**Web services :** is technology that enables communication between two application (even though they are built using diff language) through http protocal. Internet is always needed.

**HTTP** - Hyper text transfer protocol used to transfer data over the web.

**eg:** Flipkart payment validating your card detail with your bank.

Request from Flipkart: card detail verification (XML or JASON file request ) Response from bank : Status of the card detail (XML or JASON file response)

**web service architecture:**

**Provider** - web-service provider

**Requester** - Client, who fetches service from provider

**Broker** - finds the available service provider

**Two types:**

**Soap :** Simple object access protocol

its a type of communication using only XML file. both request and response will be in XML.

**Rest :** Representational state transfer

its a type of communication using many file type. supported type XML, JASON, String, Document.

Types of calls: get, post, put, delete.

Restfull api : web services implementing Rest Architecture

**Website:** website is collection web pages(contents created usng HTML) where viewers can only view the information's available they will not be able to interact.

eg: <https://www.w3schools.com/>

**Web-Application:** is a program where viewers can able to interact like creating account, modifying their details etc

eg: Facebook, flipkart.

Both were accessed using web browsers.

**API :** Application Programe Interface , is set rules and specifications that software programs can follow to communicate with each other . It doest not always require internet. **API is an interface(based on Abstraction)**

eg: offline : SMS application will get list of contacts through contact api (will be inbuilt in android ) from contact application

eg: online : Flipkart payment validating your card detail with your bank.

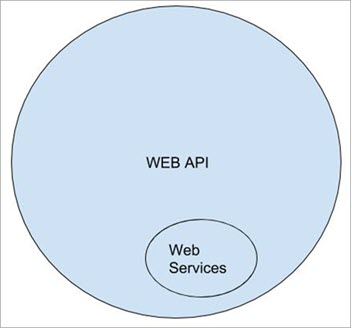
Resquest from Flipkart: card detail verification

Response from bank : Satuts of the card detail

**so, All web services are API**

**but ALL API's are not web services**

so Soap , Rest all comes under API



**API testing -** is a kind of software testing which determines if the developed APIs meet expectations regarding the functionality, reliability, performance, and security of the application.

**API testing - types (what can we test using API)**

**Validation Testing** - validation testing can be seen as an assurance of the correct development.

**Functional Testing** - Checking the API parameters for the particular scenarios

**UI testing** - backend will respond based on Front end actions , those response are through api’s. In UI testing we validate those API’s

**Security Testing** - This practice ensures the API implementation is secure from external threats

**Load Testing -** Checking app's performance at both normal and peak conditions.

**Runtime and Error Detection** - This testing type is related to the actual running of the API and will look for following aspects monitoring, execution errors, resource leaks, or error detection

**Penetration testing**- using this we can find the vulnerabilities of our API’s like how can be bypass the authentication etc.

**Fuzz testing** - a vast amount of random data (referred to as "noise" or "fuzz") will be input into the system to detect any forced crashes or negative behaviors.

**Interoperability Testing** - checks whether the software can interact with other software components and systems

**Some architectural styles for creating a Web API**

**-**REST

-SOAP

-Remote Procedure Call (RPC)

**Advantages of API testing are:**

* **Test for core functionality:** Core functionalities will be tested without the GUI.
* **Time effective:**  API test requires less code so it can provide better and faster ntest coverage compare to GUI test automation.
* **Language Independent:** In API testing data exchange is using XML or JSON. These transfer mode are completely language-independent, which allows users to select any code language
* **Easy Integration with GUI**

**common protocols used in API testing**

-Java Message service(JSM)

-REST

-SOAP

-Universal Description, Discovery and Integration (UDDI)

**Tools for API testing**

-Postman

-Apigee

-SoupUi Pro

- Advance Rest Client

**bugs that can be found using API testing**

- Duplicate functionality

- Stress

-Reliability

-Performance

-Inconsistent error handling

-Multi-Threading error

-Security

-Unused flags

**API documentation templates that are commonly used**

- Swagger(allows you to describe the structure of your APIs so that machines can read them)

- Miredot

-Slat

-API blueprint

-FlatDoc

-RestDoc

WHY: In API testing, we send a request to API with the known data and then analysis the response.

1. We will verify the accuracy of the data.
2. Will see the HTTP status code.
3. We will see the response time.
4. Error codes in case API returns any errors.
5. Authorization would be check.
6. Non-Functional testing such as performance testing, security testing.

**Diff between SOAP and REST**

* SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.
* SOAP is a protocol whereas REST is an architectural pattern.
* SOAP uses service interfaces to expose its functionality to client applications while REST uses Uniform Service locators to access to the components on the hardware device.
* SOAP needs more bandwidth for its usage whereas REST doesn’t need much bandwidth.
* SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.
* SOAP cannot make use of REST whereas REST can make use of SOAP.

## **When to use REST?**

* **Limited resources and bandwidth** – Since SOAP messages are heavier in content and consume a far greater bandwidth
* **Statelessness** – If there is no need to maintain a state of information from one request to another then REST should be used. If you need a proper information flow wherein some information from one request needs to flow into another then SOAP is more suited for that purpose. Eg. Purchase- cart - payment
* **Caching**– If there is a need to cache a lot of requests then REST is the perfect solution.By implementing a cache, the most frequent queries results can be stored in an intermediate location- thus minimizing the amount of trips which are made to the web server.
* **Ease of coding**

## **When to use SOAP?**

1. **Asynchronous processing and subsequent invocation** – if there is a requirement that the client needs a guaranteed level of reliability and security
2. **A Formal means of communication** – if both the client and server have an agreement on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction. Eg.online shopping
3. **Stateful operations –**ifthe application has a requirement that state needs to be maintained from one request to another

**Restfull API’s -** API made using REST architecture

**HTTP methods/ CRUD operation (HTTP protocol supported by REST?)**

Post: To create an entity - Create

Get : To read an entity - Read

Put : To update an entity - Update

Patch : To Partially update an entity

Delete : To delete an entity - Delete

This methods available only for REST. (for Soup mostly POST will be used , Get also available but it is too complex)

**An HTTP request have five components. These are:**

1. **Verb - Action showing HTTP method** like GET, PUT, POST, DELETE.
2. **Uniform Resource Identifier (URI):** Uniform Resource Identifier (URI) to identify the resource on server (http://www.w3.org/java)

[**URL :** is a subset of URI , in addition to identify the resource it also specify the mechanism to retrieve it]

**<protocol>://<service-name>/<Resource Type>/<ResourceID>**

1. **HTTP version:** Indicate the HTTP version like- HTTP V1.1.
2. **Request Header:** Request Header carries metadata for the HTTP request message. **Metadata** could be a client credentials, format supported by the client, format of a message body(xml or jason), cache setting etc.
3. **Request Body:** message content or resource representation.

Rest is always based on secured comunication bewteen aplications.

**Authentication**: Refers to proving correct identity (what you are)

**Authorization**: Refers to allowing a certain action (what you can do}

**different Authentication**

- HTTP authentication:

- Basic : combination of username and password

Authorization: Basic bG9sOnNlY3VyZQ==

- Bearer : static tokens

Authorization: Bearer <token>

- Api key : key will be provide during first time interaction .. after that the same key will be used for access

Authorization: Apikey 1234567890abcdef

- OAuth : open authorization .. uses tokens for authorization... the tokens will be given after sccuessful authentication

Authentication flow/Grant type:

-Authorization code :server-side and mobile web applications (gmail sign in )

-Implicit : client to retrieve an access token directly

-Client Credentials :server-to-server authentication

**Header** : always contains meta data (like client credentials)

**Parameter** : is a actual data (visible to end user)

**Response code :**

**1XX :Informational response** : An informational response indicates that the request was received and understood.It alerts the client to wait for a final response.

100 continuue 102 processing

**2XX : Success** : action requested by the client was received, understood, and accepted

**3XX : Redirection** : client must take additional action to complete the request (URL may be changed )

**4XX : Client error :** error seems to have been caused by the client

400 bad request

**5XX : Server error**

500 Internal Server Error 502 Bad Gateway 503 Service Unavailable

**Tools/technolodies for REST api**

**Manual :** Postman | Advance Rest Client

**Automation :** Java HttpURLConnection class | HTTP client | Rest assured client | Jersey client

**In a GET request, the parameters are sent as part of the URL.**

**In a POST request, the parameters are sent as a body of the request, after the headers.**

**Rest API most of the time will use JSON ( JavaScript Object Notation) for specific servers only it will use XML.**

**Serialization -** converting POJO(plain old java object) into JSON format object.

**De-Serialization -** converting JSON response into POJO

**API chaining-** is a monad allowing a series of API calls to be passed and processed using a one request/response. Monad means a single unit.

**Monolithic Architecture** - Traditional way of building the application, in which application build as a single and indivisible unit. Uses single database.

**Micro-services Architecture** - in this application build as suite of small services and each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. Different database for each services.

**Payload** - Information that we get in response or we send in the body of the request is called Payload

Three formats- Request payload , Ok response payload, Failed response payload.

**caching mechanism -** Caching is a process of storing server response at the client end. It makes the server save significant time from serving the same resource again and again.

**Jason Object -** contains named values (Key-value pair)

{ name:vino, gender:Male}

**Jason Array** - ordered list of values

[ "admin", "editor", "contributor" ]

**Types of Parameters:**

**Path parameters:** Parameters within the path of the endpoint, before the query string (?). These are usually set off within curly braces.

/service/myresource/user/{user}/bicycles/{bicycleId}

**Query parameters:** Parameters in the query string of the endpoint, after the ? . it will be in key-value pair and each parameter separated by &.

/surfreport/{beachId}?days=3&units=metric&time=1400