

White Box Testing Techniques–

White Box Testing is code level testing technique to verify internal structure of code.

There are different White Box Techniques –

1. Basis Path Testing Technique.
2. Control Structure Testing Technique.
3. Program Testing Technique.
4. Mutation Testing Technique.

These techniques are only applicable for programs.

Basis Path Testing Technique-

During this test, programmer concentrates on execution of program without any runtime errors.

To conduct this test, the corresponding programmer follows below approaches-

1. Write a program w.r.t. respect low level design (LLD).
2. Draw a flowgraph for that program.
3. Calculate cyclomatic complexity of that program.
4. Run that program more than one time to cover all executable areas.

Eg-

```
x = 10

y = 20

while x!=y:
```

```
    if x>y:
        x=x-y
        print(x)
    else:
        y=y-x
        print(y)
```

The above program should be executed 2 times to cover all executable areas.

Control Structure Testing Technique --

During this test corresponding programmers concentrate on correctness of program execution.

In this test they verify every statement of program execution i.e. using debugging.

Program Testing Technique-

During this test programmers concentrate on the execution speed of the program.

If the execution speed of a program is not reasonable then the programmer performs some changes in the code structure (refactoring) without disturbing functionality.

Mutation Testing Technique-

During this test corresponding programmers estimate correctness and completeness of program testing.

Mutation testing is performed after all other techniques are completed.

Interview Questions:

1. What is whitebox testing?
2. What are whitebox testing techniques?
3. Who performs whitebox testing and when they are applied?

Integration Testing --

It is a level of software testing where individual units are combined and tested as a group.

Testing performed to expose the defects in the interactions between integrated components or system is called integration testing.

The purpose of this testing is to expose defects or faults in the interactions between integrated units.

Integration testing is performed by developer (wrt connectivity) or by the independent testers (wrt integrity).

There are different approaches in Integration Testing

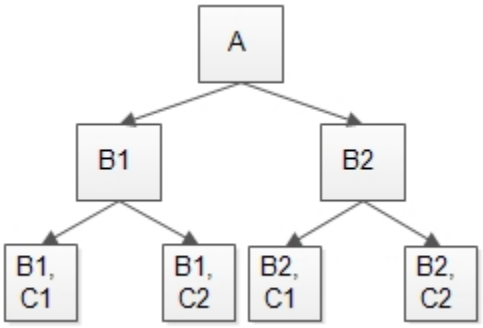
1. Top Down Approach
2. Bottom Up Approach
3. Hybrid Approach
4. System Approach (Big bang Approach)

Top down Approach-

This technique starts from the top-most module & gradually progresses towards lower modules.

Only the top module is tested in isolation, after this lower modules are integrated one by one.

The process is repeated until all modules are integrated and tested.



From above diagram testing starts from module A & lower modules B1 & B2 are integrated one by one.

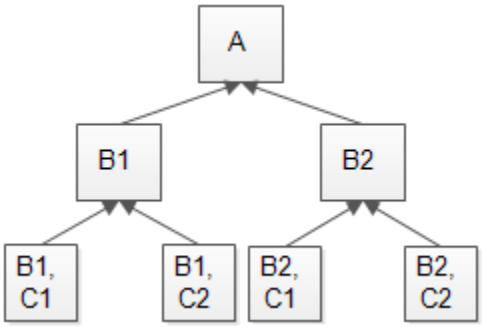
At this stage lower modules B1 & B2 are actually under development so in order to test topmost module A completely, we need to develop a temporary/dummy program called **Stub** to simulate functionality of the module which is under development.

Bottom up Approach –

As the name suggests, testing starts from the lowest or innermost units of the application & gradually moves up.

This Integration Testing starts from the lowest module & gradually progresses forward to upper modules of the application.

This integration continues until all modules are integrated & the entire application is tested as a single system.



From above diagram module B1C1 & B1C2 are lowest modules which will be unit tested.

Modules B1 & B2 are not yet developed & we need some program that will simulate functionality to call B1C1, B1C2, B2C1, & B2C2 modules.

The simulator program (dummy/temporary) is called Driver.

Driver is a dummy program which is used to call the function of the lowest module in case the calling function does not exist.

Hybrid Approach –

This is a combination of both top down & bottom up approach in order to check or test huge applications.

Integration starts with the middle layer & moves forward up and down.

System Approach –

The testing takes place only when all modules are developed & unit tested.

Steps for Integration Testing (Process)

1. Understand architecture of application.
2. Identify the modules.
3. Understand what each module does.
4. Understand how data is transferred from one module to another.
5. Understand how data is entered & received into the system.
6. Identify & create test conditions & test cases to execute.
7. Test the conditions one by one.

Sample Test Condition for Integration Testing

1. Test Condition 1:- Verify the interface link between login page & home page i.e. When a user enters valid credentials to login then he or she should be directed to the home page.

2. Test Condition 2:- Verify interface link between home page and profile page.

3. Test Condition 3:- Verify interface link between network page & connection page i.e. clicking accept button on network page, user should be able to accept the connection request & number of connections should update on connection page.

4. Test Condition 4:- Verify the interface between Amazon cart to payment gateway to check data (amount and notification (pass/fail)) exchange between both the application.

Note – Integration Testing can be performed manually as well as using tools like Rational integration testing, Protractor, Tessy.

Interview Questions:

1. What is integration testing?
2. Who performs integration testing?
3. When it is performed?
4. What do you check in integration testing?
5. What are approaches to integration testing?
6. Give examples of integration testing.

System Testing --

System testing means testing of all modules or components after integration in order to verify that system works as expected or not.

System Testing comprises Usability Testing, Functionality Testing, Non-Functionality Testing.

System Testing is done after Integration Testing and plays an important role to generate high quality software.

System testing is performed in an environment (test environment) similar to the production environment.

During the system testing application architecture & business requirements both are tested.

Following steps performed for system testing

1. Requirement Analysis & creating test plan (Test Lead creates)
2. Create Test Cases (manual) & Test Scripts (automation) – QA/Test Engineer
3. Prepare Test Data. (Test Engineer)
4. Execute System Test Cases & Test Scripts.
5. Report the defect if any & retest the bug post resolution by developer.
6. Perform Regression Testing to verify the impact of changes in code.
7. Repetition of the test cycle until the system is ready to be deployed.
8. Sign-off from the testing Team.

Sample Test Scenarios for System Testing:-

Test Scenario I --

Verify site launches with all relevant pages, features & logos.

Test Scenario II --

Verify User registration / Sign up functionality

Test Scenario III --

Verify Login to Site.

Test Scenario IV --

Verify if the user can see all the product available & can add product to cart, can do the payment & gets confirmation via email, sms, or call.

Test Scenario V --

Verify the major functionality like searching, filtering, adding, changing features are working as expected.

Test Scenario VI --

Verify if the site launches properly in all major browsers and their latest version.

Usability Testing --

Usability Testing makes sure that the interface of the application under test is built with respect to user expectations & requirements effectively & efficiently.

The primary focus is on,

1. Ease of use (UI)
2. Ease of learning and familiarizing with the system (User friendliness)
3. Satisfaction of user with entire experience (UX)

Note – Usability has got many dimensions to it. It is all about user experience during their interaction with the application & their feelings towards it.

Steps to perform Usability Testing:-

1. Identify the user to perform usability testing.
2. Designing the test cases that users are going to perform on the application.
3. Perform the usability testing.
4. Analyze the result.

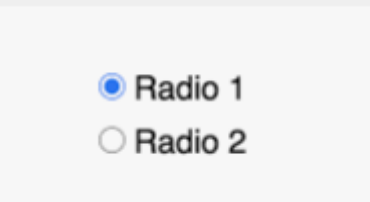
Functionality Testing:-

Functionality testing is performed to test the functionality of software whether it is working as per customer requirement or not.

There are different coverages in functionality testing:-

1) Behavioral coverage – To check objects (button, checkbox, radio button, links, drop down, etc.) property.

E.g. - Radio Button (On/Off)



2) Input Domain coverage -To check size and type of input. This will be applicable only for text fields.

There are 2 techniques;

a. Boundary Value Analysis.

b. Equivalence Class Partitioning

a) Boundary Value Analysis --

<u>In this we check size of input.</u>

E.g. – Saving account in bank has different rate of interest depending upon balance in account therefore rate of interest for balance between $1 to $100 will be 3%, $100 to $1000 will be 5% and above $1000 is 7%.

So Boundary value analysis is based on testing at boundaries between partitions.

Considering above example Boundary Values can be

$ 1.01-$100.00 $100.01-$999.99 $1000.01

b) Equivalence Class Partitioning --

It is also black box system testing technique.

In this technique input is divided into a set of test conditions or groups.

In the Equivalence Class Partitioning technique, we need to test only one condition from each partition because we can assume all conditions in one partition will be treated the same way.

Invalid	3%	5%	7%
-$1	$10	$150	$1050

For phone recharge following will be eg,

Valid	Invalid
0-9	A-z, Alphanumeric Special characters

3) Error Handling Coverage --

To check behavior of applications with incorrect operations i.e. negative testing.

4) Backend Coverage --

To check impact of front end screen operation on backend tables (Database Testing).

5) Service Level Coverage --

To check order of functionality i.e. correct sequence.

6) Calculation Based Coverage --

To check the outcome of arithmetic operations.

Steps for Performing Functionality Testing --

1. Determine the functionality of software features.

2. Create test data for functionality to be tested as per requirement specification.

3. Determine expected output.

4. Execute the prepared test cases.

5. Compare the expected and actual result.

Example test scenario and test case of airtel recharge,

TS 1:

To verify prepaid recharge of valid mobile number.

TC 1:

To verify prepaid recharge of a valid 10 digit mobile number with a plan of Rs. 10 using payment of valid credit card.

Non-Functionality Testing --

Non-functionality testing is done to verify non-functional requirements of the application like performance, reliability, compatibility etc.

It covers all aspects which are not covered in functionality testing.

The purpose is to cover testing for all extra characteristics of an application that meets business & technical requirements.

Eg.-

A Web application is developed & completely tested for functionality but non-functionality testing for the same is not done.

The application is now live (in use by customers) & faces major issues when the load is increased on the application.

It becomes too slow & takes a lot of time to open as a result customer is not satisfied with the product despite all functionality is working as expected.

It is important to capture non-functional requirements well in advance with proper documentation.

Non-Functionality Testing should capture,

1. User Story

2. Acceptance criteria (Max user logging at same time so check time to open page)

3. Artifacts (Test scenario & Test cases)

Non-Functional testing is part of black box testing.

Type of Non-Functional Testing --

1) Performance Testing --

Evaluate overall performance of the system.

The performance test is categorized in 2 types - Load Testing & Stress Testing

a) Load Testing --

Evaluate whether the system performs as expected under normal & expected conditions or not.

E.g. – Validate the systems performance when concurrent users access the application & get expected response time. This time of testing the database should be realistic & the test should be conducted on a dedicated server which simulates the actual environment.

b) Stress Testing --

Evaluate whether the system performs as expected when it is low on resources for e.g. slow internet speed, less memory etc.

E.g. – Test on low memory or low disk space on client / server applications because many time defects may not be found under normal conditions to check if multiple users can perform the same transaction on the same data.

During stress testing multiple clients are connected to servers with different work loads by reducing think time to zero for stressing the server.

Checklist for Performance Testing --

1. Response Time.

2. Interoperability (Ability to operate with other software)

3. Increasing Load.

4. Test Environment. (should similar to live/customer environment)

5. Throughput. (Number of transactions completed during load task)

6. Data Volume Testing. (Evaluate behavior of application when large volume of data is involved)

Eg -When software is expected to handle a large volume of data then check the limit whether software fails, maximum database size is created & multiple clients reach the database to create a large report.

Eg – If the app is processing a database to create a report then the data volume test should use a large result set & check if the report is printed correctly or not.

2) Compatibility Testing--

Evaluate whether the application is compatible with other hardware / Software with minimum & maximum configuration.

Generally, web application compatibility testing involves testing software on multiple operating systems & browsers with the same set of test cases that were used during functionality testing.

Browser Compatibility is categorized 2 types:

a. Cross browser compatibility - Different browsers.

b. Version browser compatibility – Different versions of browser.

3) Recovery Testing --

Evaluate whether the application recovers from abnormal to normal condition after any hardware or software failure.

Eg – Database process is aborted & permanently terminated.

Database, pointer, fields, keys are manually entered directly into the database & physically disconnect the connection, disconnect power, turn down router to check recovery after connection is restored.

4) Installation Testing --

Evaluate & confirm if the software is getting installed and uninstalled correctly.

5) Localization Testing --

It is done to verify application in different languages i.e. local language. The main focus is to check the content & GUI of the application.

6) Internationalization Testing --

It is done to verify if the application is working as expected across all language settings.

Interview Questions:

1. What is system testing?
2. Who performs system testing?

3. When it is performed?
4. What do you check in system testing?
5. What is usability testing?
6. What is functionality testing?
7. What is non-functionality testing?
8. What is compatibility testing?
9. What is the difference between load and stress testing?
10. How is system testing performed?
11. What do you check in usability testing?

User Acceptance Testing --

User Acceptance Testing (UAT) is final testing performed after system & integration testing are completed.

The software is tested by the end user or client to determine whether software meets essential user requirements or not.

The main purpose of UAT is to validate the software against business requirements.

It is conducted by end user or customer side people who are familiar with business requirements.

UAT is the last phase of testing that is carried out before software goes live (in a production environment).

UAT can be performed either by customer (or end user) or team selected by customer internally within the organization.

UAT

Alpha Beta

QA DEV BA Cust. Rep End User

(Controlled environment) (Uncontrolled environment)

UAT Process --

1. Gather acceptance criteria

All key business functionality to be validated should be discussed with customers to understand how much testing is required and what areas/features are going to be tested.

2. Identify scope of QA involvement

a) Assist in the testing --

QA involvement in this could be to assist UAT users about how to use the application or to assist them to record the result or bugs while they perform actual testing.

b) Perform UAT --

In this user or client will identify the area of application that they want to evaluate & evaluation itself will be performed by the QA (UAT) team.

Once done then the reports are presented to the client or User for them to make a decision on whether the results are satisfactory or not with respect to their expectations in order to accept the application under test.

The key steps of doing UAT as below --

1. UAT test initiation -Define testing approach for UAT.

2. Identify User who will perform testing

    a. Set up the UAT environment (beta environment)

    b. Identify the test data.

    c. Identify the support team required.

3. UAT test cases design

    a. Identify the business scenario to be validated.

    b. Identify the relevant test data.

    c. Scenarios are added in the test management tool.

    d. User access is requested and sorted.

4. UAT test execution

a. Test execution of business scenario are performed

b. Appropriate defects are raised in defect management tools (jira).

c. Defects retesting & regression testing are performed.

d. UAT test closure report is produced & go or no-go decision is discussed & recommended

Difference between System Testing & UAT

System Testing	UAT

1) SIT focuses on compliance to customer requirements. 2) Perform functional testing to validate functionality of application with respect to customer requirement. 3) SIT team will create test data to simulate business functionality. 4)SIT team will test in an environment as per system specifications.	1)UAT team focuses on key business scenario only 2)End to end key business Scenario based testing. 3) UAT team creates test data which will be real time or production like. 4) UAT environment is similar to the production environment.

Questions:

1. What is UAT?
2. Who performs UAT? Either customer side people/end user OR QA from the company on behalf of the customer.
3. Process of UAT? - 2 mains steps and 4 sub-steps
4. Steps involved in UAT? 4 activities/steps
5. Difference between System Testing and UAT?

Requirement Traceability Matrix (RTM) --

In RTM, we document the link (mapping) between user requirements to be developed and other artifacts (design document, code, test scenarios/test cases) to ensure that for each and every requirement an adequate level of testing will be covered (with ref to test scenarios/cases).

It is also helpful to determine which requirement has the most defects.

RTM helps in establishing the process by ensuring maximum test coverage which will be used to refer to the number of test cases run, pass, fail, or blocked for every requirement.

RTM helps to link requirements, test cases & defects accurately.

It is categorized in 2 types: -

1. Forward Traceability

Mapping user requirements with test cases to ensure every requirement is tested thoroughly (In Depth).

2. Backward Traceability

Mapping test cases with requirements to ensure the product being developed is on the right track.

Advantages of RTM --

1. RTM ensures complete test coverage by ensuring every requirement has been tested & covered.

2. In case of a change request from a customer for any specific requirement, we have to evaluate the impact of changes on test cases from RTM. We can evaluate the number of test cases to be modified & those which will have to be added newly from RTM.

3. It also shows requirements or documents are consistent.

Questions:

1. What is RTM? What are the objectives?
2. Why is it important?
3. What is the advantage of RTM?
4. Who creates RTM? Project Manager/Test Manager creates RTM during the requirement phase however it is maintained throughout the life cycle of the project by respective team members.
5. How do you know if the testing is completed?

Software Testing Life Cycle (STLC) --

STLC is a testing process which has steps to be executed in sequence.

In the STLC process each activity is carried out in a planned & systematic way.

Each phase of STLC has different goals and deliverables associated with it.

Testing is not just a single activity but it consists of a series of activities carried out methodologically.

1. Requirement Phase --

During this phase we analyze & study the requirements.

This phase also helps in identifying the scope of testing.

If any feature is not testable then we communicate during this phase so that mitigation strategy can be planned.

2) Planning Phase --

Technically the test planning phase is the first step of the testing process.

In this phase, we identify the activity & resources which would help to achieve testing objectives.

Test planning is done on the basis of requirement as well as test strategy & risk analysis.

3) Analysis Phase --

This phase defines what to be tested.

We identify the test conditions (scenarios) from requirement document i.e. SRS,

The identification of test condition is based on following factor:-

a. Level & depth of testing.
b. Complexity of software.
c. Project Risk.
d. SDLC process involved.
e. Test Management tool.
f. Knowledge and skill of the team
g. Availability of stakeholder.

The test condition (scenarios) should be written in an identical way by the test engineer because that increases test coverage & test cases will be written on the basis of test conditions.

4) Design Phase (Test conditions) --

This phase defines “How to Test?”

It involves following tasks

a. Create detailed test conditions.

b. Breakdown test condition in multiple sub conditions to increase the coverage.

c. Identify & create test data.

d. Identify & setup test environment.

e. Update RTM.

5)Implementation Phase --

The major task in this phase is to create detailed test cases.

Test cases are prioritized & identified to ensure which test cases will be part of the regression suite.

Carry out review of test cases to ensure correctness & finalize test cases before actual execution starts.

If your project involves test automation then identify the test cases for automation & create test scripts.

6) Execution Phase --

In this phase actual test execution (run) takes place.

Based on entry criteria execute the test cases & log the defects in case of any discrepancy between expected result & actual result.

Also update RTM to track the project progress for test cases.

7) Conclusion Phase --

This phase concentrates on exit criteria & reporting will be based on agreed protocol in the project.

There are different types of report

a. Daily Status Report.

b. Weekly Status Report.

8) Closure Phase:-

In this phase following task are covered

a. Check for completion of test cases whether all test cases are executed or not.

b. Check there are no defects open.

c. Create lesson learned documents.

Phase Name	Entry Criteria (when to begin)	Activity Performed	Deliverables (outcome)
Requirement phase	1)Requirement Specification documents, SRS & BRS available. 2)Application Design Document available. 3)User acceptance criteria document available.	1)Create list of requirements & get your doubt clear 2)Understand feasibility of requirements whether it is testable or not. 3) If the project requires automation then document the automation feasibility report.	1) Document requirement understanding 2)Test Feasibility Report & automation feasibility report.

Planning Phase	1)Updated requirement documents. 2)Test Feasibility Report. 3)Automation feasibility report.	1)Define scope of testing 2)Do risk analysis 3)Perform test estimation. 4)Determine test strategy & process. 5)Identify tools, resources & needs for any training. 6)Identify the environment	1)Test plan document 2)Risk mitigation documents. 3)Test estimation documents.
Analysis Phase	1)Updated requirement document. 2)Test plan documents. 3)Risk Documents. 4)Test estimation documents.	1)Identify detailed test scenarios.	1)Test scenario documents.
Design Phase	1)Updated requirement documents.	1)Detail test condition 2)Identify test data. 3)Create RTM.	1)Detail test condition document. 2)Updated RTM & Test coverage matrix.

Implementation Phase	1)Detail test condition document.	1)Create & review test cases. 2)Create & review automation scripts. 3)Identify test cases for regression testing & automation testing. 4) Identify and create test data & take sign off on test cases & scripts.	1)Test Case document. 2)Test data. 3)Test Scripts.
Execution Phase	1)Test cases and Test scripts	1)Execute test cases 2)Log defect. 3)Report the status.	1)Test execution report. 2)Defect report. 3)Test log. 4)Defect log. 5)Updated RTM.
Conclusion phase	1)Updated test case with results. 2)Test closure condition	1)Provide accurate results of testing. 2)Identified risks which are mitigated.	1)Updated RTM 2)Test Summary reports. 3)Updated risk management reports.

Closure phase	1)Test closure reports. 2)Test summary reports.	1)Do the retrospective meeting. 2)Understand the lessons learned.	1)Lesson learned document. 2)Test matrix. 3)Test closure report.

STLC interview questions:

1. What is STLC?
2. What are phases in STLC? Explain?
3. Why separate the life cycle for testing?
4. Explain each phase with activities performed?

Advanced Software Testing

Testing Terminology –

1. Test Strategy – It is a project level document developed by a test manager or project manager. Test strategy document defines testing approach to be followed by the testing team.

2. Testing Policy – Testing policy is a company level document and developed by management level people. Testing policy defines the objective of testing in a company.

3. Test Plan – Test plan defines - What to test?, When to test?, How to test?, Who will test?. It also defines a schedule to be followed by the testing team during testing. This is created by Test Lead.

4. Test Scenario – Test scenario defines functionality to be tested. Test scenarios are broad level conditions for test coverage.

5. Test Case – Test case defines unique test conditions to validate the software build in terms of functionality, non-functionality and usability aspect.

6. Test Log – It is the result of a test case in terms of passed, failed, blocked.

7. Error, Defect & Bug –

A mistake in coding is called an error.

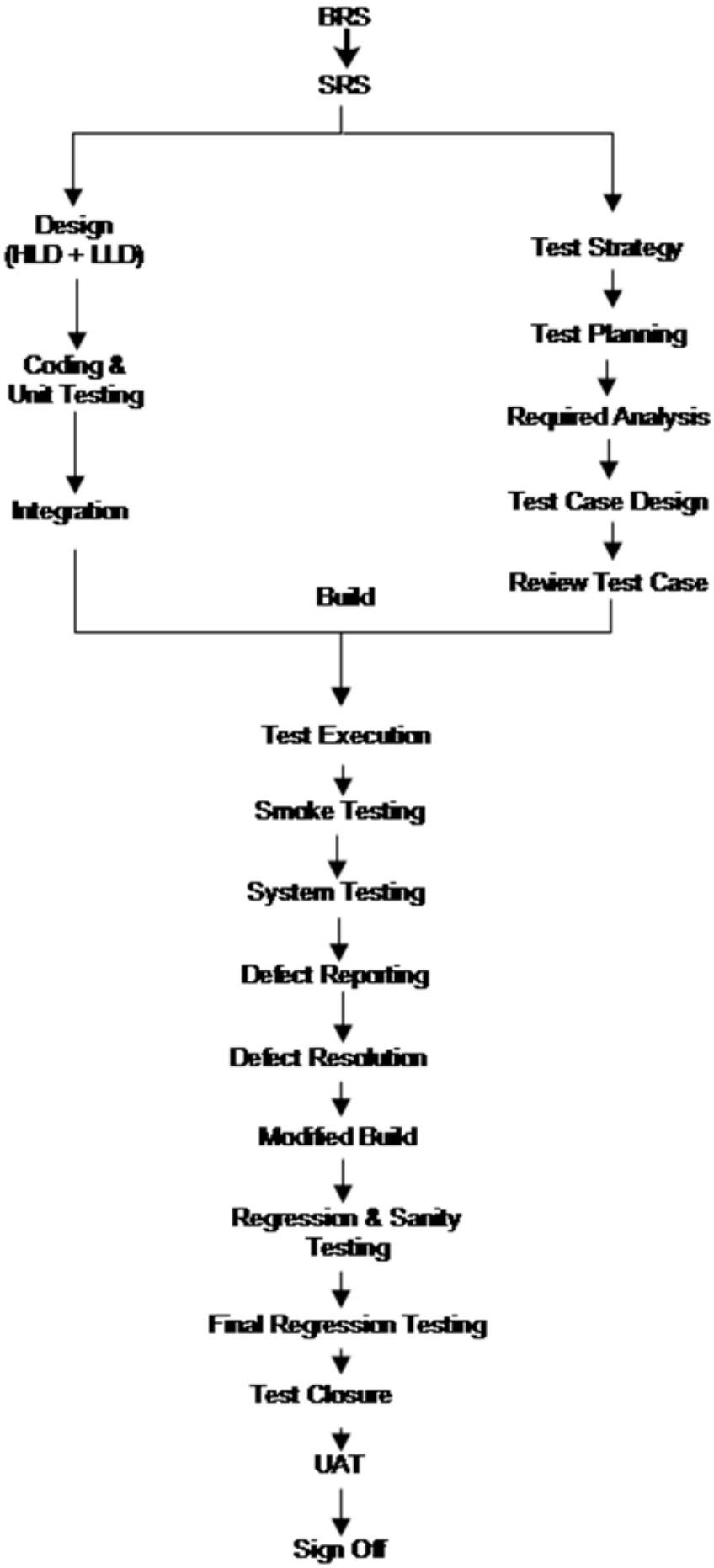
An error detected by a test engineer during testing is called defect.

Defect accepted by the developer is called a bug.

8. Summary Report – It defines work progress. There are different types of reports like DSR, WSR, MSR.

9. Test Bed – It is a combination of testing environment (combination of h/w & s/w) and testing information (Test data, Test scenarios, Test cases).

10. Test Suite (Test Set/Test Batch)– It is a combination of different test cases.


BRS
SRS
Design
(HLD + LLD)
Coding &
Unit Testing
Integration
Test Strategy
Test Planning
Required Analysis
Test Case Design
Review Test Case
Build
Test Execution
Smoke Testing
System Testing
Defect Reporting
Defect Resolution
Modified Build
Regression & Sanity
Testing
Final Regression Testing
Test Closure
UAT
Sign Off

Adhoc Testing –

In general, every testing team conducts planned testing but sometimes the testing team performs informal testing due to some challenges or risks.

E.g.- Lack of time ,Lack of resources ,Lack of team size etc.

This form of informal testing is called adhoc testing.

There are different types of adhoc testing

1. Monkey Testing
2. Exploratory Testing
3. Buddy Testing
4. Pair Testing
5. Defect seeding

Monkey Testing –

Due to a lack of time, the testing team concentrates on some of the main activities in software build for testing, this type of testing is called Monkey Testing.

Buddy Testing --

Due to lack of time management groups, programmer and tester as buddies.

Every buddy group consists of programmers and testers and the ratio is 1:1, 2:1, 3:1.

Exploratory Testing –

Due to lack of proper documentation for software being developed, test engineers depend upon past experience, internet or other social media platforms or discussions with other team members who have worked on similar type of project, contact customer side people if possible, this type of testing called exploratory testing.

Pair Testing –

Due to lack of knowledge on project domain, management groups, senior tester and junior developer conduct testing, this type of testing called pair testing.

Defect Seeding –

To estimate the efficiency of the test engineer, the programmer adds some bugs in the build to check whether the test engineer can catch the defect. This type of testing is called Defect Seeding.

Test Strategy --

Test strategy means “How are you going to test the application?”

It defines a testing approach to be followed by the testing team.

It is created by Project Manager or Test Manager.

It is very important to maintain the exact process or strategy that is going to be followed during testing of the application.

Test Strategy defines guidelines for testing approach in order to achieve test objectives and execution of testing types, defining test plan.

It deals with testing objective, approach, test environment, risk analysis along with risk mitigation plan.

Test Strategy document has following attributes –

1) **Scope and Overview** – Project overview along with information like who should use this document. It also includes details like who will review and approve this document.

Defines testing activities and phases to be carried out with timelines with respect to overall project timelines defined in test plan document.

2)**Test Approach** – Define testing process, level of testing, roles, responsibility of every team member for each type of testing defined in the test plan.

E.g. - System, Unit, Integration testing etc.

Describes why it should be conducted along with details like when to start, responsibility, testing approach, details of automation strategy tool if applicable.

In the test execution process there are various activities like adding new defect, defect triage (prioritize the defect), defect assignment, retesting, regression, final stage sign-off.

We must define the exact test followed in each activity.

Presentation of all these activities including number of testers, who will test and what activity will be helpful to identify and understand roles and responsibility for teams.

E.g.—In defect management cycle, maintain the process to log a new defect like When to log?, Where to log?, How to log a new defect?

Also define a change management process which includes defining change request submission, templates to be used and processes to handle change requests.

3)**Test Environment** -- Defines information about the number of Hardware and Software required to set up the environment.

E.g. – One test environment for SIT & another for UAT team.

Define the number of users supported on each environment, considering H/W, S/W, OS, Memory & number of systems etc.

Define test data requirements and also provide instruction on how to create test data.

Define test data backup and restore strategy.

Backup and restore process should define who will take backup? When to restore a database? and Data Masking steps to follow if the database is restored.

4)**Testing Tools** –Define Test automation tool required for test execution to perform functional, load and security testing , describe

testing approach and tools required and maintain how many users are supposed to access the application and plan according to it.

5) **Release Version** – Release management with proper version history should be defined.

E.g.- Set build management process which should cover following

a) When will the new build be made available?

b) Where should it be deployed?

c) From where to get production build

d) Who will give a go/no-go signal for production release?

6**) Risk Analysis** – List all possible risks and provide a clear plan to mitigate and also contingency plan in case risk arises.

7) **Review and Approval** – When all these activities are defined in test strategy, then get it reviewed and signed off by all entities involved in the project like PM, BA, Project Lead, QA lead, IT Admin etc.

Interview Questions:

1. What is Test Strategy?

2. Who creates Test Strategy?

3. What are attributes (fields) of a Test Strategy document?

Test Planning –

Test plan defines - What to test? When to test? How to test? Who will test?

After the test strategy phase is completed, Project Manager or Test Manager releases a test strategy document with all required details to test lead and then test lead concentrates on test plan preparation.

In this stage, the test lead prepares one system test plan and multiple detailed test plans.

The process of test plan preparation as follow,

<table>
<tr><th>Input</th><th>Process</th><th>Output</th></tr>
<tr><td rowspan="4">Project Document (BRS,SRS)

Test Strategy

Development Plan</td><td>Team formation</td><td rowspan="4">System Test plan</td></tr>
<tr><td>Identified Risk</td></tr>
<tr><td>Prepare test plan</td></tr>
<tr><td>Review test plan</td></tr>
</table>

Team Formation –

In general, the test planning process starts with testing team formation.

In this phase test lead depends upon following factors,

a) Project Size

b) Availability of test engineer

c) Test Duration

d) Availability of test environment resources

Identify Risk –

After completion of testing team formation test lead concentrates on risks at team level.

There are different type risk like

a) Lack of knowledge of the testing team on domain.

b) Lack of time

c) Lack of resources

d) Lack of documentation

e) Delay of delivery.

f) Lack of rigorous development process. (Lack of seriousness)

g) Lack of communication

Prepared Test Plan –

After testing team formation and risk analysis, the test lead prepares a test plan document.

Test plan document has following attributes –

1. Test Plan ID – Unique identification number or name.
2. Scope overview – Overview of testing required in project.
3. Test Items – Name of modules or features in the project.
4. Feature to be tested – Name of modules which are selected for testing.
5. Feature not to be tested – Name of the remaining module to be tested later.
6. Approach – Selected list of testing techniques with respect to test factors identified in test strategy document.
7. Test Environment – Required Hardware and Software to apply selected tests on specified features.
8. Entry Criteria –
    a. Prepared, Completed, Correct requirement documents like BRS, SRS should be available.
    b. Established Test Environment (Ready)

c. Receive stable build from development team.

9. Suspension Criteria –

   a. Test environment is not supporting,

   b. Showstopper (i.e. blocker without resolving blocker defect we can't start testing) defect occurred

   c. Pending defects are more.

10. Exit criteria –

   a. All modules are tested.

   b. Time duration is completed.

   c. All major defects are resolved.

11. Test Deliverables -- The names of test documents to be prepared by the test engineer.

   a) Test Scenario.

   b) Test Case Documents.

   c) Test Logs.

   d) Defect Reports.

   e) Summary Reports etc.

12. Staff and Training needs – Name of selected test engineers and required training session for them.

13. Responsibility – Mapping between test engineers and their available testing areas.

14. Risk & Assumptions – List of previously analyzed risks and assumptions.

15. Schedule – Date and time with detailed plan for each testing milestone.

16. Approvals – Sign-off from test lead

Review Test Plan –

After completion of test plan preparation test leads conduct a review meeting to check correctness and completeness of test plan documents by stakeholders.

During test plan review, review is conducted based on following factors

a) Requirement oriented review.

b) Testing technique-oriented review.

c) Risk oriented review.

Interview Questions:

1. What is the test plan?
2. Who creates the test plan?
3. When it is created?
4. What are the attributes of the test plan?
5. What process is followed by a test lead in test plan preparation?

Test Scenario –

The test scenario gives the idea of <u>what we have to test?</u>

Test Scenario is like high level test cases and covers what to be tested.

Eg.-

1. To verify the sign in functionality in LinkedIn web app
2. To verify the sign up functionality in LinkedIn web app
3. To verify forgot password functionality in Linkedin web app

Test Cases –

Test cases are a set of positive and negative conditions with executable tests for given scenarios to validate the software build in terms of usability, functionality and non-functionality aspects.

Test case covers How to test?

Test cases for testing login page functionality can be as below,

Test Case 1 – Verify linkedin login functionality with valid user name and invalid password.

Test Case 2 – Verify linkedin login functionality with valid username and valid password.

Test Case 3 -- Verify linkedin login functionality with invalid username and valid password.

Test Case 4 -- Verify linkedin login functionality with invalid username and invalid password.

Test Case 5 -- Verify linkedin login functionality using valid OTP.

Test Case 6 -- Verify linkedin login functionality using invalid OTP.

Test Case 7 -- Verify linkedin login functionality using Google account sign in.

Test Case 8 -- Verify linkedin login functionality using Apple account sign in.

Notes –

1. Test scenario guides user on what to test? and Test case guides user on how to test?

2. Purpose of the test scenario is end to end software testing and the purpose of the test case is to validate the test scenarios by executing a set of positive/negative conditions.

3. Test scenario helps in time sensitive situations (in Agile Scrum Methodology) & Creating test cases important when working testers are off-shore.

4. Test scenarios are derived from functional specification in SRS and Test Cases are derived from test scenarios/use cases.

Flow: SRS -> Test Scenario -> Test Cases

Test Case Design Techniques –

1. Dynamic Test Case Writing Technique.

2. State Transition Test case design Technique.

3. Error Guessing Test case designTechnique.

4. Field Validation Test case design Technique.

5. Boundary Value Analysis and Equivalence Class Partitioning Technique.

Dynamic Test case design Technique (Cause and Effect Graph) –

Cause and Effect Graph is a dynamic test case writing technique.

Causes are input conditions and effects are the result of those input conditions.

Cause-Effect graph is a technique that starts with a set of requirements and determines minimum possible test cases for maximum test coverage which reduces execution time and cost.

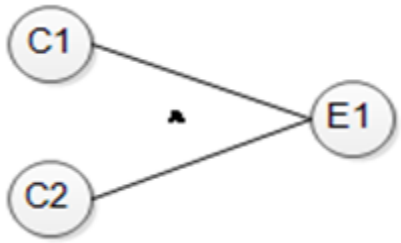
The goal is to reduce the total number of test cases by still achieving the desired quality by covering the necessary test cases for maximum coverage.

There are some challenges of using this testing technique because it takes time to model all your requirements into a cause effect graph before writing test cases.

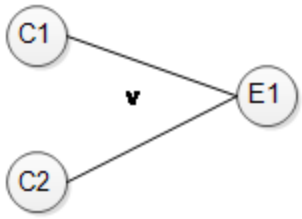
Cause and Effect Graph Technique states the requirement specification in terms of logical relationship between input and output condition.

Notation to use –

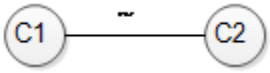
AND – For effect e1 to be true both causes C1 & C2 should be true.



OR – For effect e1 to be true either of cause C1 & C2 should be true.

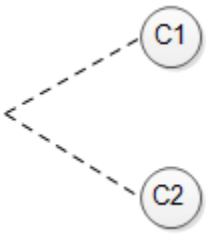


NOT – For e1 to be true C1 should be false.



E1

When only causes hold true, that is mutually exclusive.



Example—

Scenario – The “Print Message“ is a software that reads two characters and depending on their value message must be prepared. The first character must be “A” or “B”, Second character must be a digit.

If the first character is “A” or” B” and the second character is digit then the file must be updated.

If the first character is incorrect then message X should be printed.

If the second character is incorrect then message Y should be printed.

Solution –

The causes for this situation are

C1 a First character is A.

C2 a First character is B.

C3 a Second character is a digit.

The effect for this situation are

E1 for Update File.

E2 for Print message “X”.

E3 for Print message “Y”.

Effect 1 –

Effect 1 is to update the file and file is updated when,

a) First character is “A” and Second character is a digit.

b) First character is “B” and Second character is a digit.

c) Either “A” OR “B” or cannot be both

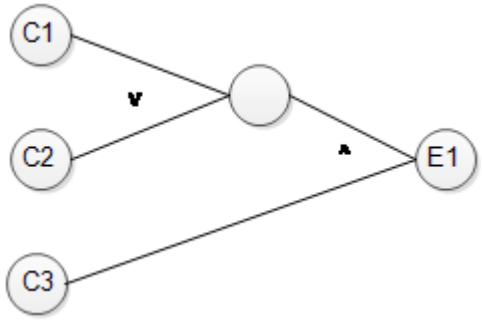
so for e1 to be true following are causes

i. C1 and C3 should be true.

ii. C2 and C3 should be true.

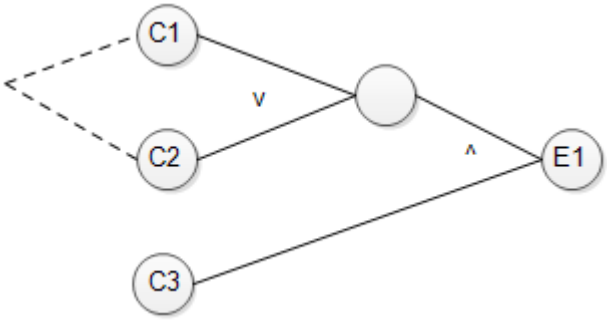
iii. C1 and C2 cannot be true together, which means C1 and C2 are mutually exclusive.

For e1 to be true the condition is (C1 V C2)^C3



For third condition where C1 and C2 mutually exclusive final graph of Effect1

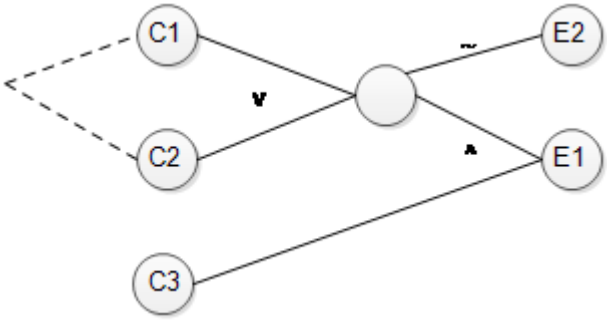
Is as below



Effect 2 –

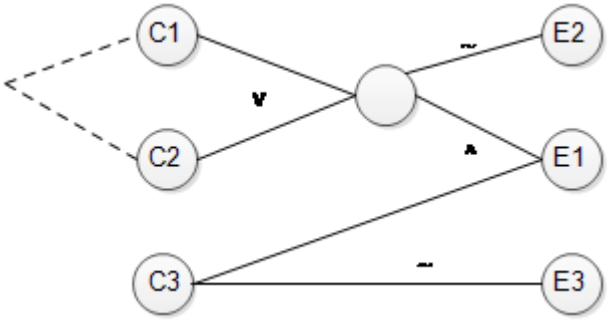
E2 states the print message “X”

Message “X” will be printed where first character neither “A” or “B” which E2 will hold true when either C1 and C2 invalid



Effect 3 –

E3 states to print message “Y” and message “Y” will be printed when the second character is incorrect which means E3 will hold true when C3 is invalid so the graph will be as below.



This is the complete cause-effect graph for the above situation.

Let’s see how to draw a decision table from the above graph.

Step 1 – Write down cause and effect in single column

Actions

C1

C2

C3

E1

E2

E3

Step 2 – Go from bottom to top means effect to cause in this case start with effect E1.

For E1 to be true the condition is (C1 v C2) ^ C3

Therefore they represent true as 1 and false as 0.

Actions

C1	1	0
C2	0	1
C3	1	1
E1	1	1

E2	0	0
E3	0	0

Step 3 – for E2 to be true either C1 and C2 has to be false.

Actions				
C1	1	0	0	0
C2	0	1	0	0
C3	1	1	0	1
E1	1	1	0	0
E2	0	0	1	1
E3	0	0	0	0

Step 4 – For E3 to be true C3 should be false.

Actions	TC1	TC2	TC3	TC4	TC5	TC6
C1	1	0	0	0	1	0
C2	0	1	0	0	0	1
C3	1	1	0	1	0	0
E1	1	1	0	0	0	0
E2	0	0	1	1	0	0
E3	0	0	0	0	1	1

Write test cases from the above situation for the decision table.

Test Case ID	Test Case Name	Test Description	Steps	Expected Result
TC1	TC1_File_Update_scenario 1	Validate that system updates file when 1st character is “A” and 2nd character is digit.	1.Open the application. 2.Enter the 1st character as “A”. 3.Enter 2nd character as digit.	File Should be updated.
TC2	TC2_File_Update_scenario 1	Validate that system update file when 1st character is “B” and 2nd character is digit.	1.Open the application. 2.Enter the 1st character is “B”. 3.Enter 2nd character as digit.	File should be updated.

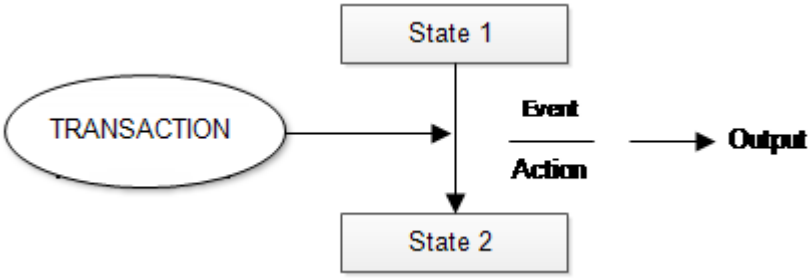
2) State Transition Test case design Technique ---

State transition testing technique is black box testing technique which can be applied to test finite state machines.

State transition technique is a dynamic testing technique which is used when the system is defined in terms of a finite number of states and

transitions between states are governed by the rule of the system. In short, this technique is used when features of a system are represented as states which transform into one another.

Transformation determines the rules of software.



From the above diagram we can see that an entity transitions from state 1 to state 2 because of some input conditions which leads to an event and results in an action and finally gives an output.

Eg – When we visit an ATM and withdraw 1000 rs successfully and the user runs out of balance and makes exactly the same request withdrawing another 1000 rs, this time ATM refuses to give money because of insufficient balance so here transition which causes change in state is the earlier withdrawal.

So state transition technique is a kind of black box testing in which the tester has to examine the behavior of the application under test against various input conditions given in sequence.

The behavior of the system is recorded for both positive and negative values.

State Transition Technique can be used in following situation

1. The application under test is a real time system with different states and transitions.

2. When application is dependent upon events/value/condition of past.

3. When the sequence of events needs to be tested.

4. When applications need to be tested against a finite set of input values.

Note –

This technique is useful when

1. Testing is not required for sequential input combinations.

2. Different functionality of applications are required to be tested like exploratory testing.

In practical scenarios testers will be given a state transition diagram by BA and testers are required to interpret those to determine test cases.

For e.g.

To verify change in state of user account to “Active” post renewal for the account that has been deactivated.

Test Case ID	Test Case Name	Test Description	Steps	Expected Result
TC1	TC_Account_Renew	To verify change in state of user account to “Active” post renewal of the account that has been deactivated.	a. Login to the system to recharge phone number. B. Recharge the phone number with Rs. 299 plan.	Recharge should be successful and phone number should be activated.
TC_2	TC_Amazon_PurchaseItem	To verify end to end purchase of item that involves placing an order to successful	Step 1: Add item to cart Step 2: Place an order	Item should be added to cart correctly and upon successful payment confirmatio

		confirmation	Step 3: Confirm address Step 4: Make a payment using credit card	n should be received on email/SMS.

3) Error Guessing Technique –

Error guessing is testing software by guessing errors which can reveal defects in the code i.e. basically an experience based testing technique where testers use their experience to guess problematic areas of the application. It is also a type of business to business testing technique.

The test cases for finding issues in software are designed based on prior testing experience with similar applications.

Error Guessing technique does not follow any testing tools.

Eg- Tester feels login page is error prone then tester will write detailed test cases concentrating on login page.

This technique can be used in any level of testing and testing common mistakes like

a. Divide by zero.

b. Enter blank space in the text field.

c. Press the submit button without entering the value.

d. Uploading file exceeding maximum limit.

e. Null pointer exception.

f. Invalid parameters.

The main purpose of this technique is to make an educated guess of possible bugs in areas where formal testing would not work.

It should achieve an exclusive set of testing without any skill in areas.

Factor can be used to guess the errors

1. Lesson learned from past releases/projects.
2. Tester intuition.
3. Historical learning.
4. Previous defect.
5. Review checklist.
6. Application UI.
7. Previous test results.
8. Variety of data used for testing.
9. General testing rules.
10. Knowledge about UAT.

Note –

This technique should perform once formal testing techniques are completed.

4) Field Validation testing technique —

This is one of the test case design techniques for validating text fields in application.

This technique is mainly used where there is field validation required.

Generally each and every field of application is validated thoroughly to identify defects which may get unnoticed in the field.

Implementation of Field Validation Table –

Step 1 –

Create a table from different data types for valid and invalid inputs.

Data Type	Valid Input	Invalid Input
Integer or Number	1.Only numbers. 2.Less than limit (n). 3.Enter Value within limit(n+1/2)	1.More than limit(n+1). 2.Number with precision. 3.Number in exponential form. 4.Negative integer. 5.Only Alphabets. 6.Number plus alphabets. 7.Number plus special characters. 8.Unicode characters.

String	1.Only alphabets. 2.Only numbers. 3.Only special characters. 4.Numbers plus alphabets. 5.Number plus special characters. 6.Alphabets plus special characters. 7.Enter Value within limit.	1.More than limit(n+1). 2.Unicode characters.
Date	1.Check if the date picker is present or not. 2.Check date field not in table. 3.Ensure that right click on date field paste option is disable and Copy option should enable. 4.Ensure picking date in calendar it should in date field. 5.Ensure that calendar having provision for any year/month.	NA

Note –

Testers should keep a field validation table in front of them before proceeding with testing the input fields in an application.

Step 2 –

Application specific tables should be created with specific fields and columns to validate each and every field in application.

Eg –

Serial No	User story no. and Requirement ID	Test case ID	Feature	Session	Field type	Data Type	Field Size	Result
1	123	Logic_TC1	Login	User ID Password Submit Cancel	Text Text Button Button	Alphanumeric Alphanumeric NA NA	15 10	Pass Pass Pass Pass
2	200	Recharge_TC1	Prepaid recharge	Mobile number	Integer	Numeric	10	Pass

Test Case Attribute or Format –

Test Case ID – Unique id for each test case.

Test Priority – (LOW/MEDIUM/HIGH) Test priority for business rules and functional test cases can be medium or high whereas minor usability test cases can be of low priority.

Module Link – Mention name of main module or submodule.

Test Case Designed By – Name of Tester.

Test Designed Date – Date when it was written.

Test Executed By – Name of test engineer who executed this test.

Test Execution Date – Date when test is executed.

Test Title/Name – Test case title.

Test Summary / Description – Describe test objective in brief.

Precondition – Any prerequisite that must be fulfilled before execution of the test case.

Test Steps – List all test execution in detail, write the steps in order in which they should be executed.

Test Data – Use of test data as an input for this test case. We can provide different data sets with exact values used for input.

Expected Result – What should be expected system output after test execution? Describe expected result in detail including message/error that should be displayed on screen.

Actual Result – Actual result fill after test execution. Describe system behavior after test execution.

Status (Pass/Fail) – If actual result not as per expected then mark test case as failed otherwise update as passed.

· Comment – If there are some special conditions to support the above fields then mention them there.

Difference between Test Scenario and Test Case –

Test Scenario	Test Cases
1.Test Scenario gives idea what to test	1. Test case are set of positive and negative test which has set of attribute like test description, precondition, data, expected result, actual result, status.
2. Test scenario like high level test cases i.e what to be tested for e.g. – Verify login functionality.	2. Test case answer How to test? E.g. -To test login functionality test case can be Test Case 1 – Verify login functionality with valid user name and invalid password. Test Case 2 – Verify login functionality with valid username and valid password. Test Case 3 -- Verify login functionality with invalid username and valid password. Test Case 4 -- Verify login functionality with invalid username and invalid password.
3. Purpose of the Test Scenario is to test end to end business functionality of software applications.	3. Purpose of the Test Case is to validate the Test Scenario by executing a set of test conditions.

4.Creating of test scenario helps in time sensitive condition (in Agile Development)	4) Creation of Test Cases important when testers are working off-site.
5. Test Scenario easy to maintain because of high level design.	5. In test case software application changes often it leads to redesigning the test case and adding new functionality its hard to maintain test cases.
6. Less time consuming.	6. More time consuming.
7. Requires less resources to create and Test using test scenario	7. Requires more resource to create and execute test case.
8. Helps in agile environment to test end to end functionality.	8.It helps exhaustive testing of application.(deep/thorough testing)
9. Test scenario derived from use cases in SRS or requirement document.	9. Test cases are derived from test scenarios.

Note – Using Test Scenario and Test case together will ensure maximum test coverage. It helps to write test scenarios and test cases and then move on to test execution. However in the Agile way of development most of the companies prefer test scenarios to save time.

Interview Questions:

1. What is the test scenario and test case?

2. from where do you write test cases? From Requirement specification.

3. What are test case design techniques?

4. What are test case attributes?

Example requirement specs. for test case writing

Requirement specifications for LinkedIn modules

1. User Registration Module:

· The system shall provide a user registration page where new users can create an account.
· The registration page shall include fields for email address, password, name, and other required information.
· The system shall validate the email address format and ensure it is not already registered.
· Upon successful registration, the system shall send a confirmation email to the user.
· The user should be able to access the LinkedIn platform using the registered credentials.

2. Profile Editing Module:

· The system shall allow users to edit their profile information, including name, headline, location, education, and work experience details.
· Users should be able to upload and update their profile picture.
· The system shall validate and store the updated profile information in the database.
· Users should have control over their privacy settings and be able to customize the visibility of their profile information.

3. Connection Module:

· Users should be able to connect with other LinkedIn users by sending connection requests.
· The system shall provide a search functionality to find other users based on name, industry, location, or other criteria.
· Users should receive notifications for new connection requests and be able to accept or reject them.
· Users should be able to view and manage their connections, including sending messages and accessing their profiles.

4. Job Search Module:

· The system shall offer a job search feature that allows users to search for job postings based on keywords, location, industry, etc.
· Job listings should display relevant details such as job title, company, location, and a brief description.
· Users should have the ability to save job listings for later viewing or apply directly from the platform.
· The system shall facilitate the submission of job applications, including the option to attach resumes and cover letters.

5. Groups Module:

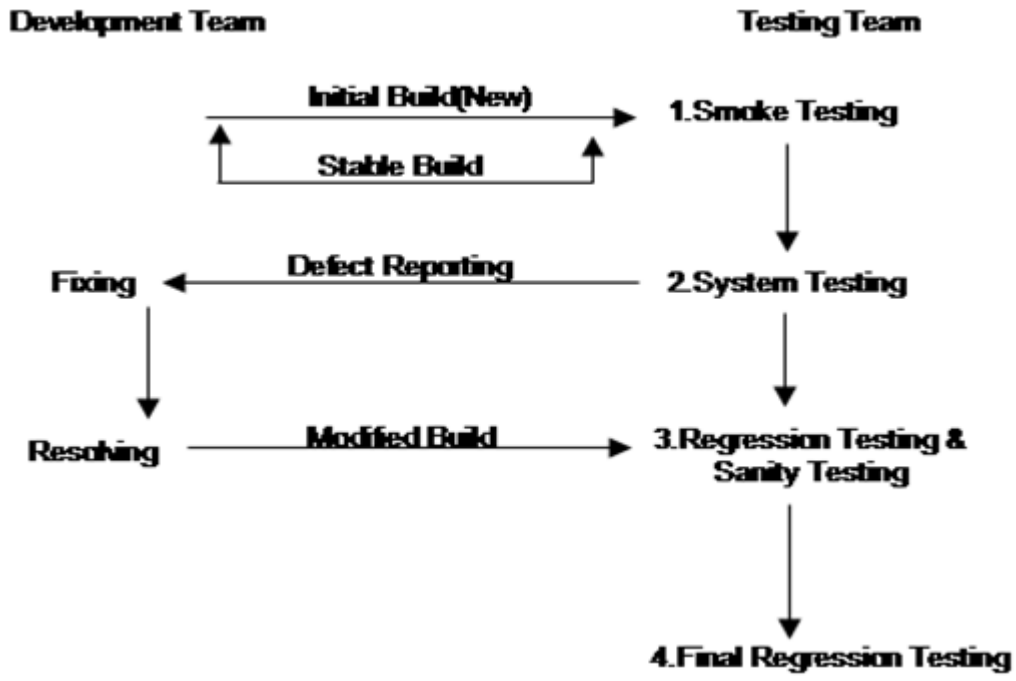
· Users should be able to join professional groups related to their interests or industry.
· The system shall provide a group search functionality to find and explore different groups.
· Users should have the ability to participate in group discussions, post articles or announcements, and comment on other users' contributions.
· Group administrators should have the ability to manage group membership, moderate discussions, and enforce group rules.

Test Execution –

Test execution phase consists of executing the test conditions (Test Cases for manual and test script for automation testing).

Therefore when planning this phase schedule and effort should be estimated taking into account all this aspect and not just script execution.

Level of Test Execution –



Smoke Testing –

Smoke Testing is not an executable testing but it is a group of tests that have to be executed to verify the basic functionality of the build is working fine as expected or not.

This will always be the first test to be done on any build.

If the smoke test is successful and passed then only the build is released to the QA team for the next level of testing i.e. system testing.

If the smoke test result is not successful then the build is rejected and the development team needs to fix the issues and release the revised build for testing.

System Test Execution–

After receiving the stable build from the development team and successful smoke testing, the testing team performs exhaustive (Thorough/complete) testing that is system testing (consisting of usability, functionality and non-functionality test cases).

During this process if the testing team comes across any mismatch between expected result and actual result the testing team reports those defects to the development team.

Development team then performs fixes on build coding and resolves defects to release modified builds to the testing team.

This process is followed until all the test cases are executed and closed.

Regression Testing –

Regression means retesting the unchanged part of the application.

Regression testing is done to verify that code changes in software does not impact existing functionality of the application, to make sure that software works fine as previously with newly added functionality or any changes in existing feature or bug fixes done by the developer.

In regression testing we execute the existing test cases, to verify impact of change.

In regression testing, test cases are generally automated because test cases are required to be executed again and again for every round of tests and running test cases again and again is time consuming.

E.g. – Consider an E-Commerce application in which one of the functionality is to trigger confirmation, acceptance and Notification when Confirm, Accept and dispatch Button are clicked.

Scenario –

Some issues occur in confirmation emails/messages and in order to fix the same, some code changes are required.

In this case not only confirmation email/message need to be tested but also acceptance and dispatch message need to be tested to ensure that change in the code has not affected them.

Regression testing is not depending upon any program instead it is a method used to test software for modifications/updates done to the software.

It verifies any modification in the software does not affect existing features of software.

Regression tests should be part of the release cycle and must be considered in test execution.

When to perform regression testing –

Regression testing is usually performed after verification of changes in a new software build.

However, for a release that is taking a month, regression testing must be incorporated in the daily/weekly cycle.

For weekly releases, regression testing can be performed when functional testing is over for the new changes.

How much regression testing is required?

Depends upon newly added features.

If the scope is fixed and the feature is too large then the application area gets affected which requires thorough testing but this is decided when the testing team gets input from developers about scope, nature and amount of changes in the build.

These are repetitive tests therefore test cases can be automated so that a set of test cases alone can be easily executed on a new build.

Regression test cases need to be selected very carefully so that maximum functionality is covered in the minimum set of coverage (Test Cases).

This set of test cases needs continuous improvement for newly added functionality.

The selection of test cases are based on the environment, changes done to the software and parts where it can affect the most.

Note - Best practice is to run automated test cases every day in the evening so that any regression side effect can be fixed in the new build.

Regression Testing Techniques --

There are four techniques

1) Retest All –

As the name suggests, entire test cases in the suite are re-executed to ensure that there are no bugs that occurred because of a change in code.

This is an expensive method because it requires more time and resources.

2) Regression Test Selection –

In this technique test cases selected from the test suite are re-executed, not the entire suite needs to be re-executed.

Selection of test cases is done on the basis of code changes in the build.

Test cases leads into 2 categories—

a) Reusable Test Cases – It can be used for future regression cycles.

b) Absolute Test Cases – It can be used for the upcoming regression cycle.

3) Test case prioritization –

Test cases with high priority will be tested first then medium and low priority.

Priority of test cases depends on criticality and impact on product as well as functionality of product.

4) Hybrid –

This technique is the combination of regression test selection and test case prioritization that means only selected test cases which are to be re-executed and depending on their priority.

Selection on regression test suite –

Most of the defects found in the production environment occur because of changes performed or defects fixed at a later stage.

The defect fixed at the last stage might introduce issues in the software.

The selection of regression test cases is very simple, following are some examples of test cases that can be used or selected for regression testing.

a) **Functionality which are frequently used.**

b) **Test cases which cover the module where changes have been performed.**

c) **Complex functionality test cases.**

d) **Integration test cases that include all major components.**

e) **Test cases for the core functionality or feature of software.**

f) **Priority 1 and Priority 2 test cases.**

g) **Test cases which have been frequently failed.**

Steps to perform Regression Testing –

1. Prepare test suite (test batch/test set) for regression testing considering the point mentioned in how to select regression test suite.

2. Automated all test cases for the regression test suite. (Optional)

3. Update regression suite whenever it's required.

E.g.- If a new defect which is not covered in existing test cases is found then the test case for the same should be updated in the test suite.

4. Execute regression test cases whenever any changes in code, bug fixed, new functionality added, enhancement done in the existing functionality.

5. Create a test execution report which includes pass/fail status of executed test cases.

Regression Testing (Release 1)—

Application Name	XYZ Net Banking.
Version Number (Build)	1.0
Number of requirement(Scope)	10
Number of test cases	100
Number of days to create test cases	10
Number of days to execute test	5
Number of tester	2

Release 2 –

Application Name	XYZ Net Banking.
Version Number Build	2.0
Number of requirement(Scope)	10 + 5(New requirement)
Number of test cases	100 + 50
Number of days to create test cases	5
Number of days to execute test	5 + 2.5
Number of tester	2

Release 3 –

Application Name	XYZ Net Banking.
Version Number	3.0
Number of requirement(Scope)	10 + 5 + 5
Number of test cases	100 + 50 + 50
Number of days to create test cases	5
Number of days to execute test	5 + 2.5 + 2.5
Number of tester	2

Note 1 –

1. Understand what kind of changes are made in software.

2. Analyze and determine what modules of software might be impacted (Developer/BA can provide this information).

3. Review the test cases to determine if you have to do full, partial or unit regression.

4. Plan and test the regression build.

Note 2 –

1. Regression testing in Agile software development –As we know agile is an iterative and incremental framework, the software is developed in short iteration called a sprint of 1 to 4 weeks.

Because of the number of iterations, testing is very critical as new functionality or code changes in every iteration.

Regression testing in agile is covered in 2 categories –

a) Sprint level regression—

Sprint level regression is done for the latest sprint.

Test cases from the test suite are selected as per newly added functionality or enhancement.

b) End to End regression –

This includes all test cases re-executed to test complete software end to end by covering all core functions of the product. As agile has a short sprint duration the test cases execute again and that needs to be completed in a short duration of time therefore automation test cases reduce time of execution and defect slippage.

Interview:

1. What is regression testing?

2. When it is performed?

3. How do you select test cases for Regression Testing?

4. What techniques are there for selection of test cases for regression testing?

5. What are the steps to perform regression testing?

6. What is the difference between regression testing and retesting?

Defect Reporting –

What is the defect? – Defect is caused when there are any errors made in the coding and caught by the tester.

When a tester during test execution reports a mismatch between expected result and actual result then it is reported as defect.

Defect report has following attribute—

1. Defect ID -- Unique identification of defect (System generated).

2. Project Name

3. Product Name

4. Build Version – Release version of the build 1, 2, 3.

5. Module Name – Specific module of an application where defects are detected.

6. Summary –Summary of defect should be clear and concise.

7. Description – Detailed description of defect should be simple and comprehensive.

8. Steps to replicate – Step by step description of the way to reproduce defects.

9. Expected Result

10. Actual Result

11. Severity – Severity of defect can be High, Medium, Low. Severity means seriousness of defect with respect to functionality

12. Priority – Priority of defect as High, Medium, Low. It defines the importance of defects with respect to customer requirements.

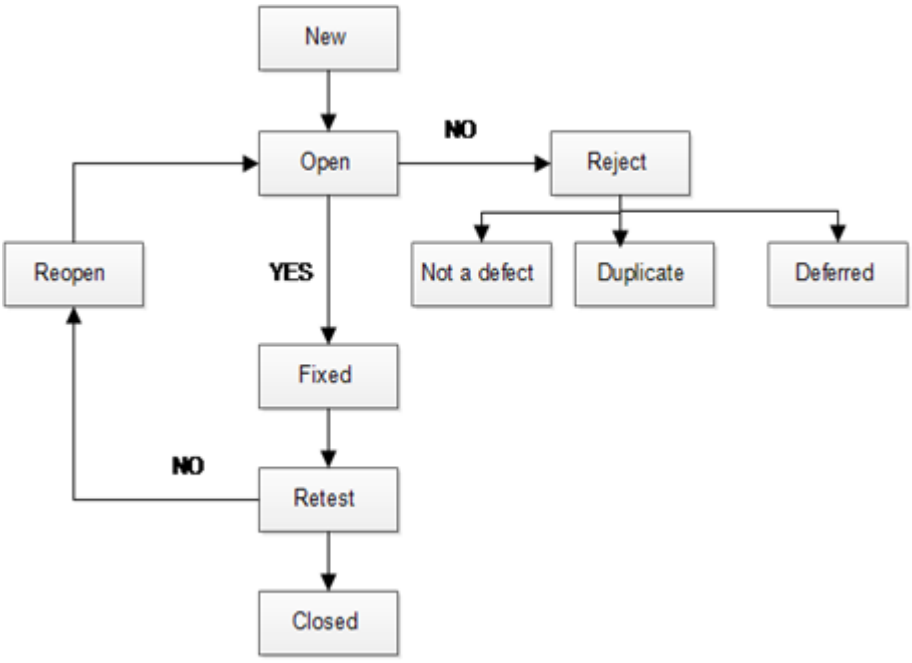
13. Reported By – Name of person who has found the defect.

14. Assign to – Name of the person who is assigned to analyze and fix the defect.

15. Status – Current status of defect it has following options

i) New ii) Open iii) Fixed iv) Rejected v) Reopen Vi) Closed.

Defect Life Cycle –



1. When a defect is detected for the first time status is set to new.

2. Defect is assigned to the development team (Dev Lead) for working on defect, this is assigned by lead of the project to the developer.

3. When the developer starts the process of analysis on the defect and works on fixing it, then the status of defect is set as open.

4. At this stage the developer might accept the defect or reject the defect. If developers reject the defect it may be because -It is not defect, Duplicate, deferred base upon specific reason. The status is set as Rejected

5. If the developer accepts the defect then the developer will perform fixes on build by doing changes in code.

6. After performing a fix, the developer changes the status to Fixed.

7. After fixing the defect, the developer assigns the defect to the tester for retesting the defect.

8. Tester will then perform retesting to verify the defect has been fixed correctly or not.

9. If the defect is not fixed correctly it will be assigned to the developer again and the status of the defect will change to reopened.

10. If the defect is fixed correctly then the status of defect changes to Closed.

Interview questions:

1. What is the defect?

2. Where do you log the defect? - Jira

3. What is the difference between severity and priority? Give an example.

4. What is the defect life cycle?

5. What are the attributes (fields) of defects?

6. Which are mandatory fields and which are optional in Jira?

Introduction to Web Services/API

- What is a web service/API?

- API stands for Application Programming Interface and it is used to communicate between two systems.
- It is simply known as sending the request from one system to another system and getting the required response.
- For Ex. Communication between IRCTC and OTP. Here we send request from IRCTC App to get the response as OTP.

There are some protocols/standards for creating APIs. One of the protocols is SOAP which uses XML for request/response. REST is architecture which uses JSON format. There is lot of flexibility with REST APIs and mostly used these days.

Systematic Flow--- One should use any specific common language which all the parties must understand. These are xml , json

- oXML is tag based language
- oJSON consist of key and value pair enclosed in braces and double quotes.

Secure Flow--- Connection must be secure between both the parties. This can be achieved by some protocols and standards. These are SOAP, REST.

- oSoap is a messaging Protocol used to assign rules for communication
- oREST is an Architecture used for assigning some building blocks for communication

Systematic Flow

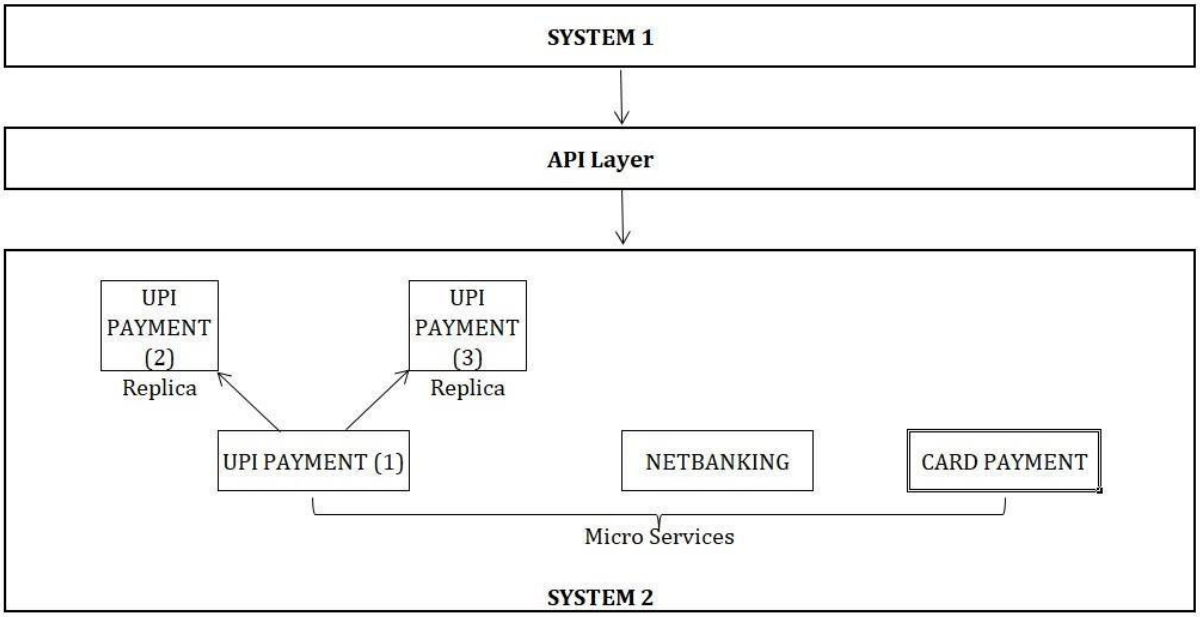
Secure Flow

Advantages of API-

- API provides the security.
  While communicating between two systems, API will provide an API layer in between them which helps to secure out data.

- To avoid the data breaching.
  As we know API provides the API layer for security purpose, So it also avoids Data hacking or breaching.

- To increase the performance and balance the load.



In API, Developer already created a Replica of Microservices which helps to increase the performance of the system by sharing the request with replicas in case load on system 2 increases. Here API will decide how much load to be

shared with each micro service and replica. As this balances the load, the performance will automatically increase.

- API helps in Data hiding.
  API helps to hide the data also.

- API helps for Proper communication between two systems.
  When first system is sending the request to second system, then the request should go for second system only not the third one. For ex. User wants to do a payment for Amazon order by using Gpay app. Then the payment request should go for Gpay only. This is nothing but the proper communication. In between this communication process, API will provide API layer for Security purpose.

- API also checks and authenticates the data which we are passing.
- API tests core functionality.
- API is time effective.
  We can hit a lot of API's at a time with less time.
- Language Independent.
  API is Language independent i. e API reads multiple languages like XML, JSON, HTML, TEXT etc.
- Easy interaction with GUI.

TYPES OF

API-

1) REST API/SERVCES- Uses POSTMAN tool(Representational state transfer)

2) SOAP API/SERVCES - Uses SOAPUI tool(simple object access protocol)

SOAP	REST
SOAP is Protocol.	REST is Architecture.
Uses XML only.	Uses XML, JSON, HTML, TEXT, JavaScript etc
SOAP reads WSDL file.	REST reads API only.
Type of security provided by SOAP is SOAP Enviornment.	Type of security provided by REST is AUTH Token, HEADER, PARAMS etc.
Heavy in weight API.	Light in weight API.
Response time is more.	Response time is less.
Not best for CRUD operations.	Best for CRUD operations.

CONCEPTS UNDER REST:

REST- REST is an architecture used to create rest API *(To create Rest API we need a REST).* REST Assured- To automate rest API we need rest assured libraries. *(For Ex. to automate web browser UI we need External JAR files in Selenium)*

RESTFUL- when we automate rest API successfully it is called as restful services.

Web Services:

Whenever we are calling any API over http (internet) protocol it called as Webservice.

DIFF. BETWEEN REST & WEBSERVICES-

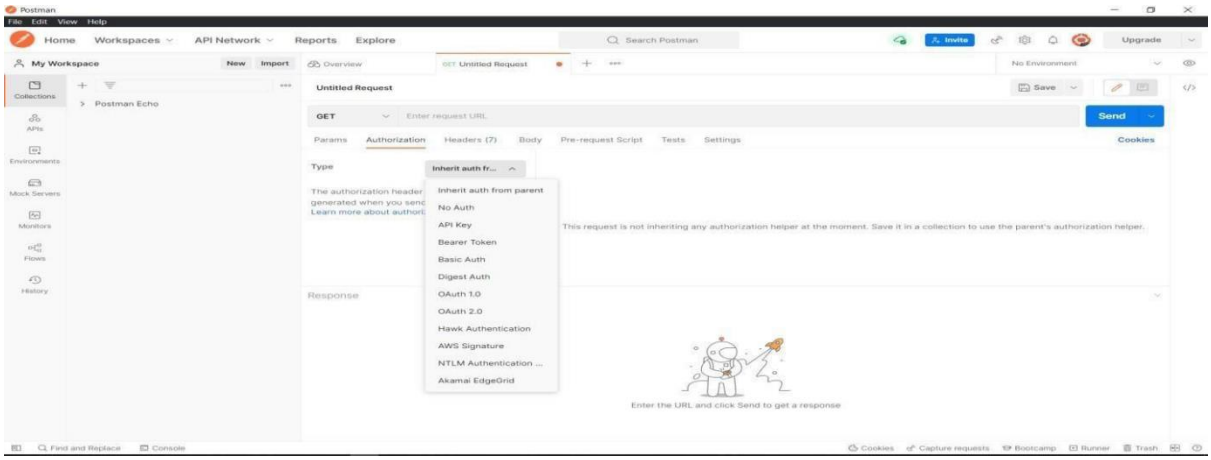
- API call without internet or over the internet and webservices call over the internet.

- The only difference is that a Web service facilitates interaction between two machines over a network. An API acts as an interface between two different applications so that they can communicate with each other. Web service also uses SOAP, REST, and XML-RPC as a means of communication.

DIFF. TYPES OF AUTHORIZATIONS-

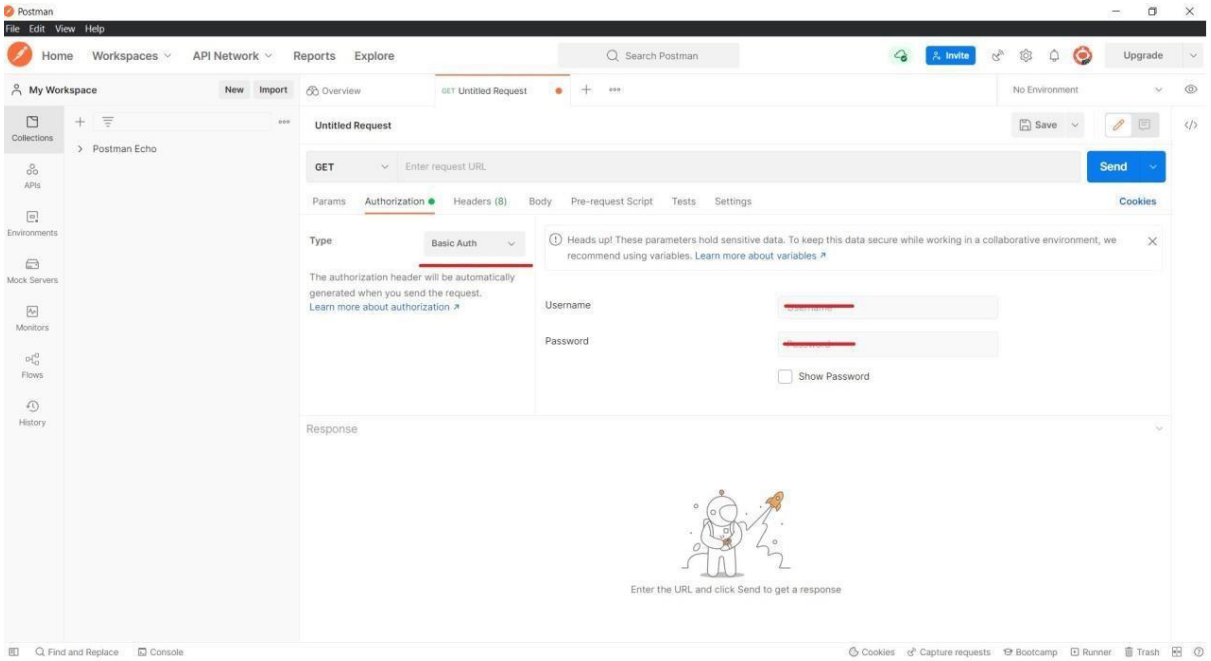
- Basic Auth
- Digest
- Token
- OAuth1

- OAuth2
- No auth
- AWS signature



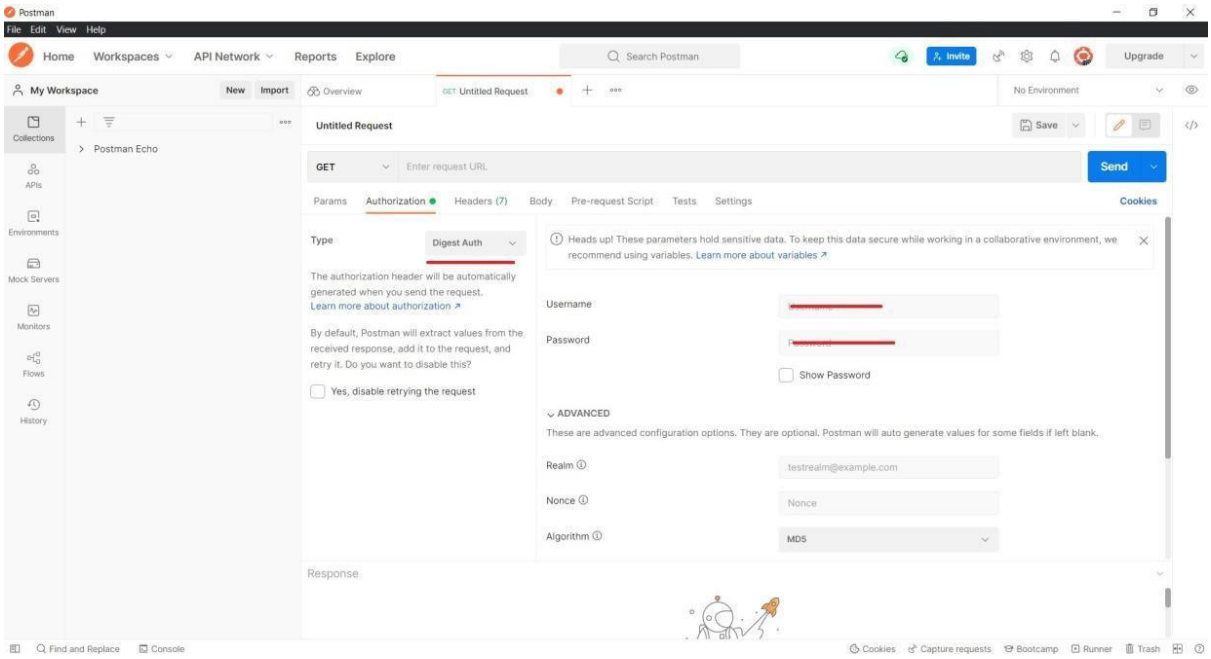
1) Basic Auth-

In Basic we have to pass only Username and Password.



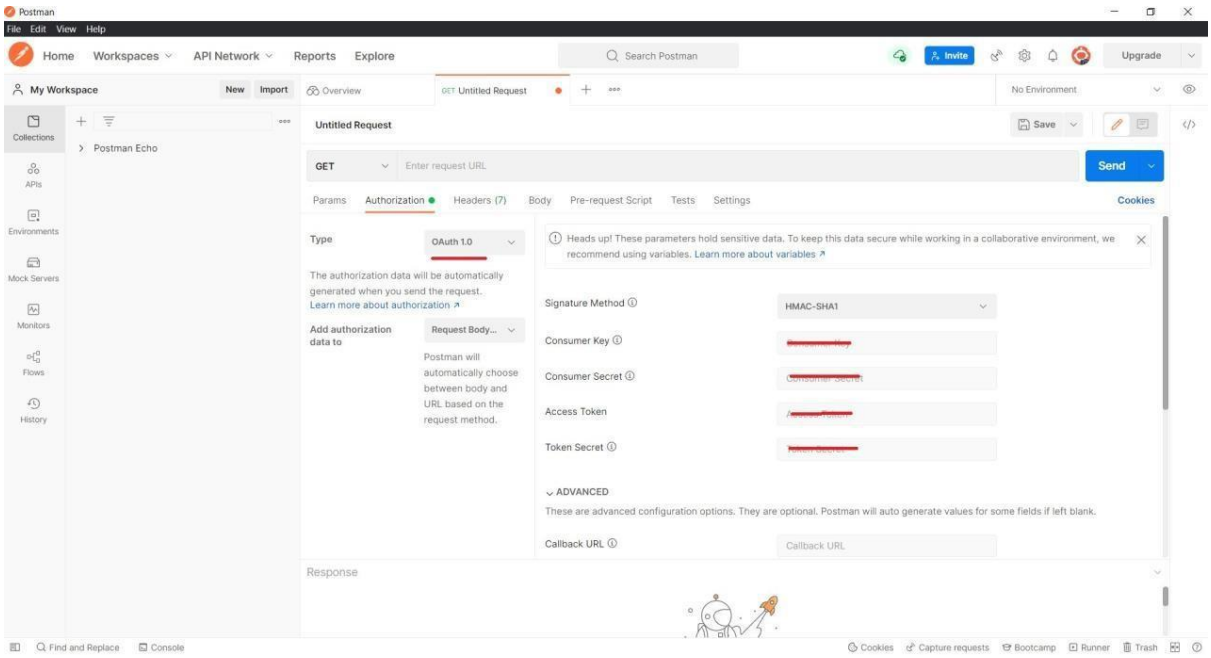
2) Digest Auth

In Digest auth we have to pass Only Username and Password same as that of Basic Auth but Digest Auth is more secure than Basic auth.



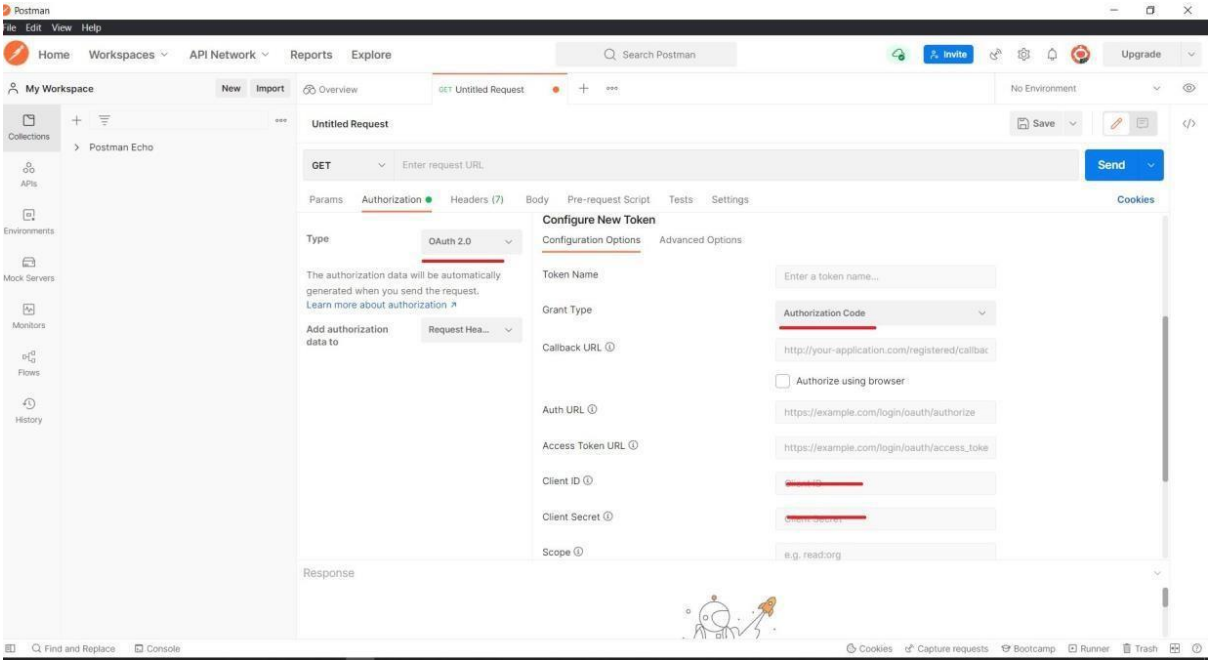
3) OAuth 1.0

In OAuth 1.0 we have to pass Consumer Key, Consumers secrete, Access Token, Token Secret. All this details will be provided by the developer side.



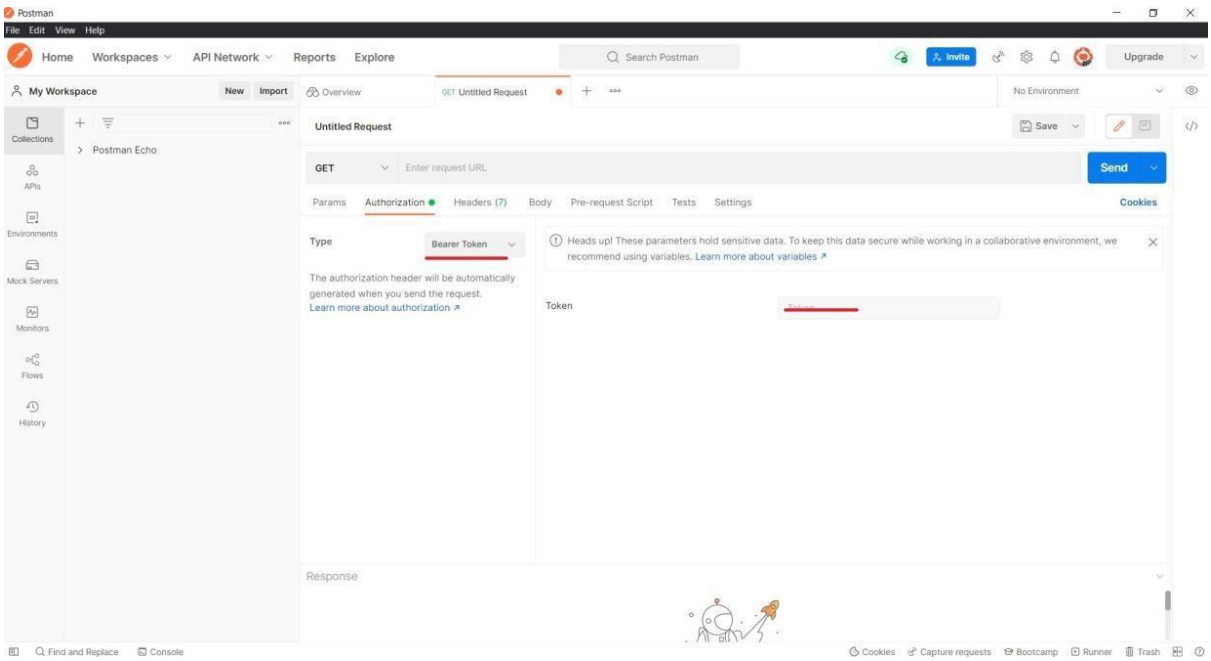
4) OAuth 2.0

In OAuth 2.0 we have to pass Grant type, Client Id, Client Secret. All this details will be provided by developer side.



5. Token-

Here we have to just put the value of token. Token value may be the combination of integer and character values. While entering the token value, we need to mention the Bearer keyword. Bearer means the identification for that token. Token is mostly used type of authorizations.



CRUD OPERATIONS-

C-Create the Data- **POST**

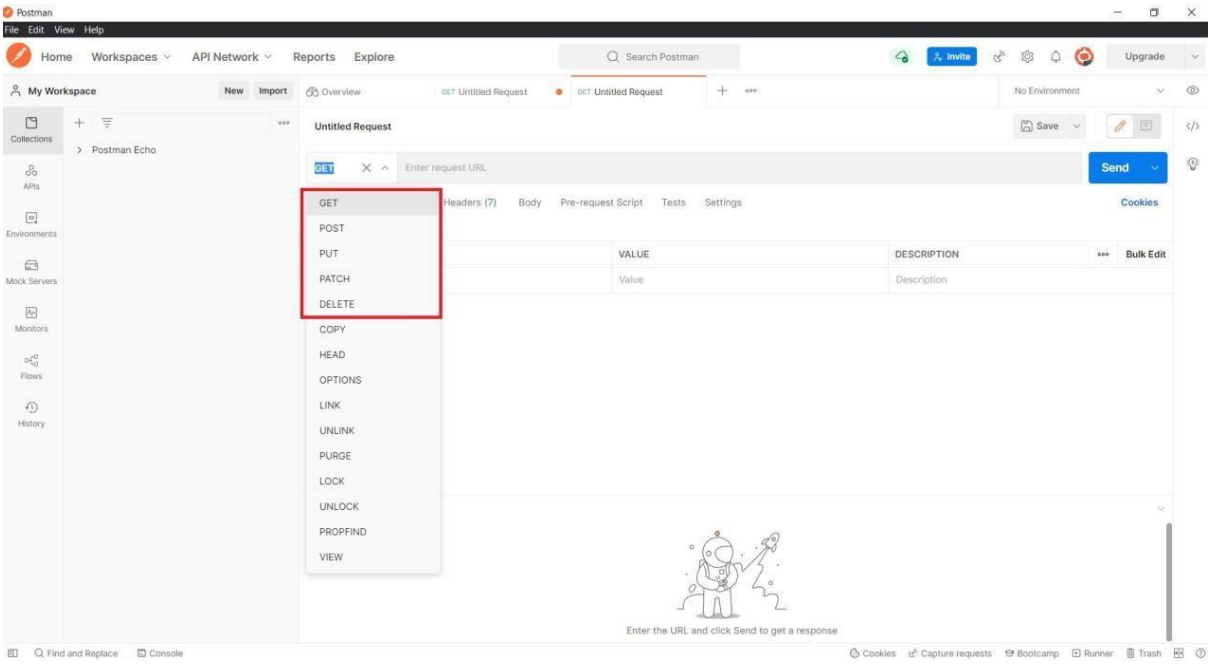
R- Retrieve/ Fetch the Data-**GET**

U- Update the data- **PUT**/PATCH

D- Delete the data- **DELETE**

There are four different methods present in API which are-

1) GET- Used to fetch the data. Read-only method.
2) POST- Used to create the data. Write method
3) PUT- Used to update the data. Write method
4) DELETE- Used to delete the data.Write method



Student App

- Retrieve all students
- Retrieve a single student using ID
- Add new student
- Update existing student information
- Removing existing student

Get all students: http://localhost:8080/student/list
Get specific students: http://localhost:8080/student/list?programme=Financial%20Analysis
Get with limit: http://localhost:8080/student/list?programme=Financial%20Analysis&limit=2
Get by ID: http://localhost:8080/student/90

Add new student: http://localhost:8080/student

New data

```
{
    "firstName": "Piyush",
    "lastName": "Jethwa",
    "email": "piyushjethwa3@gmail.com",
    "programme": "API Testing",
```

```
    "courses": [
      "Rest API",
      "Postman"
    ]
}
```


REST Specific HTTP Status Codes

200 (OK)

It indicates that the REST API successfully carried out whatever action the client requested and that no more specific code in the 2xx series is appropriate.
Unlike the 204 status code, a 200 response should include a response body. The information returned with the response is dependent on the method used in the request, for example:

- GET an entity corresponding to the requested resource is sent in the response;
- HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;
- POST an entity describing or containing the result of the action;
- TRACE an entity containing the request message as received by the end server.

201 (Created)

A REST API responds with the 201 status code whenever a resource is created inside a collection. There may also be times when a new resource is created as a result of some controller action, in which case 201 would also be an appropriate response.
The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a Location header field.
The origin server MUST create the resource before returning the 201 status code. If the action cannot be carried out immediately, the server SHOULD respond with a 202 (Accepted) response instead.

202 (Accepted)

A 202 response is typically used for actions that take a long while to process. It indicates that the request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, or even maybe disallowed when processing occurs.
Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent's connection to the server persist until the process is completed.
The entity returned with this response SHOULD include an indication of the request's current status and either a pointer to a status monitor (job queue location) or some estimate of when the user can expect the request to be fulfilled.

204 (No Content)

The 204 status code is usually sent out in response to a PUT, POST, or DELETE request when the REST API declines to send back any status message or representation in the response message's body.
An API may also send 204 in conjunction with a GET request to indicate that the requested resource exists, but has no state representation to include in the body.

If the client is a user agent, it SHOULD NOT change its document view from that which caused the request to be sent. This response is primarily intended to allow input for actions to take place without causing a change to the user agent's active document view. However, any new or updated metainformation SHOULD be applied to the document currently in the user agent's dynamic view.
The 204 response MUST NOT include a message-body and thus is always terminated by the first empty line after the header fields.
301 (Moved Permanently)
The 301 status code indicates that the REST API's resource model has been significantly redesigned, and a new permanent URI has been assigned to the client's requested resource. The REST API should specify the new URI in the response's Location header, and all future requests should be directed to the given URI.
You will hardly use this response code in your API as you can always use the API versioning for the new API while retaining the old one.
302 (Found)
The HTTP response status code 302 Found is a common way of performing URL redirection. An HTTP response with this status code will additionally provide a URL in the Location header field. The user agent (e.g., a web browser) is invited by a response with this code to make a second. Otherwise identical, request to the new URL specified in the location field.
Many web browsers implemented this code in a manner that violated this standard, changing the request type of the new request to GET, regardless of the type employed in the original request (e.g., POST). RFC 1945 and RFC 2068 specify that the client is not allowed to change the method on the redirected request. The status codes 303 and 307 have been added for servers that wish to make unambiguously clear which kind of reaction is expected of the client.
303 (See Other)
A 303 response indicates that a controller resource has finished its work, but instead of sending a potentially unwanted response body, it sends the client the URI of a response resource. The response can be the URI of the temporary status message, or the URI to some already existing, more permanent, resource.
Generally speaking, the 303 status code allows a REST API to send a reference to a resource without forcing the client to download its state. Instead, the client may send a GET request to the value of the Location header.
The 303 response MUST NOT be cached, but the response to the second (redirected) request might be cacheable.
304 (Not Modified)
This status code is similar to 204 ("No Content") in that the response body must be empty. The critical distinction is that 204 is used when there is nothing to send in the body, whereas 304 is used when the resource has not been modified since the version specified by the request headers If-Modified-Since or If-None-Match.
In such a case, there is no need to retransmit the resource since the client still has a previously-downloaded copy.
Using this saves bandwidth and reprocessing on both the server and client, as only the header data must be sent and received in comparison to the entirety of the page being re-processed by the server, then sent again using more bandwidth of the server and client.
307 (Temporary Redirect)
A 307 response indicates that the REST API is not going to process the client's request. Instead, the client should resubmit the request to the URI specified by the response message's Location header. However, future requests should still use the original URI.

A REST API can use this status code to assign a temporary URI to the client's requested resource. For example, a 307 response can be used to shift a client request over to another host.
The temporary URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s). If the 307 status code is received in response to a request other than GET or HEAD, the user agent MUST NOT automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.
400 (Bad Request)
400 is the generic client-side error status, used when no other 4xx error code is appropriate. Errors can be like malformed request syntax, invalid request message parameters, or deceptive request routing etc.
The client SHOULD NOT repeat the request without modifications.
401 (Unauthorized)
A 401 error response indicates that the client tried to operate on a protected resource without providing the proper authorization. It may have provided the wrong credentials or none at all. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.
The client MAY repeat the request with a suitable Authorization header field. If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the entity that was given in the response, since that entity might include relevant diagnostic information.
403 (Forbidden)
A 403 error response indicates that the client's request is formed correctly, but the REST API refuses to honor it, i.e., the user does not have the necessary permissions for the resource. A 403 response is not a case of insufficient client credentials; that would be 401 ("Unauthorized").
Authentication will not help, and the request SHOULD NOT be repeated. Unlike a 401 Unauthorized response, authenticating will make no difference.
404 (Not Found)
The 404 error status code indicates that the REST API can't map the client's URI to a resource but may be available in the future. Subsequent requests by the client are permissible.
No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.
405 (Method Not Allowed)
The API responds with a 405 error to indicate that the client tried to use an HTTP method that the resource does not allow. For instance, a read-only resource could support only GET and HEAD, while a controller resource might allow GET and POST, but not PUT or DELETE.
A 405 response must include the Allow header, which lists the HTTP methods that the resource supports. For example:
Allow: GET, POST

406 (Not Acceptable)

The 406 error response indicates that the API is not able to generate any of the client's preferred media types, as indicated by the Accept request header. For example, a client request for data formatted as application/xml will receive a 406 response if the API is only willing to format data as application/json.
If the response could be unacceptable, a user agent SHOULD temporarily stop receipt of more data and query the user for a decision on further actions.
412 (Precondition Failed)
The 412 error response indicates that the client specified one or more preconditions in its request headers, effectively telling the REST API to carry out its request only if certain conditions were met. A 412 response indicates that those conditions were not met, so instead of carrying out the request, the API sends this status code.
415 (Unsupported Media Type)
The 415 error response indicates that the API is not able to process the client's supplied media type, as indicated by the Content-Type request header. For example, a client request including data formatted as application/xml will receive a 415 response if the API is only willing to process data formatted as application/json.
For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.
500 (Internal Server Error)
500 is the generic REST API error response. Most web frameworks automatically respond with this response status code whenever they execute some request handler code that raises an exception.
A 500 error is never the client's fault, and therefore, it is reasonable for the client to retry the same request that triggered this response and hope to get a different response.
The API response is the generic error message, given when an unexpected condition was encountered and no more specific message is suitable.
501 (Not Implemented)
The server either does not recognize the request method, or it cannot fulfill the request. Usually, this implies future availability (e.g., a new feature of a web-service API).

ERROR/STATUS codes-
1) 201- Created || when we create data
2) 200- OK || when we get Successful data.
3) 400-Bad Request || when URL wrong or end point missing.
4) 401- Unauthorized || when the session expired, passing invalid token/ username/pass.
5) 403- Forbidden
6) 404-Page not found. || when we are trying to access the URL but URL not present
7) 500- Internal server Error || when any server is down or network issues.
8) 503-Service not available

API Testing Interview Questions

Which type of Authorization is mostly used in organization? Why?

Token Authorization is mostly used in any organization because if it expires then we can easily regenerate the new token by simply clicking on

regenerate the token. Token expiry limit will be decided by the Developer. Token also provides more security.

WEBSERVICES-

Whenever we are hitting any service over the internet it known as webservice. Webservice is any piece of software that makes it available over the internet and uses a standardized XML messaging system.

WSDL-

WSDL stands for Web Service Description Language. WSDL is basically an XML document containing all the details about web services & all API requests.

UDDI-

UDDI stands for Universal Description Discovery integration. UDDI is an XML based standard for describing, publishing and finding web services.

SOAP ELEMENTS-

- Envelop- It is the beginning and end of a message.
- Header- Header elements contain header information.
- Body- Body element contains call & response information.
- Fault- Fault contains error and status information.

WSDL ELEMENTS:

- Type- Define the data types used by the web services.
- Message – Define the data element for each operation.
- Port Type- Describe the operation that can be performed and message involve.
- Binding- Defines the protocol and data format for each port type.

---

1. What are the different methods present in API?

There are different methods present in API: GET POST PUT DELETE PATCH

2. What are the different operations performed in API?

Below are the operations performed in API:

GET- used to fetch the data

POST- Used to create data

PUT- Used to update data (Single and Multiple fields)

DELETE- used to delete data

3. What is the difference between PUT and PATCH?

PUT- we can update all the fields as well as single field

PATCH- we can update single/ partial fields

4. What are the main differences between API and Web Service?

API call internally and web services call over the internet

The only difference is that a Web service facilitates interaction between two machines over network. An API acts as an interface between two different applications so that they can communicate with each other. Web service also uses SOAP, REST, and XML- RPC as a means of communication.

5. What are the advantages of API Testing?

a) API provides the security.
b) API checks the authentication and the data that we are passing.
c) Can transfer the load to diff micro services.
d) API helps to avoid data breaching.
e) Test for Core Functionality.
f) Time Effective- we can hit lots of APIs within less time.
g) Language-Independent- like Json, XML, html, text.
h) Easy Integration with GUI.

6. What is the different Test Environment in the project?

- Generally we will have below four test Environments:

DEV- where developers work.
SIT/QA- where Testers work.
UAT- where Testers and Clients work.
PROD- It's a live environment.

7. What are the test environments of API?

Global- Global has large scope (used to pass variables between diff collections)
Local – Local has small scope (Used to pass variable from one request to another)
-we are using a QA/UAT environment in which we are using a Global and Local environment for API methods.

8. What must be checked when performing API testing?

Error codes, data which are coming (Retrieval data), Time.

9. What tools could be used for API testing?

Postman, Swagger Etc.

10. What are differences between API Testing and UI Testing?

API doesn't provide the GUI (Graphical User interface) but UI provides.

11. What are common API errors that are often found?

These are the common error getting during API testing
400-Bad request
401-Unauthorized
403- Forbidden
404- Page not found
500- Internal server error
503-service not available

12. Any examples of status code?

200- When we get successful data.
201- When we create data into a database.
400- URL wrong or end point missing.
401- When the session expired, passing invalid token/ username/password.
404- When we are trying to access the URL but the URL is not present.
405- Method not allowed.
500- Any server down or network issue.

13. What are the collections?

Collections are used to store the services (API methods)

By using collections we can run all the methods at the same time. We can Import/Export Collection.

14. What is the bearer token?

Bearer token is one of the Authentication pass in headers Bearer means identification for the token.

15. Where do we pass the data in post?

We pass the data in Body-> Raw-> in the form of Json, XML. Html, text

16. Can we run a collection?

Yes, we can run the collection and collection methods at the same time, but before we run the previous or old collection we have to update the authentication.

17. What is meant by endpoints/service URL?

End points are the different service URLs which are used to hit the URL with domain URL.

18. What is meant by API?

API stands for Application programming interface.

- Used to communicate between two systems.
- It is simply known as sending the request and getting the response.

19. What are headers?

Headers is nothing but the what kind of request it is

{content-type= application json/ application xml/application text }

20. What is a bearer?

Bearer is the identifier for a particular token used for the Authentication.

21. Difference between SOAP and REST

SOAP	REST
SOAP is Protocol.	REST is Architecture.
Uses XML only.	Uses XML, JSON, HTML, TEXT, JavaScript etc
SOAP reads WSDL file.	REST reads API only.
Type of security provided by SOAP is SOAP Enviornment.	Type of security provided by REST is AUTH Token, HEADER, PARAMS etc.
Heavy in weight API.	Light in weight API.
Response time is more.	Response time is less.
Not best for CRUD operations.	Best for CRUD operations.

22. Types of API

REST API- Uses Postman tool (Representational state transfer)

SOAP API- Uses SOAPUI tool (simple object access protocol)

23. Concept under REST:

REST- REST is an architecture used to create rest API.

REST Assured- To automate rest API we need rest assured libraries.

RESTFUL- when we automate rest API it is called restful services.

24. What is the difference between ‘/’ and ‘?’

/- Path parameter

?- Query parameter

25. What is producer and consumer?

Producer- who produce the data

Consumer- who consumes the data

26. What is URI?

URI- Unique resource identifier URI= URL+ENDPOINT

Eg. https://www.amazon.com+/login/home

27. What are diff ways to pass the data/ scripting languages?

a) JSON:

{

name": "Suraj ",

"email": "Suraj123@gmail.com",

"gender": "Male",

"status": "Active"

b) XML:

<name>suraj</name>

<email>suraj@gmail.com</email>

c) String

d) Text

e) Html

f) Javascript Etc.

28. What are headers?

Headers mean what kind of data we are passing.

a. Authorization

b. Content Type

c. Language etc.

29. What do we pass in http request?

a. URI

b. Headers

c. Payload

30. What are different authorizations?

a. Basic Auth

pass the username and password.

b. Digest

Whenever we are passing username and pass it will get convert in # keys. It means your username/pass will be secured on the server side too.

c. Oauth1

Oauth1 required below things:

1. Consumer Key
2. Consumer Secret
3. Access Token
4. Secret Token

Above info will get from developers.

e. Oauth2

Oauth2 required below things:

1. Client Id
2. Client Secret
3. Grant type

Above info will get from developers.

f. Bearer Token

g. NoAuth

31. What are Oauth1 and Oauth2?

Oauth1- this auth is used when we need third party logins.

Oauth2- this auth uses when we have single url and different endpoints

32. What is WSDL file

WSDL basically an XML document contains all the details about web service and all API request

33. What is Web service?

Whenever we are hitting any service over the internet it known as webservice.

Webservice is any piece of software that makes it available over the internet and uses a standardized XML messaging system.

34. What is UDDI?

- Universal description discovery integration

-UDDI is an XML based standard for describing, publishing and finding the webservices.

35. What are diff soap elements/components?

a. Envelop – It is beginning and end of message.

b. Header – Header elements contain header information.

c. Body – body element contains call and response information.

d. Fault – Fault contain error and status information.

36. What is diff WSDL element/component?

a. Type- Define the data types used by the webservices.

b. Message – Define the data element for each operation.

c. Port Type- Describe the operation that can be performed and message involve.

d. Binding- Defines the protocol and data format for each port type.

38. What are different API gateways?

a. SSL certificate

b. Routing

c. Adapter

d. Cache

e. Load balancer

39. Difference between monolithic and microservice?

Monolithic - all api available under one service

Microservice- all api have different microservice.

40. What is means URI(API) URI= URL+endpoints(resource)

url: www.facebook.com

endpoints: /login/home

URI- uniform resource identifier url- uniform resource locator

Automation Scenarios to Test Library API -

Verify if API responses returns Success Codes with Proper Assertions

Verify if Response Json Schema is displayed as expected

Create Environment/Global/Collection variables to dynamically switch end points of APIs to test in different stages of QA Life cycle

Pass response ID of Add Book to Get Book and Delete ID request Parameters for full functional Automation Testing

Validate the book ID response calculation Logic

Validate if Get Book API retrieves the correct response with Book Details requested

Validate if Delete Book API successfully deleted Book

Implement Try Catch Error Handling Mechanism for every Automation test as Tear Down Script

If Book Already exists in DB, Implement Smart Strategy to delete the existing Book first before Adding the Book again as Prerequisite Automation Step

Parse the complete Json response and verify if the values are displayed as expected

Generate Unique ISBN/ aisle value for every run to make Book Unique using Automation Script

Import the Book Details from CSV/Json without hard coding for Validating API's

Run the APIs with multiple data sets by iterating the data from the CSV as a Data driven testing

What are Environments and variables in Postman?

An environment is a set of variables you can use in your Postman requests. You can use environments to group related sets of values together and manage access to shared Postman data if you are working as part of a team.

Variables allow you to store and reuse values in your requests and scripts. By storing a value in a variable, you can reference it throughout your collections, environments, and requests—and if you need to update the value, you only have to change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

How to use Environments and Variables in Postman?

Variable scopes
Postman supports the following variable scopes:

- Global
- Collection
- Environment
- Data
- Local

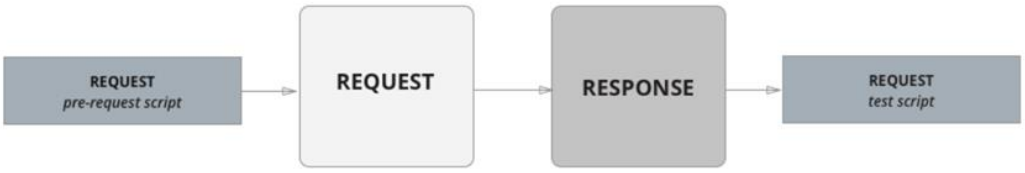
Scripting in Postman

Postman contains a powerful runtime based on Node.js that allows you to add dynamic behavior to requests and collections. This allows you to write test suites, build requests that can contain dynamic parameters, pass data between requests,

Execution order of scripts

In Postman, the script execution order for a single request looks like this:

- A pre-request script associated with a request will execute before the request is sent
- A test script associated with a request will execute after the request is sent



Library API:

Base URI: http://216.10.245.166

1. **Method**: POST

Add Book Complete URL - http://216.10.245.166/Library/Addbook.php

aisle value should be number only. Isbn should be unique to insert. So provide random isbn which makes unique

Input Payload: Json:

{

"name":"Learn Appium Automation with Java",
"isbn":"bcd",
"aisle":"227",
"author":"John foe"

}

1. **Resource** : /Library/GetBook.php?AuthorName=somename **Method** : GET

Output Json:

Output the array of Json object books with all below details

{

Name : “bookname” (String)
Isbn : “A2fdsf” (String)
Aisle : 32 (Integer)

}

1. **Resource**: Library/GetBook.php?ID=3389 - **Method** : GET

Output Json:

```
{
        "book_name": "Selenium automation using Java",
        "isbn": "a23hd738",
        "aisle": "1223"
  }
```

Output Json

```
{
  "Msg": "successfully added",
  "ID": "bcd227"
}
```

1. **Resource** :/Library/DeleteBook.php **Method** : POST

Input Payload: Json:

```
{

"ID" : "a23h345122332"

}
```

Output Response:

```
{

 msg : book is successfully deleted”

}
```

The “PM” object

You will carry out most of the Postman JavaScript API functionality using “pm” which provides access to request and response data, and variables.

API Automation

Topic 1: Importance of Postman variables and environments for Automation testing.

- Create environment
- Create variables

Topic 2: Getting started with scripting in the Postman

Create a test script in Tests to validate the response

```
const jsonData = pm.response.json();
pm.test("validate 200 status code",function()
{
    pm.response.to.have.status(200);
    pm.expect(jsonData).have.property("Msg");
    pm.expect(jsonData).have.property("ID");
    pm.expect(jsonData.Msg).to.eql("successfully added");
    }
```

Dynamically generate values from variables into input payload requests

- Create a global variable “companyCode” with initial value as BK and collection variable as isbn

```
console.log(pm.globals.get("companyCode"));
const code = pm.globals.get("companyCode");
const val = pm.variables.replaceIn('{{$randomInt}}');
pm.collectionVariables.set("isbn", code + val)
```

Testing OAuth 2.0 API’s using Postman and Rest Assured (Automation)

- What is OAuth 2.0? - OAuth 2.0, which stands for “Open Authorization”, is **a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user**. It replaced OAuth 1.0 in 2012 and is now the de facto industry standard for online authorization.
- OAuth 2.0 comes with multiple Grant types - https://oauth.net/2/grant-types/
- Authorization code and client credentials are the most commonly used Grant types for OAuth
- Understand the flow of OAuth (Authorization Code) Grant type with real example
- Backend implementation of Authorization code with different layers of security
- Plan for generating Access Token using APIs in postman for complex flow of Authorization code OAuth 2.0
- API testing with generated access token
- Automate complete OAuth 2.0 flow using Rest Assured
- Shortcut to generate Access token for OAuth in Postman

Questions on OAuth:

How many grant types do you have in OAuth 2.0? Authorization code I have used. There are others like client credentials, PKCE, Refresh Token.

Which Authorization server URL you have used in your testing? Google

What is Scope? What details do we need.