**Library API:**

Base URI: <https://rahulshettyacademy.com>  
  
 (or) http://216.10.245.166

1. **Method**: POST

**Add Book Complete URL** - <https://rahulshettyacademy.com/Library/Addbook.php>

**Input Payload: Json**:

{

"name":"Learn Appium Automation with Java",

"isbn":"bcd",

"aisle":"227",

"author":"John foe"

}

**Output Json**

{

"Msg": "successfully added",

"ID": "bcd227"

}

1. **Resource** : /Library/GetBook.php?AuthorName=somename **Method** : GET

**Output Json**:

Output the array of Json object books with all below details

{

Name : “bookname”   ( String)

Isbn :  “A2fdsf”   (String)

Aisle : 32 (Integer)

}

1. **Resource**: Library/GetBook.php?ID=3389      - **Method** : GET

**Output Json:**

{

"book\_name": "Selenium automation using Java",

"isbn": "a23hd738",

"aisle": "1223"

}

1. **Resource** :/Library/DeleteBook.php      **Method** : POST

**Input Payload: Json:**

{

"ID" : "a23h345122332"

}

**Output Response**:

{  
  
msg : book is successfully deleted”

}

Automation Scenarios to Test Library API -

Verify if API responses returns Success Codes with Proper Assertions

Verify if Response Json Schema is displayed as expected

Create Environment/Global/Collection variables to dynamically switch end points of APIs to test in different stages of QA Life cycle

Pass response ID of Add Book to Get Book and Delete ID request Parameters for full functional Automation Testing

Validate the book ID response calculation Logic

Validate if Get Book API retrieves the correct response with Book Details requested

Validate if Delete Book API successfully deleted Book

Implement Try Catch Error Handling Mechanism for every Automation test as Tear Down Script

If Book Already exists in DB, Implement Smart Strategy to delete the existing Book first before Adding the Book again as Prerequisite Automation Step

Parse the complete Json response and verify if the values are displayed as expected

Generate Unique ISBN/ aisle value for every run to make Book Unique using Automation Script

Import the Book Details from CSV/Json without hard coding for Validating API’s

Run the APIs with multiple data sets by iterating the data from the CSV as a Data driven testing

//

What are Environments and variables in Postman?

An environment is a set of [variables](https://learning.postman.com/docs/sending-requests/variables/) you can use in your Postman requests. You can use environments to group related sets of values together and manage access to shared Postman data if you are working as part of a team.

Variables allow you to store and reuse values in your requests and scripts. By storing a value in a variable, you can reference it throughout your collections, environments, and requests—and if you need to update the value, you only have to change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

How to use Environments and Variables in Postman?

Variable scopes

Postman supports the following variable scopes:

* Global
* Collection
* Environment
* Data
* Local

Scripting in Postman

Postman contains a powerful runtime based on Node.js that allows you to add dynamic behavior to requests and collections. This allows you to write test suites, build requests that can contain dynamic parameters, pass data between requests,

Diagram

Description automatically generated

## The pm object

You will carry out most of the Postman JavaScript API functionality using “pm” which provides access to request and response data, and variables.

Verify if there is an entry called Cypress.

Verify if Cypress entry has course Title and Price properties/keys

Verify if sum of API Courses equals to 90

Verify if Web Automation Course titles are shown as expected titles

**Postman can also test Soap Webservices in addition to Rest HTTP requests**

How Soap Webservices are different from Rest Services?

Rest API’s use HTTP protocol to send the request and receive the response

Whereas Soap Webservices/API’s use Soap Protocol to send the request ad receive the response

In Soap Services, requests/responses are sent in XML Format.

What will we learn from this Course?

How to Call Soap Services from Postman- Understand the rules of setting up Soap Project in Postman

How to write automation tests on the XML response of Soap Services.

Soap End Point example  
https://www.dataaccess.com/webservicesserver/NumberConversion.wso

What is Newman?

* Newman is a command line Collection Runner for Postman.
* It allows you to run and test a Postman Collection directly from the command line.
* Using Newman, you can easily integrate it with your continuous integration servers and build systems.
* Newman is built on Node.js. To run Newman, Install Node.js as prerequisite.
* After Node.js install, Install Newman from npm globally on your system, which allows you to run it from anywhere.  
  npm install -g newman
* How to run collection from Newman?

$ newman run <CollectionFile>

* Learn about Newman Configuration options
* Generate HTML reports for Test execution results with newman - htmlextra plugin  
  <https://www.npmjs.com/package/newman-reporter-htmlextra>
* Integrate Postman Automation Scripts to Jenkins CI/CD with the help of Newman commands

Trigger the Postman Automation Scripts from Terminal with the help of Newman CLI

Integrate the Automation Tests with Jenkins for CI/CD Implementation

Prepare neat HTML reports for the Postman API Test Automation results

Understand how to collaborate as a Team by forking the existing Project – Creating branches- Creating Pull requests – Merging