**Client-Server:**

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

**HTTP:**

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests. Though often based on a TCP/IP layer, it can be used on any reliable transport layer; that is, a protocol that doesn't lose messages silently, such as UDP.

**HTTP Methods:**

GET: The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

**Example:** getText, getTitle, getAlertText, getCurrentURL, getWindowHandle, getSessions

POST: The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

**Example:** navigate to, set Cookie, find element

DELETE: The DELETE method deletes the specified resource.

**Example:** delete cookie, close browser

**HTTP status response Codes:**

200: OK [The request has succeeded]

400: Bad Request [server could not understand the request due to invalid syntax]

404: Not Found [The server cannot find requested resource]

502 Bad Gateway

501 Not Implemented

500 Internal Server Error.

**HTTP Requests and Responses:**

GET <https://www.ibm.com/support/knowledgecenter/SSSHTQ_7.4.0/com.ibm.netcool_OMNIbus.doc_7.4.0/omnibus/wip/api/reference/omn_api_http_eg_gettablecollectionresponse.html>

POST <https://www.ibm.com/support/knowledgecenter/SSSHTQ_7.4.0/com.ibm.netcool_OMNIbus.doc_7.4.0/omnibus/wip/api/reference/omn_api_http_eg_posttablecollectionresponse.html>

**Selenium WebDriver:**

* Open source application
* Automates different browsers actions using client libraries.
* Actions like click, open, enter text, keyboard actions, mouse actions, screenshots
* Works on Client Server Architecture.
* Uses JSON Wire Protocol [uses RESTful Web Services] over HTTP.
* WebDriver uses JSON wire protocol to communicate between client libraries [Java, python, C#, PHP, others] and servers [browser specific drivers like chromedriver.exe, geckodriver.exe, internetexplorerdriver.exe]
* Triggers request and response.

**JsonWireProtocol:**

JavaScript Object Notation (JSON) is used to represent objects with complex data structures. It is used primarily to transfer data between a server and a client on the web. JSON is an industry standard for various REST web services, playing a strong alternative to XML.

WebDriver that communicate with the browser, or a RemoteWebDriver server shall use a common wire protocol. This wire protocol defines a RESTful web service using JSON over HTTP.

The protocol will assume that the WebDriver API has been "flattened", but there is an expectation that client implementations will take a more Object-Oriented approach, as demonstrated in the existing Java API. The wire protocol is implemented in request/response pairs of "commands" and "responses".

* Use any programming language
* Run tests on cloud, sauce labs or selenium grid
* Different browser drivers are used to communicate with the client libraries and real browsers

**WebDriver Architecture:**

Ruby

C#

Java

PHP

Python

JSON Wire Protocol over HTTP

Browser Drivers

Chrome, IE, FF, Safari, Opera, etc.

**HTTP over**

**HTTP Server**

**HTTP over**

**HTTP Server**

***Client***

The machine on which the WebDriver API is being used.

***Server***

The machine running the RemoteWebDriver. This term may also refer to a specific browser that implements the wire protocol directly, such as the FirefoxDriver or GeckoDriver.

***Session***

The server should maintain one browser per session. Commands sent to a session will be directed to the corresponding browser.

***WebElement***

An object in the WebDriver API that represents a DOM element on the page.

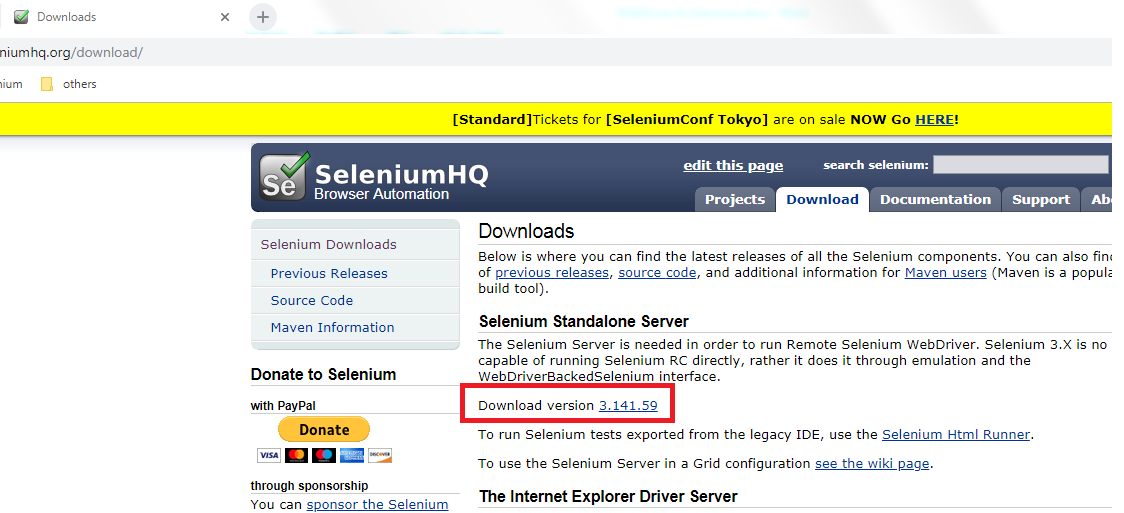
***WebElement JSON Object***

The JSON representation of a WebElement for transmission over the wire. This object will have the following properties:

| **Key** | **Type** | **Description** |
| --- | --- | --- |
| ELEMENT | string | The opaque ID assigned to the element by the server. This ID should be used in all subsequent commands issued against the element. |

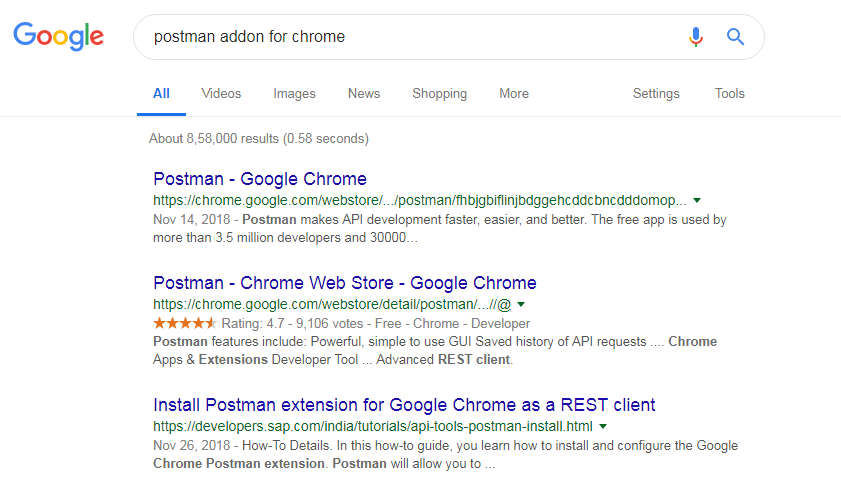
**Demo:**

<https://www.seleniumhq.org/download/>

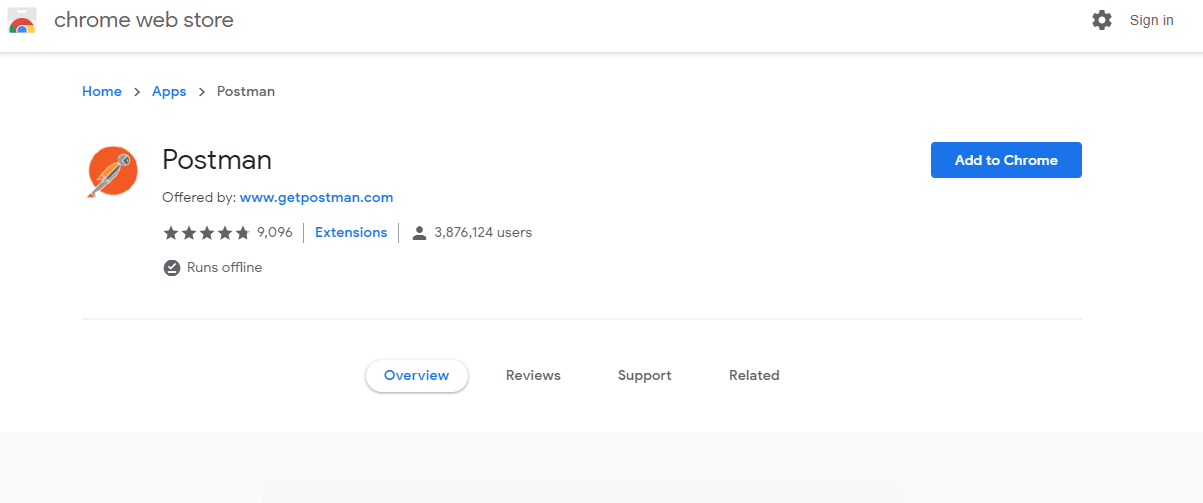


Also download the different browser drivers. [Like internetexplorerdriver.exe, chromdriver.exe, geckodriver.exe]

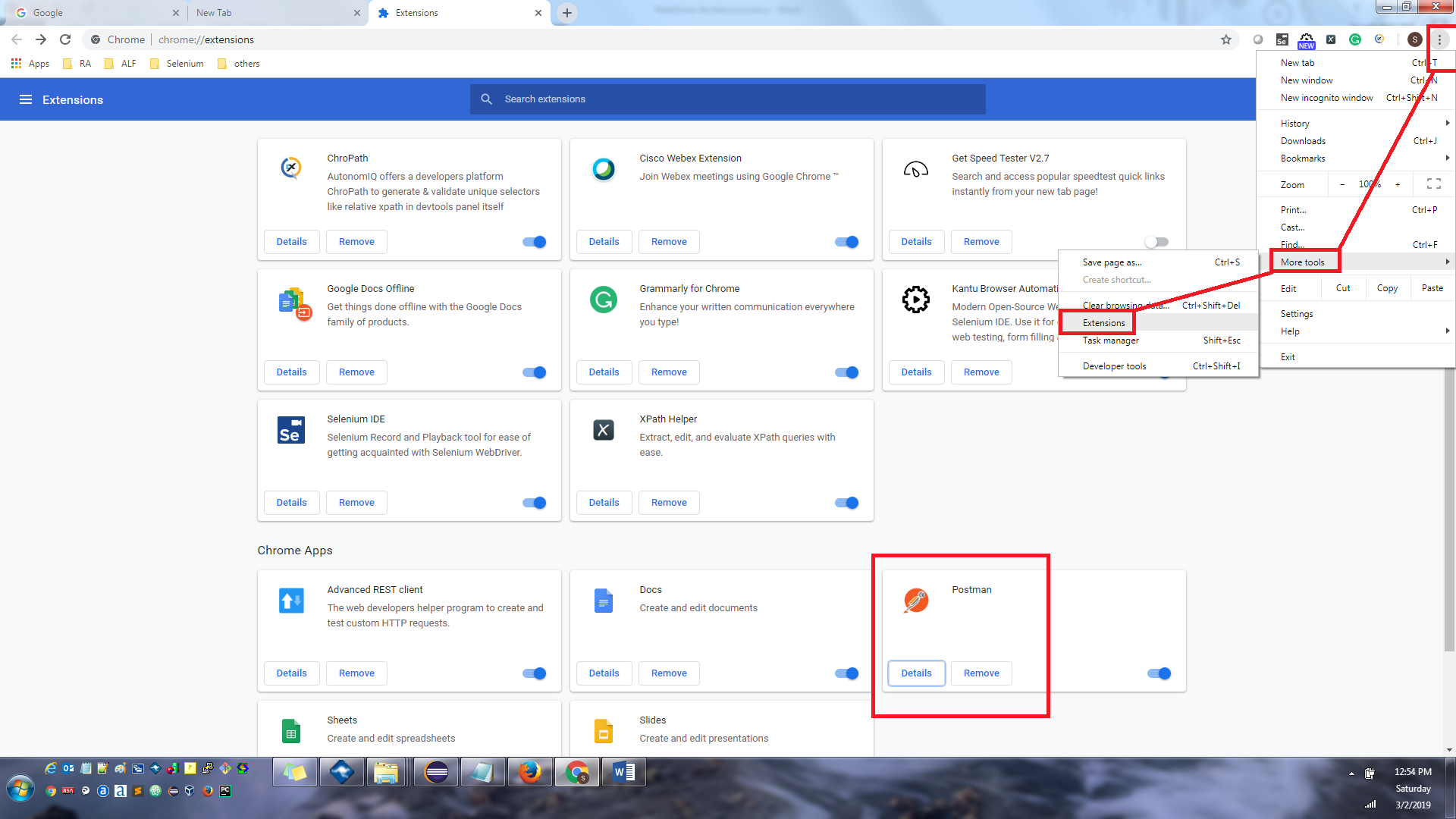
1. Copy the selenium server jar and all browser driver into same folder.
2. Postman add-on to chrome browser (please follow below steps)
   1. Google search for POSTMAN add-on for chrome browser and install



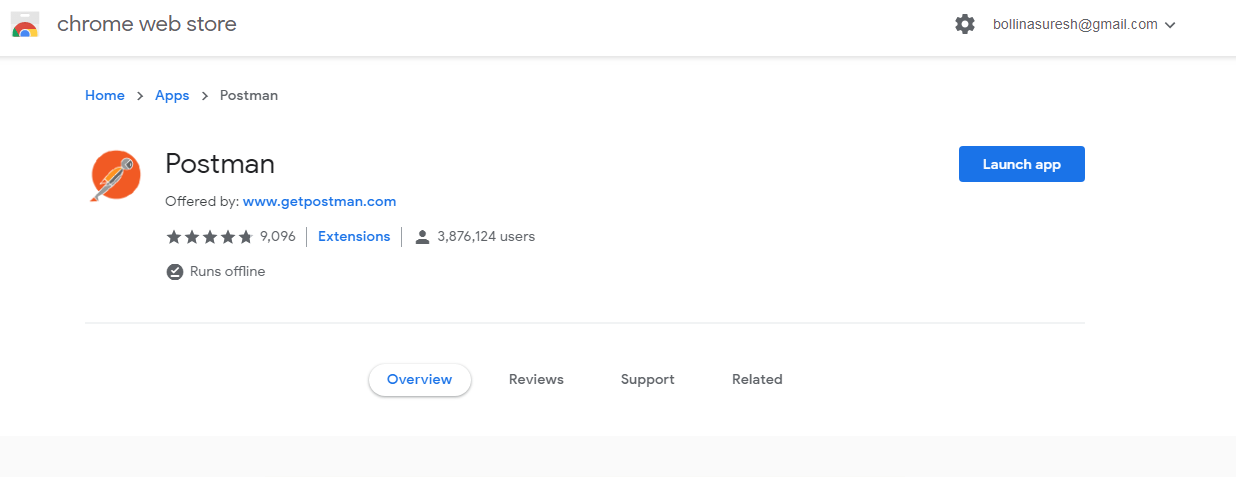
* 1. Click on first link and then Add to Chrome



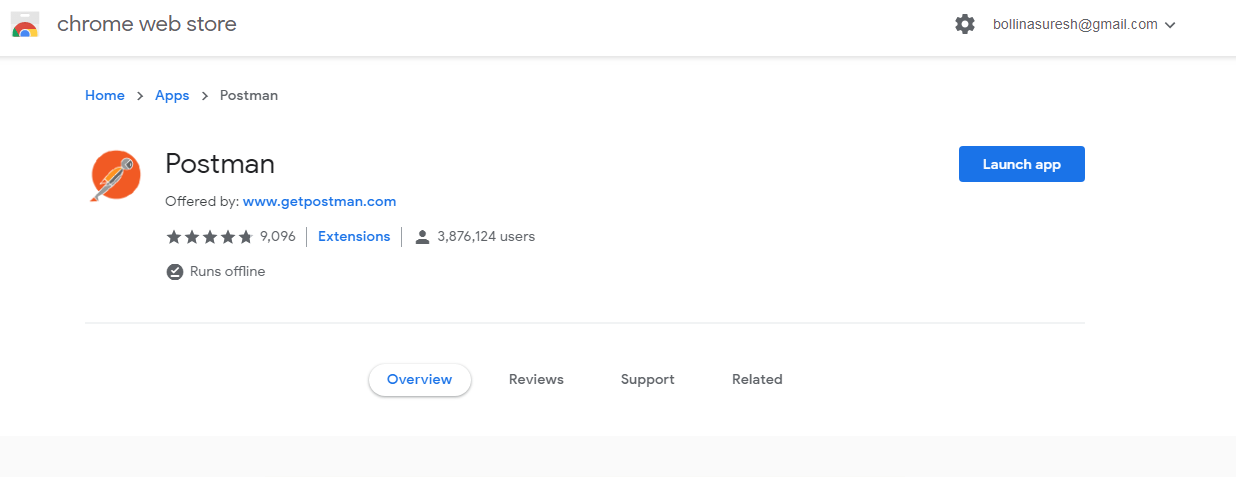
* 1. Click On extensions and see Postman is added successfully or not



* 1. Click on POSTMAN details



* 1. Click on Launch App to start the POSTMAN



**http://localhost:4444/wd/hub/session/1ae1fd730b1e8fc36f2e691eff9fcaf8/element/0.9498212264096286-1/value**

**{**

**"value" : ["selenium"]**

**}**

**http://localhost:4444/wd/hub/session/1ae1fd730b1e8fc36f2e691eff9fcaf8/element/0.9498212264096286-2/submit**

**References:**

WebDriver <https://hexdocs.pm/webdriver/WebDriver.Protocol.html>

WebDriver <https://w3c.github.io/webdriver/>

Client - Server Model <https://en.wikipedia.org/wiki/Client%E2%80%93server_model>

JsonWireProtocol <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol>

HTTP <https://developer.mozilla.org/en-US/docs/Web/HTTP>