What is an API?

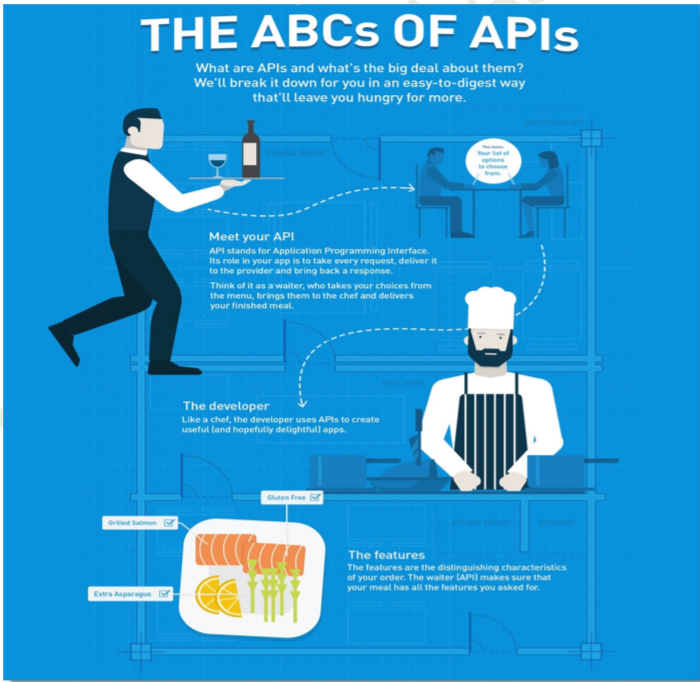
**Definition 1:**

API stands for Application Programming Interface. Talking in technical terms an API is a set of procedures, functions, and other points of access which an application, an operating system, a library, etc., makes available to programmers in order to allow it to interact with other software.

**Definition 2:**

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

**Example 1: Order Food in Restaurant through Waiter**

****

See above picture: let say you went to a restaurant. There is no waiter present, so you need to see the menu lying on the table and then make a request to the kitchen where the chef will prepare the dish for you. But it does not always work that way. **What if the dish is** **not available?** You will have to go to your seat again and decide something else. There will be many customers present in the restaurant which will slow the process of the chef since now he will be listening to the orders instead of preparing them. Also, how can we forget we live in this multilingual world . **What if you** **do not understand the chef's language?** **We need a waiter here.** **A** **waiter is what can be seen as** **an API in the internet world.** The waiter will come and take your requests, give it to the chef and then in response bring back the food. This waiter is bilingual and speaks both of your (chef and you) languages fluently. **What if the dish is not available?** Well, waiter knows beforehand you made the wrong request, so he will tell you then and there on the table that the food item is not available. How much time and energy is saved? This is exactly what an API does.

As we visually depict the above analogy using an image, we can see that you are working as a user in the API world. You make the requests while the waiter works as an API who is an intermediary and takes the request to the appropriate server. This server will be processing your request and responding back to you. As said above, your server or application is the chef who is in the kitchen. He will process your request, cook your desired food and present it back to you as a response. The methods and parameters will be discussed in detail later but here in the analogy, you can think of it as the special requests you make according to your taste and liking. For example, you order something from the menu and describe explicitly that it should be extra spicy. This will help the chef to cook according to your demands.

**Example 2: Hotel Booking Through trivago.com site**

Suppose you are searching for a hotel room through an online travel booking site www.trivago.com. Using the site’s online form, you will fill the necessary information like the city you want to stay in, check-in and check-out dates, number of guests, and number of rooms. Then you click “search.”. But what’s going on between entering your information to receiving your hotel choices? APIs, that’s what! The site aggregates information from many different hotels. When you click “search,” the site then interacts with each hotel’s API, which delivers results for available rooms that meet your criteria. And all this happens in seconds because of an API, which acts like a messenger that runs back and forth between applications, databases, and devices.

**Example 3: Book Flight ticket through Makemytrip.com site**

Let’s take a very common scenario of booking a flight on www.makemytrip.com, which is an online travel service that aggregates information from multiple airlines. When you go for a flight booking, you enter information like journey date/return date, class, etc. and click on search.

This will show you the price of multiple airlines and their availability. In this case, the application interacts with APIs of multiple airlines and thereby gives access to the airline’s data.

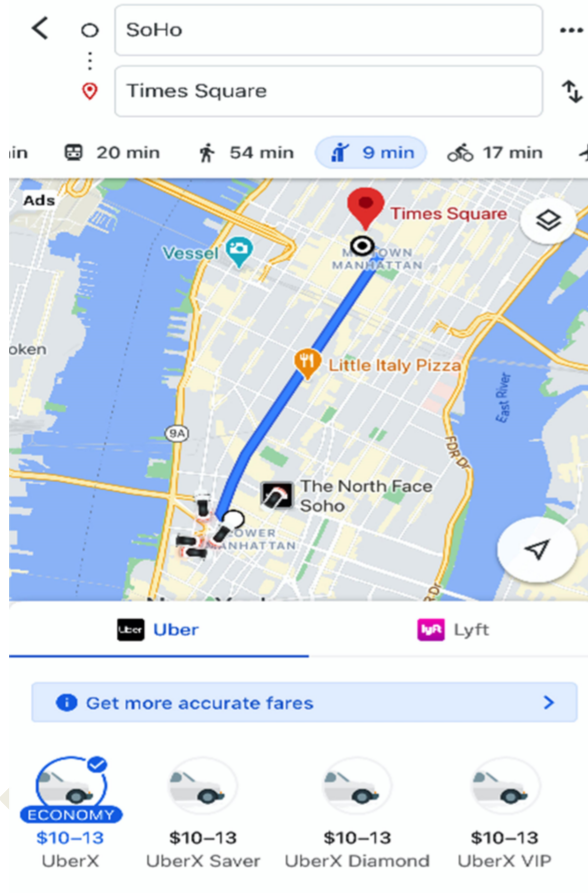
**Example 4: Google Map with OLA/UBER APP**

See below picture: When you search for directions on Google Maps, have you ever noticed that there is an option to book a ride from different ride services?

In the example to the left, Google Maps is using Uber’s or OLA API to request information by sending a HTTP request function with the start and end points of the ride.

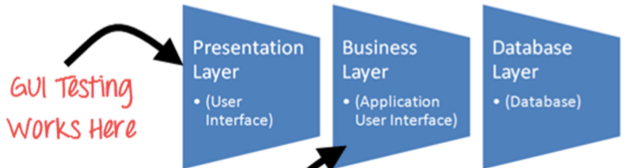
Uber’s API then sends information back such as how many cars are available, where are they currently located, and the costs for the various ride options.

This information is then processed by Google Maps and is available to the user directly linking them the Uber app if it is previously downloaded.



**API Testing**

So now that we have established what an API is and why APIs are critical to modern interconnected, globally distributed applications and services, it is important to understand why API testing is critical.



API testing Work here

API testing is intended to reveal bugs, inconsistencies or deviations from the expected behaviour of an API. Commonly, applications have three separate layers:

 Presentation Layer or user interface

 Business Layer or application user interface for business logic processing

 Database Layer for modelling and manipulating data

API testing is performed at the most critical layer of software architecture, the Business Layer. It is in the business layer, business logic processing is carried out, and all transactions between User Interface (UI) and database happen. So, making sure that, API offers complete intended functionality allows for easy future expansion of the software product.

Why should we do API Testing? API testing is an important activity that testing teams should focus on. It offers a number of advantages over other kinds of testing,

 Language Independent: Data is exchanged via XML and JSON, so any language can be used for automation, independent from the languages used to develop the application.

 GUI Independent: API testing can be done to check the core functionality of the application to expose small errors and to evaluate a build’s strength. Also, we can test the APIs without a user interface.

 Reduce Testing Cost: With API testing we can find minor bugs before the GUI testing. Usually, these minor bugs will become bigger during GUI testing. So finding those bugs while performing API testing will be cost effective.  Enables Faster Releases: The test cases that take about 8-10 hrs When executing UI regression testing take only about 1-2 hrs with API testing. This allows companies to release builds faster with API testing.

GUI testing is too brittle, GUI automated scripts break easily and more time-consuming. But more importantly, when the application is under developed, teams can’t wait for the entire system to be updated and the GUI to be ready before testing occurs.

In the age of agile, testing must take place at a lower level, i.e. at the API level as early as possible. Developers can even do it themselves. API's can be tested even when the GUI of the application is not yet ready. On top of that API tests, because of “API contracts”, can even be created before development is complete.

An API or rather all the API present in the software/application should be tested perfectly. This is the job of a software tester and bears a huge responsibility. A perfect working API leads to the perfect working application. Testing the API solves lot of issues in the application which may arise at some point of time in the future.

 There are many software available for API Testing and one such software is Postman.

API Testing Tools: SOAPUI, Katalon Studio, Postman, Apigee, Karate DSL, JMETER

**Roles & Responsibilities of a Software tester for testing API's**

1. Should able to use all the web methods like GET, POST, DELETE, etc.

2. Should able to use all the web methods like GET, POST, DELETE, etc.

3. Able to validate the XML and Json body by using Json parsers

Become master in a tool of your own choice SOAP UI and Postman are not Automation tools

**What is Postman?**

Postman is rest client software that started as a chrome extension but is now available as native application only. Postman is basically used for API testing in which you can test your APIs with different types of request method types like post, put etc. and parameters, headers and cookies.

 Download Postman and Install on Windows

 How to Download and Install Postman on Windows Machine?  Postman is available for Windows, Linux and Mac as a standalone application. You can install the application in two ways. 1. As a Standalone Application. 2. As a Chrome Extension

1. **Download POSTMAN as a Standalone Application**

A native application means the application will not be supportive of any particular application on an operating system. For example if you install an extension from chrome web store then that application or extension will only run on chrome and no other browser.

A native application is like any software that you install like a game or IDM or any antivirus for instance. A native application takes support of just your operating system and not another application because they are written system specific and not application specific.

1. Go to https://www.getpostman.com/apps OR
2. Direct For Windows 64 bit :https://www.filehorse.com/download-postman-64/
3. Direct For Windows 32 bit: https://www.filehorse.com/download-postman-32/
4. There are Four Options available for download as per your operating system

A) macOs B) Linux c) Windows 64-bit and 32-bit

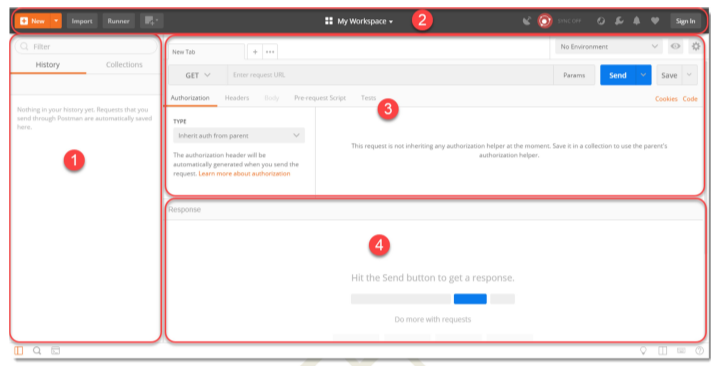
Note: If you are using windows then there is an option for downloading for 32 bit system and 64 bit system. If you are using 64 bit windows then select x64 and if you are using 32 bit system then select x86. Default value is x64.

1. Choose the Operating System on which you want to download Postman and click on "Download" button. Since I have got Windows 64-bit machine, I am going to install x64-Windows
2. Once you download the exe file, you will be needed to install the application. Since I am using Chrome browser, the downloaded exe will appear at the bottom left of the browser.
3. Click on the exe file to install it on the system. First it will install the POSTMAN application
4. There are no further steps for installing. After completion, it will automatically start opening the Postman tool.
5. Once you have the application window up, click on Take me straight to the app. I'll create an account another time as highlighted. Alternatively, you can sign up with Google but it does not matter at present.
6. If you see this page then you have successfully installed Postman on your system

Postman Navigation

Now that we have installed Postman on our system, we will navigate through the UI of Postman in this Chapter.

 Postman navigation can be divided into four UI structures



1. **Side bar Section:**

 History

 Collection

1. **Header Section**

 New

 Import

 Interceptor

 Sync

1. **Builder Section**

 Tabs

 HTTP Methods Types

 URL Bar

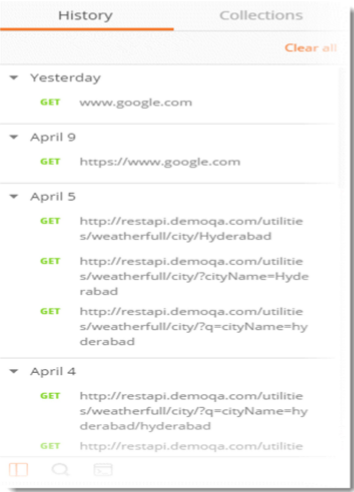
 Header List

4**. Response Section**

It is filled only when to invoke a REST request. This section will be populated with the details of received Response.

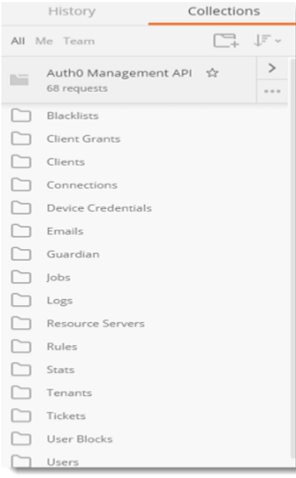
Postman - Left Sidebar Section

1. History Tab



Postman records history of your API request just like any other web browser automatically. As soon as you invoke a REST request, it is saved in the history. It comes handy when you have to search for some particular request that you entered in the past without entering again.

1. Collections



The concept of grouping requests is called Collections and each Collection is displayed under the Collection Tab. A collection in Postman can be imagined similar to a folder in your system. You create a folder, for example movies, and keep movies in it so that you know where all your movies are. Similarly in Postman we save the similar kind of requests under some collection name (that we define) and when we open any collection we get all the Requests under that heading

Postman - Header Section

**New**

Choosing this option will let you choose what "new" you want to start. For example, a collection would open the panel where you can enter a new collection to start and its corresponding requests. Selecting "request" in New will open the request panel where you can enter and save the requests in to the collection of your choice.

New option lets you create the following:

 Request

 Collection

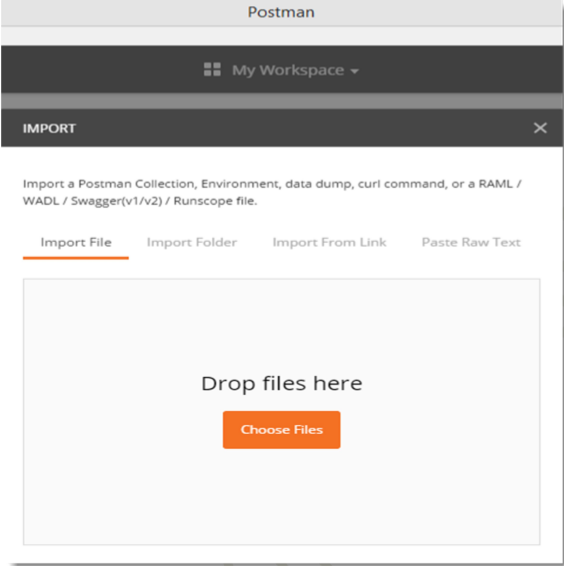
 Environment

 Documentation

 Mock Server

 Monitor

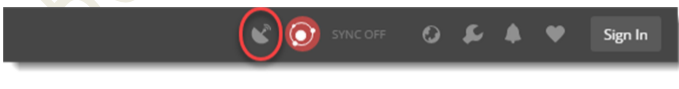
**IMPORT**



Import option lets you import files of different format. Importing means choosing the files located in your system or through a link and running it through Postman. As can be seen from the image it allows you to import a **Postman Collection, Environment, Curl command etc**. Importing a collection is the most common among all.

1. **Interceptor**

Recall we learnt that if you are installing the application from chrome then a separate interceptor is required for the proxy server. This interceptor is inbuilt in the native app. You can set proxy server here to capture the entire API request that you send through your browser. A proxy server can be used to capture all the requests that you send through your browser or from your phone or any other system.



**4.Sync**

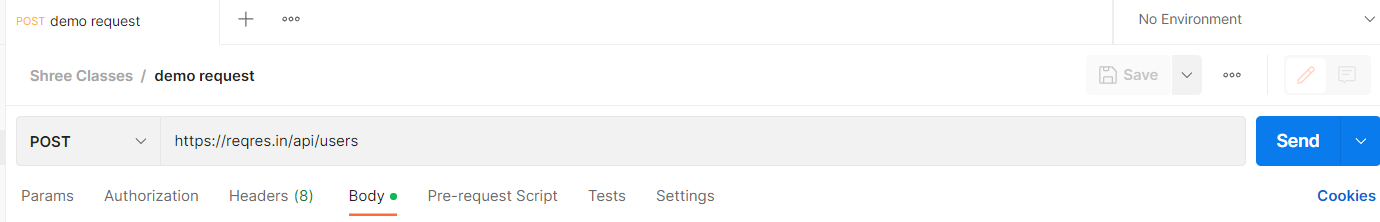
Sync option is for synchronizing the API requests that you have sent on any machine to the Postman cloud. When you are working in Postman and making changes or sending requests, if you Sync is on, it will automatically be saved in your Postman's cloud storage. This way you can have them saved and whenever you sign in on different machine to use Postman, they will automatically appear. This feature requires you to sign in (If you did not during the installation part).



Postman - Builder Section

A builder part of the Postman is basically what a CPU is to a computer. It is the main part that controls all the functionalities and methods to be incorporated inside the API.

A builder part has the following main parts:



Request Type:

 Endpoint Address Bar:

 Params: This option let the user define different Query Parameters for the request.

1. **Request Type**

This is the request type method for the API. It indicates the type of HTTP Request that has been sent. There are different kinds of requests which we will discuss as we proceeds further, but just to know, there are four main types of requests namely GET, POST. PUT and DELETE.

1. **Endpoint Address Bar**

This is the box, beside request type option, to enter the Endpoint (API). It acts just like a browser with the similar interface for New tab. We enter our required endpoint into the bar which is our main URL.

**3. Params**

Params is the parameter option which allows us to write the parameters of the URL. The parameters are embedded into a URL and are very important to get the desired result. They also help us in getting efficient usage of the memory and bandwidth. This will be discussed in a complete chapter later on.

**4. Authorization**

The authorization process verifies whether you have permission to access the data you want from the server. Not all data is available for everyone inside a company, so there lies the solution as Authorization. With the authorization, server first checks whether the data you are asking can be shown to you. If it can be, you get the desired response.

**5. Header**

A header in the HTTP request or response is the additional information that is needed to be conveyed between the client-server. HTTP headers are mainly intended for the communication between the server and client in both directions.

**Postman - Response Section**

A response box is the box which shows the response from the server that we receive after requesting through API. Following things we check in Response



**What is Response?**

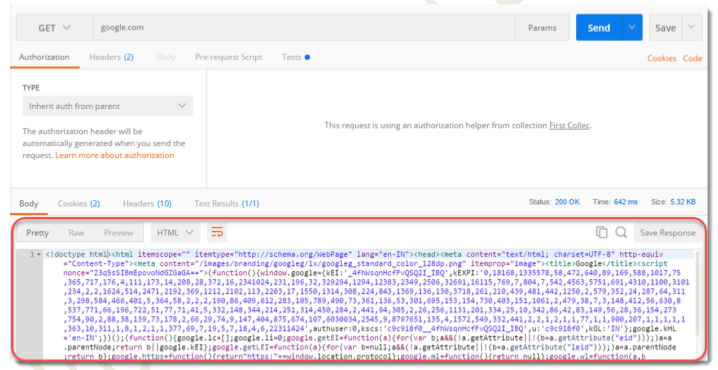
A Response is a message that is received by the server in return to a Request that we send. When we request something, the server acts upon the Request and send back a packet of the requested information. A response depends on the request mainly.

Talking about Response in Postman, the Response user interface contains lots of different things. We will deal with them in detail in this tutorial.

The user interface has the following information blocks

* Response Status and Information
* Response Body
* Response Header

Let's start by getting a response for www.google.com which looks like this



**Response Status and Information**

**Status Code:**

A status code tells you the status of the request. There can be a lot of mistakes in the request and without looking at the status code; we might not always get what went wrong to our request. Sometimes, there can be a typing mistake in the URL or there can be a problem at the server side, status code helps us know about what went wrong (if something went wrong). There are different status codes and each of them

has a different meaning.



Status code 200 OK means that the request was correct and the desired response has been sent to the client. Now, change the URL to http://restapi.demoqa.com/utilities/weatherfull/city/hyderabd. Press Send and see the status code now.



It says 400 BAD REQUEST. It is so because we have changed the name of the city from Hyderabad to Hyderabd. **This means the request was not correct, hence the bad request response.**

**Time**

Time is the duration which the response took after we sent the request and received the response. This is very important sometimes because many projects have Service Level Agreements (SLA) for the time it should take a web service to return a response, this time can be a used to determine the SLA of the web service endpoint



Screenshot (39).png

**Size**

Size is just the response size when it will be saved inside the memory. This response size is the size of complete response and headers and cookies and everything that has been sent along with the response.



**Response Body**

A body depicts the body of the response, which is the main response content, that has been sent from the server.

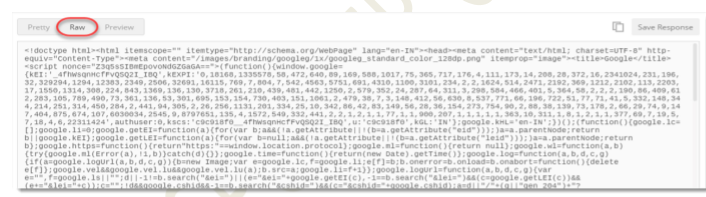
In this case as you can see it is a web page code being sent to us as a response? Now, there lies three ways ahead of us to look at this response:

1. **Pretty**

Pretty is a prettier version of the content being sent. The content is prettier as it is more readable. It has coloured key words and different colours have different meanings. These make a code more readable and look nicer.



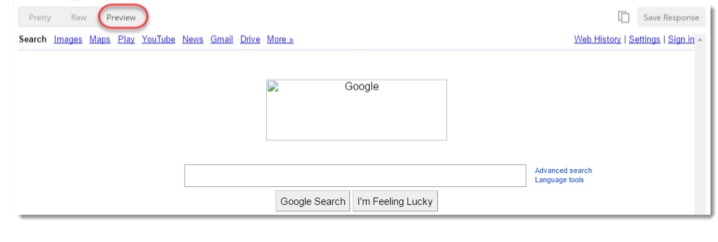
**Raw**



Once you click on Preview you will get just the plain view of the content, as received from the server. It is just a raw version of the code without any colourful keywords.

1. **Preview Preview**

of the code will show you the preview of the page, had the page been run inside a browser. Click on preview and you will see the exact page as you would have seen inside a browser. So this would let you know the response preview without visiting the browser



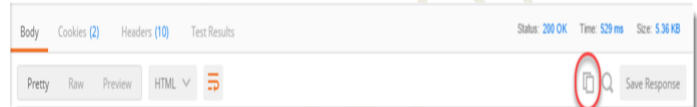
** Format Type**

As discussed above, a request has a defined response to it as defined by the ContentType header. That response can be in any format. For example, in this case we have the response as a HTML code file.

1. JSON
2. HTML
3. XML
4. TEXT
5. AUTO

**Copy Response**

The icon with two rectangles that you see in the corner is used for copying the complete response to the clipboard which is very handy to send the response to your teammates or using afterwards.



**Cookies**

Cookies are the small files which are related to the server files (website pages). Once you visit a website for the first time, a cookie is downloaded on the client's machine. This cookie contains the information which can be used by the same website when you visit again. This helps the website to get you the specific response and specific information based on your last visit. In postman we can clearly see the cookies that have been sent from the server as a response.



 **Response Header**

Headers in a HTTP request or response is the additional information that is transferred to the user or the server. In postman, the headers can be seen in the Headers tab.



Once you click on header you can see different information such as below. Although, every entry in the Headers tab is a header item we will just take a look at the most important ones.

 **Content- Type:** This is the content type of the response. In the above example when we used www.google.com the content type is given as text/html because the response is being sent in the HTML which is one of the options.

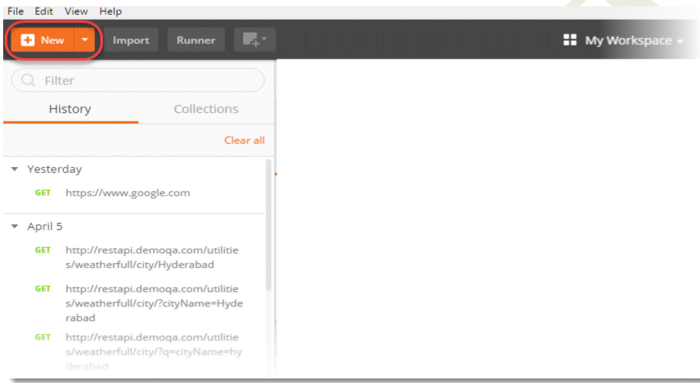
 **Date**: This option shows the date, day and time of the response along with the time zone

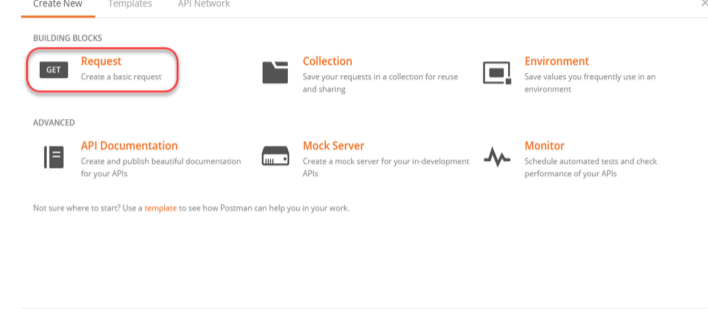
 **Server**: This option tells the name of the server which has responded to the request. In the above example, the server name is shown as gws which corresponds to Google Web Server.

 **Cookie expire time:** As the name suggests, this option tells the expire time of the cookie that has been sent along with the response.

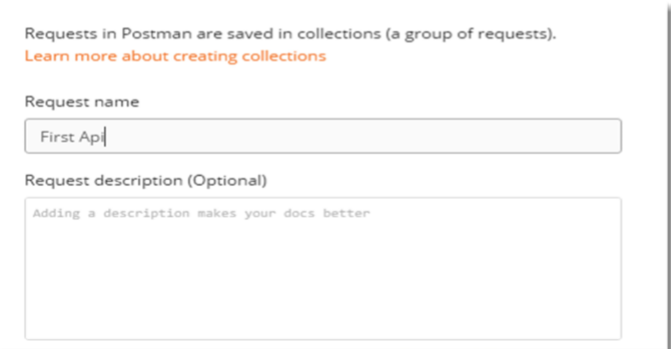
**How to Create New Request in Postman?**

Now, since we have installed Postman and have become familiar with the interface, it's time to start our first steps on Postman for which Postman is actually used for. To start with this tutorial, we will start with the Header part of the Postman and follow the steps.

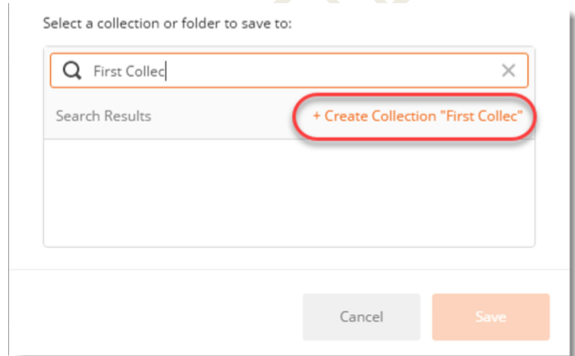
Create New Request in Postman

1. Click on the NEW option in the header part.
2. Click on Request.

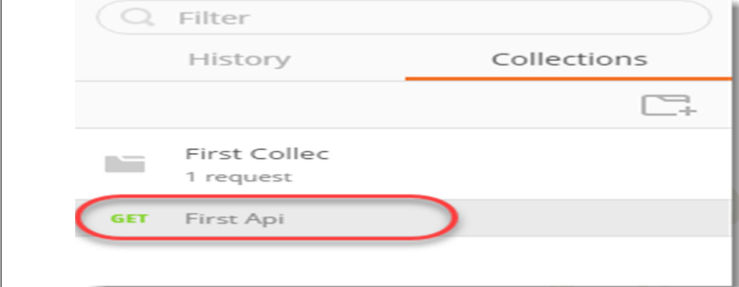
3.Enter a meaningful **Request Name**, like **First Api** we are using. You can also use the description about the API to remember it later about what that API did for other team mates and yourself, but it's optional and we won't be using that in this tutorial.



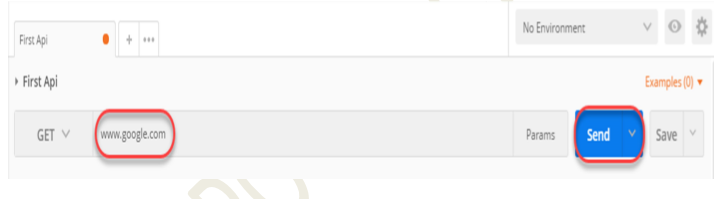
4.Enter a meaningful Collection name in the bottom panel, like First Collec we are using and select **+Create Collection** as shown. **Press Save**

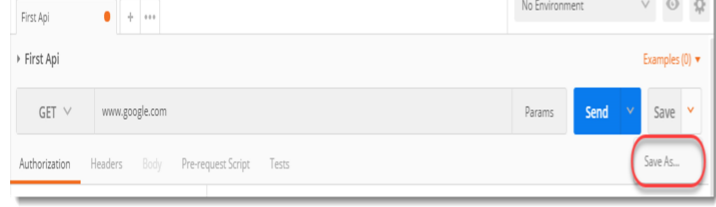


5.Select Collections tab in sidebar, then you will notice all the collections folders, select First Collec and then select First Api under the First Collec tab.



6.Enter www.google.com in the Address Bar and press Send





7. Press Save if you wish to overwrite "First Api" or press the dropdown as shown and Save as a new request.

The **Save As** option opens the same panel which opened through New Request at the start of this tutorial. It gives the option to enter the name and associate the request to some collection.

**HTTP Protocol between Client and Server**

**Example:**

Let us say that we need to get the weather data for my city today. To do this I will need to ask someone who knows about the weather conditions in my city. Assuming that computers are not yet available, we would typically look at the day’s newspaper or may be listen to the radio.

In this process, there are two distinct participants. First one is you, who want the information about the weather. The second one is the Radio or Newspaper who provides the information. If we were to name these two participants we have

1. Consumer who wants to consume specific information. Also, called as Client in ClientServer context.

2. Provider who provides the information. Also, called as Server in Client-Server context.

we would read a newspaper or listen to the radio to get the weather updates. Newspaper and Radio use your local language and you will be able to understand what is written in the paper or spoken on the Radio.

However, for the Clients and Servers on the Web we have to come up with two things

1. A medium for communication, specifically a protocol for two systems to interact. Also called HTTP communication protocol

2. A protocol to ask for the required details from the server. This could be in any form of formatted data. Most commonly used formats are XML and Json.

3. Server responds by sending a Response in any form of formatted data, here also it could be XML or JSON**.**

**Dummy Site for API Testing:**

1. [**https://reqres.in/**](https://reqres.in/)

**2)** [**https://restful-booker.herokuapp.com/apidoc/index.html**](https://restful-booker.herokuapp.com/apidoc/index.html)

**3)** [**https://restfulapi.net/http-methods/**](https://restfulapi.net/http-methods/)

**Response Status Code**

Code Description 1xx Information, i.e., it denotes that the request has been received and under process.

100 Continue: The client can continue with the request as long as it doesn't get rejected.

101 Switching Protocols: The server is switching protocols.

**2xx**

**Success**, i.e., it denotes a successful receipt, processing, and acceptance of the request message.

200 OK: The request is OK.

201 Created: A successfully created new resource

202 Accepted: Request accepted for processing, but in progress

203 Non-Authoritative Information: The information in the entity header is not from an original source but a third-party

204 No Content: Response with status code and header but no response body

205 Reset Content: The form for the transaction should clear for additional input

206 Partial Content: Response with partial data as specified in Range header

**3xx**

**Redirection**, i.e., further action has to be taken for the request to complete

300 Multiple Choices: Response with a list for the user to select and go to a location

301 Moved Permanently: Requested page moved to a new URL

302 Found: Requested page moved to a temporary new URL

303 See Other: One can find the Requested page under a different URL

305 Use Proxy: Requested URL need to access through the proxy mentioned in the Location header 307 Temporary Redirect: Requested page moved to a temporary new URL

**4xx**

**Client Error**, i.e., incorrect syntax or error in fulfilment of the request

400 Bad Request: Server unable to understand the request

401 Unauthorized: Requested content needs authentication credentials

403 Forbidden: Access is forbidden

404 Not Found: Server is unable to find the requested page

405 Method Not Allowed: Method in the request is not allowed

407 Proxy Authentication Required: Need to authenticate with a proxy server

408 Request Timeout: The request took a long time as expected by the server

409 Conflict: Error in completing request due to a conflict

411 Length Required: We require the "Content-Length" for the request to process

415 Unsupported Media Type: Unsupported media-type

**5xx**

**Server Error**, i.e., error invalid request fulfilment at server side

500 Internal Server Error: Request not completed due to server error

501 Not Implemented: Server doesn't support the functionality

502 Bad Gateway: Invalid response from an upstream server to the server. Hence, the request not complete

503 Service Unavailable: The server is temporarily down

504 Gateway Timeout: The gateway has timed out

505 HTTP Version Not Supported: Unsupported HTTP protocol version

**API Testing Interview Questions and Answers:**

**1) What is API Testing and what are the Advantages of API Testing?**

API (Application Programming Interface) is a computing interface which enables communication and data exchange between two separate software systems.

API TESTING is a software testing type that validates Application Programming Interfaces (APIs). The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces.

**Advantages of API Testing**

1. API Testing provide access to the application without a user interface. The core and code level of functionalities of application will be tested and evaluated before GUI Testing. This will help detect miner issues that will be bigger during GUI Testing.
2. API Testing usually less time consuming than GUI Testing
3. API Test Automation require less code so it can be provide better and fast coverage compare to GUI Automation.
4. In API Testing data exchanged using XML or JSON. These transfer modes are completely language –independent allowing users to select any code language when adopting automation testing services for project.

2) **What is Web Service Testing?**

 Any service which is available on the internet or intranet or virtual private network (VPN).

 A service that uses standard XML message format.

 A service that is platform-independent as well as not restricted to a particular operating system or programming language.

 A service that can be discovered by a simple find mechanism over the World Wide Web which is a URL.

 A service capable of exchanging information via HTTP or HTTPS protocol over the World Wide Web.

Web Services Components: The basic web services platform is XML message format and HTTP request and response. All the standard web services work using the following components

 SOAP (Simple Object Access Protocol)

 UDDI (Universal Description, Discovery, and Integration)

 WSDL (Web Services Description Language)

**3.What is the difference between authorization and authentication?**

**Authentication** is a process of presenting your credentials to the system and the system validating your credentials. These credentials tell the system about who you are.

**Authorization** is a process of allowing or denying someone from accessing something, once authentication is done.

**Authorization Vs. Authentication Authorization and Authentication** are two closely related terms. These two terms can also be confusing at first. In this section, we will clear the confusion about these two terms.

**Authentication** is a process of presenting your credentials to the system and the system validating

your credentials. These credentials tell the system about who you are. This enables the system to ensures and confirm a user’s identity. Here system can be anything; it can be a computer, phone, bank or any physical office premises.

Whereas Authorization is a process of allowing or denying someone from accessing something, once Authentication is done. So in layman terms Authentication tells who you are while Authorization tells what you can do.

When a person accesses the server with the key/password, the server checks whether the person is available in directory and is also associated with the same key/password. If it is, you are good to go (Authentication). If you have access to the resource, then you will be granted access to the resource (Authorized).

**4.What must be checked when performing API Testing? OR What Actually you test as a Manual API Tester?**

During the API testing process, a request is raised to the API with the known data. While testing an API, you should consider:

 Accuracy of data & Data type, validations, order and completeness

 Schema validation

 HTTP status codes

 Authorization checks

 Implementation of response timeout

 Non-functional testing like performance and security testing

**5.Please define status code 401. Also, a situation in which we can incur such status code**

Status code 401 is referred for an unauthorized request. An unauthorized request is a request for which you are not authorized. We can incur such a status code when you are not authorized to access the server or you have entered wrong credentials.

Other status codes which are seen commonly are

 200 (OK): Defines that the request was correct.

 201 (Created): The value wrapped with the request has been created in the database. It is needless to say that the request was correct.

 204(No Content): This status code means that the request was correct and received but there is no response to send to the client by the server.

 400 (Bad Request) : A bad request means that the syntax of the request was incorrect. It can happen if you have sent wrong parameters along with the request url or in the body of the request.

 404 (Not Found): A response code 404 means that the server was connected but it could not find what was requested. You can normally see this status code when you request a web page which is not available.  409 (CONFLICT), states conflict situation while executing the method for example, adding duplicate entry.

 500 (INTERNAL SERVER ERROR) , states that server has thrown some exception while executing the method**.**

**6.What are the core components of an HTTP request?**

An HTTP request contains five key elements:

1. An action showing HTTP methods like GET, PUT, POST, DELETE.

2. Uniform Resource Identifier (URI), which is the identifier for the resource on the server.

3. HTTP Version, which indicates HTTP version, for example-HTTP v1.1.

4. Request Header, which carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, format supported by the client, format of a message body format, cache settings, and so on.

5. Request Body, which indicates the message content or resource representation**.**

**7.** **) What are the different HTTP Request methods?**

HTTP request methods specify the action to perform through the request. These are also known as verbs and generally used for CRUD operations, i.e., Create, Read, and Update & Delete. Moreover, HTTP request methods are case-sensitive and should always be uppercase. Subsequently, let us see some commonly used HTTP method

**1. GET** - As the name suggests, the Get method fetches the information from the server. Moreover, it is the most commonly used method which does not have a request body. Every time you open a website, the Get request fires to retrieve the website contents. Additionally, it is equivalent to the read operation. Some of the main features of the GET method are-

 We can easily bookmark the data using the GET method.

 The limit of the length of values is generally 255 characters for the GET method.  GET requests are cacheable.

 The parameters passed in GET methods store in the browser history.

 For any given HTTP GET API, if the resource is found on the server, then it must return HTTP response code 200 (OK) – along with the response body, which is usually either XML or JSON content (due to their platform-independent nature).

 In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

 Note: Just copy below URL and Paste in Postman URL Search Box

 Example request URIs for GET Methods:

 HTTP GET <http://www.appdomain.com/users>

 HTTP GET <http://www.appdomain.com/users?size=20&page=5>

 HTTP GET <http://www.appdomain.com/users/123>

 HTTP GET <http://www.appdomain.com/users/123/address>

 HTTP GET [https://bookstore.toolsqa.com/BookStore/v1/Books HTTP/1](https://bookstore.toolsqa.com/BookStore/v1/Books%20HTTP/1).

**2) POST:** The Post method works to send data to the server. You may add or update data using the Post request. We send the information that needs to update in the request body. In the real world, the form data on website updates using the Post request. Some of the critical features of a POST method are-

 Data passed through the POST method is not visible in the browser URL.

 Additionally, values passed through POST are not stored in browser history.

 Moreover, there is no restriction on the length of data sent through the POST method.

 Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a Location header.

 Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status. Example request URIs

 HTTP POST <http://www.appdomain.com/users>

 HTTP POST <http://www.appdomain.com/users/123/accounts>

**3) PUT:** (Update the data): Overall data will be update

If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.

Example request URIs

 HTTP PUT <http://www.appdomain.com/users/123>

 HTTP PUT <http://www.appdomain.com/users/123/accounts/456>

**4) PATCH**: This method is again similar to Post and Put methods, but we use it when we have to update some data partially. Moreover, unlike the Post and Put methods, you may send only the entity that needs updating in the request body with the Patch method.

**5) DELETE**: Like its name, the Delete method deletes the server's representations of resources through the specific URL. Additionally, just like the Get method, they do not have a request body.

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity

Example request URIs

 HTTP DELETE <http://www.appdomain.com/users/123>

 HTTP DELETE <http://www.appdomain.com/users/123/accounts/456>

**8).** **Enlist some of the API examples which are very well known and popular.**

**Answer:** There are several such examples. Enlisted below are some most popular ones:

 Google Maps API: These are designed mainly for mobile and desktop use with the help of a flash interface and JavaScript.

 Amazon Advertising API: Amazon is known for their products and thus their advertising API accesses their product to discover their functionality and thus advertise accordingly.

 Twitter: The API for twitter is usually in two categories, one for accessing data and the other for interacting with the twitter search.

 YouTube: This API used for YouTube includes various functionalities including videos, live streaming, player, etc.

**9). What is difference between Web Service Testing and API Testing?**

|  |  |
| --- | --- |
| **Web Service Testing** | **API Testing** |
| All web services are APIs and internet connection is must for this | **.** All API are not Web Services and internet connection is not mandatory. Ex. Java Jar file or Selenium Jar file |
| . It is not open source but can be used by any client that understands XML | It is open source and it can be used by any client that understands JSON or XML. |
| It requires aSOAP protocol to receive and send data over the network, so it is not light-weight architecture**.** | . It is light-weight architecture and good for devices which have limited bandwidth, like mobile devices |
| A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication | API may use any style of communication |
| It only supports the HTTP protocol. | It supports the HTTP and HTTPS protocol: URL, Request/Response Headers, caching, versioning, content formats. |

**10). What is HTTP and HTTPS?**

**What is HTTP?**

An HTTP stands for Hypertext Transfer Protocol. The HTTP protocol provides communication between different communication systems. When the user makes an HTTP request on the browser, then the webserver sends the requested data to the user in the form of web pages.

**What is HTTPS?**

The full form of HTTPS is Hypertext Transfer Protocol Secure. The HTTP protocol does not provide the security of the data, while HTTP ensures the security of the data. Therefore, we can say that HTTPS is a secure version of the HTTP protocol. This protocol allows transferring the data in an encrypted form.

|  |  |
| --- | --- |
| **HTTP** | **HTTPS** |
| The full form of HTTP is the Hypertext Transfer Protocol | The full form of HTTPS is Hypertext Transfer Protocol Secure |
| It is written in address bar like http://. | It is written in address bar like https://. |
| It is unsecured as the plain text sent, which can be accessible by hacker | It is secured as the encrypted data sent, which cannot be accessible by hacker |
| It is mainly used for those websites that provide information like blog writing | It is a secure protocol, so it is used for those websites that require transmitting the bank account details or credit card numbers |
| . It does not use SSL | It uses SSL that provides the encryption of the data |
| Google does not give preference to HTTP Website | Google give preference to HTTPS Website |
| The page loading speed is fast | The page loading speed is slow as compared to HTTP because of the additional feature that it supports, i.e., security |

**11.) What is difference between API Testing and UNIT Testing?**

|  |  |
| --- | --- |
| **API Testing** | **Unit Testing** |
| Developers perform it | Testers perform it |
| Separate functionality is tested | End to end functionality is tested |
| A developer can access the source code | Testers cannot access the source code |
| UI testing is also involved | Only API functions are tested 5 |
| Only basic functionalities are tested | All functional issues are tested |
| Limited in scope | Broader in scope |
| Usually ran before check-in | Ran after build is created |

**12.Name some of the common protocols used in API Testing?**

Some of the protocols using in API Testing are as follows:

 HTTP (Hypertext Transfer Protocol)

 REST (Representational State Transfer)

 A RESTful web application exposes information about itself in the form of information about its resources. It also enables the client to take actions on those resources, such as create new resources (i.e. create a new user) or change existing resources (i.e. edit a post).

 It means when a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource.

 For example, when a developer calls Instagram API to fetch a specific user (the resource), the API will return the state of that user, including their name, the number of posts that user posted on Instagram so far, how many followers they have, and more.

The representation of the state can be in a JSON format, and probably for most APIs this is indeed the case. It can also be in XML or HTML format.

What the server does when you, the client, call one of its APIs depends on 2 things that you need to provide to the server:

 An identifier for the resource you are interested in. This is the URL for the resource, also known as the endpoint. In fact, URL stands for Uniform Resource Locator.

 The operation you want the server to perform on that resource, in the form of an HTTP method, or verb. The common HTTP methods are GET, POST, PUT, and DELETE.

** SOAP (Simple Object Access Protocol)**

 It is important for web applications to be able to communicate over the Internet.  The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.  SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

** JMS (Java Message Service)**

The Java Message Service (JMS) API is a messaging standard that allows application components based on the Java Platform Enterprise Edition (Java EE) to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

 **UDDI (Universal Description, Discovery, and Integration)**

Is an XML-based registry for businesses worldwide to list themselves on the Internet. It helps companies to find one another on the Web and call their services. Basically, with UDDI, companies can register/describe their web services on the Internet so that other companies can find their WSDL and generate stubs out of it

**13.What are different types by which we can see response body in Postman? Explain.**

In Postman, a response body can be seen by three different types

 ***Pretty***

*** Raw***

*** Preview***

Although all the three have their own importance and value in Postman, the most commonly used is Pretty as it shows the response code in different format and colours which is easy to read and analyse the response.

**14.)What are the primary security issues of web service?**

Security issues for web services are broadly divided into three sections as described below

– Confidentiality: A single web service can have multiple applications and their service path contains a potential weak link at its nodes. Whenever messages or say XML requests are sent by the client along with the service path to the server, they must be encrypted. Thus, maintaining the confidentiality of the communication is a must. – Authentication: Authentication is basically performed to verify the identity of the users as well as ensuring that the user using the web service has the right to use or not? Authentication is also done to track user’s activity. There are several options that can be considered for this purpose  Application level authentication  HTTP digest and HTTP basic authentication  Client certificates – Network Security: This is a serious issue which requires tools to filter web service traffic.

**15) What Is an HTTP Status Code?**

Answer: Web Services and HTTP APIs use HTTP Status Codes to tell us what happened when the server processed the request.

The simple grouping for HTTP Status Codes is:

• 1xx - Informational

• 2xx - Success e.g. 200 Success

• 3xx - Redirection e.g. 302 Temporary Redirect

• 4xx - Client Error e.g. 400 Bad Request, 404 Not Found

• 5xx - Server Error e.g. 500 Internal Server Error

**16.)** **What is difference between XML and HTML?**

|  |  |
| --- | --- |
| **HTML** | **XML** |
| . HTML is used to display data and focuses on how data looks | . XML is a software and hardware independent tool used to transport and store data. It focuses on what data is |
| .HTML is markup Language itself | XML provides a framework to define markup languages |
| . HTML is not case sensitive. | XML is case sensitive |
| HTML is presentation language | XML is neither a presentation language nor a programming language |
| HTML has its own predefined tags | You can define tags according to your need |
| In HTML it is not necessary to use closing tag. | . XML makes it mandatory to use a closing tag. |
| HTML is static because it is used to display data. | XML is dynamic because it is used to transport data. |