API Testing (Testing funda)

↳ API is a software intermediary that allows two applications to talk to each other. it helps to keep source code private and also provides a layer of abstraction to the user of the API.

↳ Web APIs are APIs that can be accessed using the HTTP protocol. It can be categorized on basis of who all can view and use it as follows:

1. Open/Public APIs
2. Internal APIs or Private APIs
3. Partner APIs
4. Composite APIs

↳ Different use cases call for different implementations and architectures i.e. different accepted data types and commands.

(1) REST (Representational State Transfer) : Architectural Pattern

→ To be REST API, an API must adhese to certain constraints like client-server architecture, statelessness, cacheability, layered system etc.

→ These are primarily used to access & work with data.

→ it requires minimum bandwidth & it is faster

→ it supports multiple formats such as JSON, XML, HTML, YAML & plain text but provides less security

② SOAP (Simple Object Access Protocol): Protocol & message

↳ It is a well established web API protocol, which uses XML as the message format to transfer data.

↳ Its main function is to define the structure of messages and method of communication.

↳ These are driven by functionality rather than data and require more bandwidth.

↳ it supports only XML message format.

③ JSON & XML RPC:

↳ RPC is a remote procedural call protocol. they are the simplest and oldest types of APIs.

↳ RPC was developed for client to execute code on a server.

↳ JSON-RPC need JSON to encode its calls whereas XML-RPC used XML for encoding.

* On API testing, we send a request to API with known data and analyze the response:

1. Data Accuracy
2. HTTP status codes
3. Response time
4. Error code in case API return any error
5. Authorization checks
6. Non-functional testing such as security & performance testing

* Postma

↳ It is
testi

* Creat

1. Cli
2. C
3. C

* Diff

1. G
2. P
3.
4. P
5. D

* Col

eg:

* **Postman** :

↳ It is a collaborative platform for developing and testing APIs.

* **Create API (Rest) request** :

① Click on create a request

② click on New button → Request

③ Click on File → New → Request

↓

Select the method and pass details in body according to the method

↓

click on send button

* **Difference between various methods**:

① **GET** : To fetch the details of any record

② **POST** : To create a record

③ **PUT** : To modify the 'complete record'

④ **PATCH** : To modify the 'record partially'

⑤ Delete, connect, options, trace, head (similar to GET request but without response body)

* **Collection** : used to organize and structure API requests

eg: Project Name

→ Project Module 1

→ Project Module 2

↳ ways to create collection :

① Launchpad → Create a Collection → Provide name of collection → Create

② Click on collections beside History on left pane → Create a collection → provide name → Create

or

New collection → provide name → create

③ Click on New → Collection → provide name → Cr

④ Click on New → Request → provide request na create collection → give name → save to sample collection

* Environments : these are multiple hosts/servers on which application or apis are deployed.

↳ ways to create environment :

① Launchpad → create an environment → Add

② Click on New → Environment → Add environment

③ Click on File → New → Environment → Add

↳ Add baseurls as above and in endpoint, mention :

{{baseUrl}}/api/v1/employees

i.e. {{baseUrl}}/endpoint

* How to run collection:

↳ Click on Runner option beside 'New' → select the folder/collection to run → select env., iteration, data, delay → Run → Run summary → Export Results

↳ Click on Retry if we want to retry after run.

↳ Click on console to see log/info/warning/error after run.

* Variables in postman:

↳ variables are used to store data or value. stored value can be reused in requests and scripts.

↳ stored value can be called throughout environments, collections & requests. this will help to update your collections, environments and request from one place instead of updating in every request.

↳ Postman supports 5 types of variables based on scope:

① Global: they have broader (larger) scope and can be accessed bet^n collections, environments, requests and test scripts. they are available throughout workspace.

② Collection: it is limited to collection and APIs in that collection can be collection variable.

③ Environment: scope is limited to environment as dev, test or QA.

④ Data: these comes from external csv or json set value which can be used in collection runner newman.

⑤ Local: these are limited to API request or a these are temporary variables and will no longer be available after execution.

* Creating tests in postman: Running test on api request to check to status code, value

① Click on Tests

② click on Folder → Edit → Tests → Runner

* Creating Pre-scripts in postman: To generate unique value for body, use dynamic

↳ Pre-scripts: Execute before API requests to set dynamic values

↓

↳ API Requests: Actual API request to be executed

↓

↳ Test Scripts: Execute after API requests to test API

* Chaining API requests :

↳ Chaining of APIs means extracting value from one API and pass that value to other APIs.

↳ It is done to eliminate the manual effort while handling with dynamic values that are referring to other APIs.

↳ it helps in automating API collections.

* How to debug or troubleshoot :

↳ ways to go to console :

① Click on console at bottom of launchpad

② Click on View → Show postman Console

↳ Troubleshooting common issues while working in postman!

① Connectivity issues

② Firewall blocking connections

③ Proxy Configuration issue

④ SSL certificates / Client certificates

⑤ Incorrect URLs / Protocol

⑥ Short timeouts

⑦ invalid responses

* Data Driven testing in Postman: pass data in collection runner

① Prepare studentdata.csv or studentdata.json file!

② studentdata.csv:

```
name, salary, age
Ravi, 15000, 25
Ram, 25000, 30
```

⑥ studentdata.json:

```
[
  {
    "name": "Ravi",
    "salary": "15000",
    "age": "26"
  },
  {
    "name": "Ram",
    "salary": "25000",
    "age": "30"
  }
]
```

② Provide data file in collection runner

③ Pass variables in a request body.

④ Use the console to verify the iterations with test data.

* Data Variable : they come from external csv / JSON to set value which can be used in collection runner and newman.

eg: name, salary & age

* How to run collections from command line :

1. Download node.js with npm
2. install node.js
3. verify node installation

   cmd → node -v
4. verify npm installation

   cmd → npm -v
5. install newman

   cmd → npm install -g newman
6. Export the collection which needs to be run →

   save file → copy path from properties →

   cmd → cd "path"
7. Run the collection

   newman run filename → Enter

* Postman Workspaces : For Collaboration purpose

↳ Workspace helps you to organize your postman work and collaborate with teammates. you can group your projects together, with workspace acting as the single source of related APIs.

↳ Types of workspaces :

① Personal
② Team

* How to visualize postman response :

↳ Pm. visualizer. set() method will apply your visualizer code to data and present it in visualize tab when the request runs.

* Postman interceptor :

↳ It is a chrome extension that acts as a browser companion to postman.

↳ Authentication : Verifying identity of client sending request

↳ Authorization : Verifying that client has permission to carry out the endpoint operation

↳ Headers : Providing additional metadata with request

* HTTP status codes : to convey results of a client's request

① 1XX: → Communicates transfer protocol-level information.
informational

② 2XX: → indicates that client's request was accepted successfully.
Success

③ 3XX: → indicates that client must take some additional action in order to complete request.
Redirection

④ 4XX: → it points the finger at clients.
client Error

⑤ 5XX: → server takes responsibility for error.
Server Error

eg: ① 200 OK → for successful get, patch or delete

② 201 created → for response to POST
202 Accepted → Put request

③ 204 No Content → for delete request

④ 400 Bad request → Misformatted json/ incorrect syntax

⑤ 401 Unauthorized → invalid authentication
402 payment required

⑥ 403 Forbidden → authenticated user doesn't have access to source
429 Too many requests

⑦ 404 Not Found → Non existent resource is requested ie. can't found resource.

⑧ 500 internal server Error → encountered unexpected condition which prevented from fulfilling request.

⑨ 504 gateway timeout → can't get response in time
501 Not implemented → http method not supported by & can't be handled
502 bad gateway → server got invalid response
503 service unavailable → server is not ready to handle the request

↳ Workbench is a practice of documenting how a specific activity must be performed. it is often referred to as phases, steps and tasks. these will be five tasks in every workbench : input, execute, check, output and rework.

↳ URI : Universe Resource Identifier
string of characters that identifies a particular resource by following predefined set of syntax rules
eg : https://

↳ URL : Universe Resource Locator (subset of URI)
subset of URI that identifies a resource & location explains how to access that resource by providing an explicit method like https:// or ftp://.

↳ URN : Uniform Resource Name (Subset of URI)
it includes a name within a given space but no location.