**REST API**(Representational State Transfer)

Rest API is an application program interface that is backed by the architectural style of REST. It refers to tools, service or software that is based on the REST architectural principle. Although REST can be used on nearly any protocol, they take advantage of HTTP when used for web APIs. The primary advantage of REST APIs is that they offer more flexibility. In REST APIs, data is not constrained to resources or methods. Therefore, it can make multiple types of calls, return various data formats, and even change structurally with the appropriate implement of hypermedia.

**Example)**

1. Instagram API permits your applications to retrieve user tags, photos, account and much more.
2. Twitter also provides a REST API which a developer can query to source the latest tweets, or provide a search query that will return the results in JSON format.
3. GitHub also offers super REST API that you can utilize to perform actions such as following GitHub issues, tracking user activity, and create repositories from your app.

**GET**

* Get parameters will be stored on browser’s session history
* Get responses can be cached
* Get is a safe method (safe methods are HTTP methods that do not modify resources)
* Static websites(application which does not need any other tool to process its files as they will return browser readable content) only responds to GET requests.
* It is used to read a representation of a resource. In a non-error path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST)
* According to the design of the HTTP specification, GET requests are used only to read data and not change it. Therefore, when used this way, they are considered safe. That is, they can be called without risk of data modification or corruption- calling it once has the same effect as calling it 10 times, or none at all. Additionally, GET is idempotent, meaning that making multiple identical requests ends up having the same result as a single request.
* Ex) GET http://www.example.com/customers/12345

**POST**

* It is used to create subordinate resources. That is, subordinate to some other (e.g parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc
* Ex. POST <http://www.example.com/customers/12345/orders>