

2) Second link-Tricentis Tosca Fundamentals – Optimizing test automation with centralized test data (AS2):--In this two Part—I and II

Part I-lesson1

Test Sheet Creation

Definition of TestSheet Creation:

A "Test Sheet" typically refers to a structured document or template used in software testing to outline the test cases, test data, and expected results. It serves as a guide for testers to execute tests systematically.

Here we will focus on Centralizing our Test data and Testcases by Introducing Test Sheets and Templates.

We will also learn about API TESTING.

Lesson Challenge

X ✓	
	Subject: TestSheet Creation
	<p>Dear Tester,</p> <p>Your assignment is to test the size selector functionality in our WebShop. Your first step is to gather all the information that will be needed for testing.</p> <p>Sincerely, Your Test Manager</p>



This is our Requirement:

BOOKS | COMPUTERS | ELECTRONICS | APPAREL & SHOES | DIGITAL DOWNLOADS | JEWELRY | GIFT CARDS

CATEGORIES

- » Books
- » Computers
- » Electronics
- » Apparel & Shoes
- » Digital downloads
- » Jewelry
- » Gift Cards

MANUFACTURERS

- » Tricentis

NEWSLETTER

Sign up for our newsletter:

Subscribe

HOME / APPAREL & SHOES / MEN'S WRINKLE FREE LONG SLEEVE



Men's Wrinkle Free Long Sleeve
Availability: In stock

★★★★★
2 review(s) | Add your review

Size: (highlighted with orange circle)

24.00

Qty: 1 (highlighted with orange circle)

Add to wishlist Email a friend Add to compare list

f t g + 2

This Wrinkle Free Long Sleeve Dress Shirt needs minimum ironing. It is a great product at a great value!

Product tags

cool (18) . apparel (11) . shirt (3)

Customers who bought this item also bought



So we need to check this both positive and negative testcases with the help of Test Sheet.



Check

Sizes available – Verify that **valid options work**

Sizes unavailable – Verify that **invalid options do not work**



Create TestSheet

This makes it much easier to **maintain**, and later **automate**, the process of using the data in the TestCases

Lesson Objectives

1. Understand the **concept of TestSheets**
2. Identify the differences between **Attributes** and **Instances**
3. **Create a TestSheet**

What TestSheet Definition:

Definition



TestSheet

Basic framework of TestCaseDesign

It consists of **Attributes** and **Instances** which have their own values

These elements specify the different combinations of the different **TestCases** in a structured way

Explain them in detail:

The screenshot shows the Tosca Copilot interface with the 'TestCaseDesign' workspace selected. The left sidebar lists various test cases and modules. In the center, a table is displayed with rows labeled '1 Create Log In Data TestSheet', '4a Set Conditions', '7 Calculate Discount', and '8 API'. One row under '8 API' is selected, showing columns for 'Name' (Tosca Copilot) and 'Value'. A context menu is open over this row, with several items circled in orange: 'Create Instances from Values (Ctrl+N, Ctrl+V)', 'Step' (under 'Modify'), and 'Just the Instance' (in the list). The 'Step' item is highlighted with a blue selection bar.

In General our Excel sheet:--Row wise it will pass the DATA

	A	B	C	D	E	F
1	Scenario	Username	Password			
2	Verify order process in demowebshop with user1	u1	p1			
3	Verify order process in demowebshop with user2	u2	p2			
4	Verify order process in demowebshop with user3	u3	p3			
5	Verify order process in demowebshop with user4	u4	p4			
6						
7						
8						
9						
10						
11						

Now in TOSCA—Column wise it will pass the DATA

Name	TestSh...	TestSheet_1	TestSheet_userna...
TestSheet_username and pwd	TestSheet	TestSheet_1	TestSheet_username...
Instances	TestSheet	TestSheet_1	
TestSheet			
TestSheet_1			
TestSheet_username and pwd			
UN	monika	Kavya	
Names	monika		
monika			
Kavya			
PWD	Ugand...	6770898	
12345	Ugandhar - Personal		
Ugandhar - Personal			
6770898			

Screenshot of a software interface showing a tree view on the left and a details table on the right.

Left Panel (Tree View):

- TestCaseDesign
 - TestCaseDesign_both ways folder with in and out
 - TestCaseDesign_UN&pwd
 - TestSheet_username and pwd
 - Instances
 - TestSheet
 - TestSheet_1
 - UN
 - PWD
 - Verify order process in demowebshop
 - TestCaseDesign-sizes of sheet
 - TestSheet
 - Instance-Multiple
 - Valid values
 - Size Medium
 - Size Large

Right Panel (Details Table):

Name	TestSh...	TestSh...
TestSheet_username and pwd	TestSheet	TestSheet_1
Instances		
TestSheet		
TestSheet_1		
UN		
monika	monika	Kavya
monika	monika	
Kavya		
PWD		
12345		
Ugandhar - Personal	Ugand...	6770898
6770898		

Annotations:

- Red circles highlight specific items in the tree view: "TestCaseDesign_both ways folder with in and out", "TestCaseDesign_UN&pwd", "TestSheet_username and pwd", "Verify order process in demowebshop", and "TestCaseDesign-sizes of sheet".
- A red circle highlights the "Instances" row in the table.
- A red circle highlights the "TestSheet" row in the table.
- A red circle highlights the "TestSheet_1" row in the table.
- A red circle highlights the "UN" row in the table.
- A red circle highlights the "monika" row in the table.
- A red circle highlights the "Kavya" row in the table.
- A red circle highlights the "PWD" row in the table.
- A red circle highlights the "12345" row in the table.
- A red circle highlights the "Ugandhar - Personal" row in the table.
- A red circle highlights the "6770898" row in the table.
- A red circle highlights the "Create Instance value for it." text.
- A red circle highlights the "It is a single instance -- select 1" text.
- A red circle highlights the "This is an Attribute" text.

Then this is another example:

Screenshot of a software interface showing a tree view on the left and a details table on the right.

Left Panel (Tree View):

- TestCaseDesign
 - TestCaseDesign_both ways folder with in and out
 - TestCaseDesign_UN&pwd
 - TestSheet_username and pwd
 - Instances
 - TestSheet
 - TestSheet_1
 - Verify order process in demowebshop
 - Instances
 - Verify order process in demowebshop
 - Verify order process in demowebshop
 - Email:_1
 - Instances
 - p.monikanaidu@gmail.com
 - ravi@gmail.com
 - user1@gmail.com
 - user2@gmail.com
 - Ugandhar@1975

Right Panel (Details Table):

Name	Verify order ...	Verify order ...	Verify or...	Verify order pr...
Verify order process in demowebshop	Verify order pr...	Verify order p...	Verify order ...	Verify order proc...
Instances				
Verify order process in demowebshop				
Verify order process in demowebshop				
Verify order process in demowebshop				
Verify order process in demowebshop				
Email:_1				
Instances				
p.monikanaidu@gmail.com	p.monikana...	ravi@gma...	Ugandha...	user2@gmail...
ravi@gmail.com				
user1@gmail.com				
user2@gmail.com				
Ugandhar@1975				
Password				
Instances				
Ugandhar@2024	Ugandhar@...	798787@...	09989#lk...	In32333_1
700707@...@...				

What is there in video just as an Example:

The screenshot shows the Tosca Commander interface with the title "Tosca Commander: AS2_Lessons (C:\Tosca_Projects\TOSCA_Workspaces\AS2_Lessons.tws)". The menu bar includes PROJECT, HOME, VIEW, TOOLS, TESTCASEDESIGN, and API TESTING. The toolbar has various icons for operations like Cut, Copy, Paste, Delete, Modify, Attach File, Search, Project, Scratchbook, My Area, Section, Update all, Checkin all, Checkout, Import Subset, Export Subset, Exploratory Scenario, Manual Test Case, Automated Test Case, and REC.

The main workspace is titled "TestCaseDesign". It displays an "Order Process" instance with the following steps:

- Instances
 - Log In
 - Choose Product
 - Delivery Method
 - Credit Card Number
 - Verify Order
 - Log Out

To the right, a "Relations" table lists the associations between the Order Process and other entities:

Name	Jeans Ground	Jeans Air	Shirt Ground	Shirt Air
Order Process	Jeans Ground	Jeans Air	Shirt Ground	Shirt Air
Instances				
Jeans Ground				
Jeans Air				
Shirt Ground				
Shirt Air				
Log In	User1	User1	User 2	User 2
Choose Product	BlueJeans	BlueJeans	Shirt	Shirt
Delivery Method	Ground	Next Day Air	Ground	Next Day Air
Credit Card Number	12345678910	12345678910	12345678910	12345678910
Verify Order	Verify Order	Verify Order	Verify Order	Verify Order
Log Out	Log Out	Log Out	Log Out	Log Out

Our Sizes TestSheet



Attribute

Sizes



Instances

Valid: **S, M, L**

Invalid: **XL**

Definitions



Attribute

Expresses the **feature** of an object, person or functionality

e.g. hair color



Instance

Expresses the different **variants** of an Attribute

e.g. blond, brown, black

More Detail:

Functions

Attributes

Describe the **overall feature**

Parent Attribute can be seen as an **organizational element** for the SubAttributes

Functions



Instances

Can be placed on:

The **TestSheet level**, where they represent the **individual TestCases**

The **Attributes level**, where they represent the **variants of the Attribute** to which they are attached

Now Exercise-1

The screenshot shows a software interface with a navigation bar at the top. The 'TestCaseDesign' tab is selected. Below the navigation bar is a tree view of project components. Under 'TestCaseDesign', there is a folder named 'Shirts_sizes'. This folder contains a sub-node 'Sizes' which further contains four instances: 'Small', 'Medium', 'Large', and 'XLarge'. To the right of the tree view is a table titled 'Details' with a single row corresponding to the 'Shirts_sizes' node.

Name
Shirts_sizes

The screenshot shows a software interface with a top navigation bar containing various icons and buttons such as Paste, Cut, Copy, Duplicate, Delete, Modify, Search, My Area, Section, Update all, Checkin all, Checkout, Import Subset, Export Subset, Exploratory Scenario, Manual Test Case, and Automated Test Case.

The main area displays a table titled "Shirts_sizes" under the "Details" tab. The table has columns: Name, Valid Values, Size Medium, Size Large, and Size XLarge. The "Valid Values" column contains "Size Medium", "Size Large", and "Size XLarge". The "Size Medium" column contains "Size Large" and "Size XLarge". The "Size Large" column contains "Size XLarge". The "Size XLarge" column is empty.

A dropdown menu is open over the "Small" entry in the "Sizes" column, showing options: Small, Medium, Large, and XLarge. The "Medium" option is selected.

Invalid Instance vs Failed TestCase

Invalid Instance

Indicates that the TestCase should be **verifying** that the **correct error message** is shown when an Invalid Value is entered

Failed TestCase

Results when the values we are trying to verify are **different** from the **values expected** in the System Under Test

Useful Shortcuts

CTRL N T

Create new **TestSheet**

CTRL N A

Create new **Attribute**

CTRL N I

Create new **Instance**

Lesson -02 Templates

The Template will create a series of slightly different Test Cases that use the Template Test Case as a starting point.

Recap



Task

Test the **size selector** function

Individual TestCases?

Even with Libraries **repetitive process**

Also each TestCase requires **individual maintenance**

Solution: Create **TestCase Template**

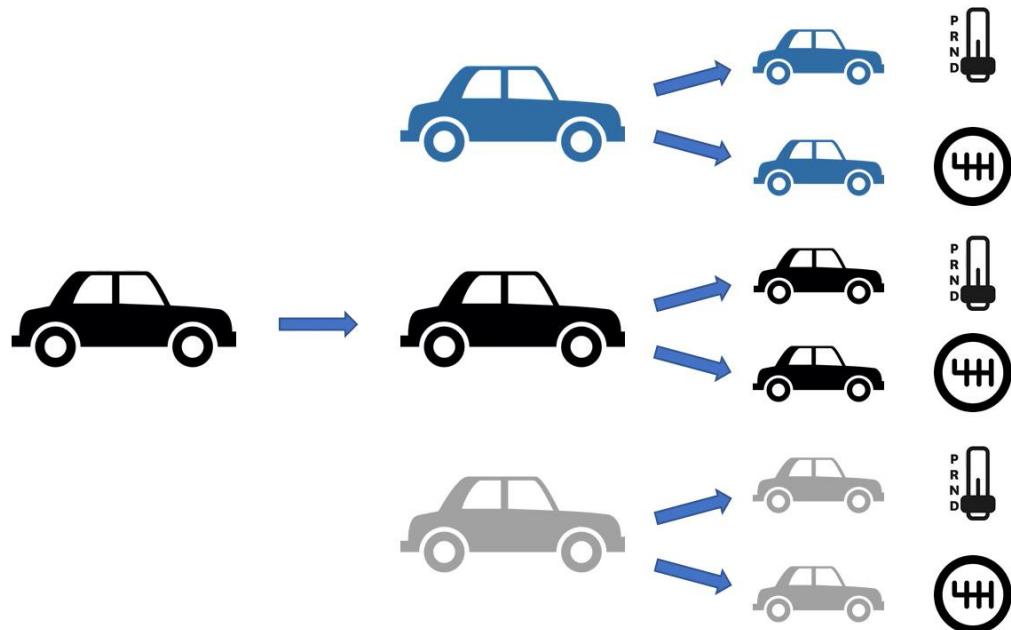
Very important: This Test case Template automatically create number of Testcases using the Test data contained in the Test Sheets.

Lesson Objectives

1. Define what Templates are and why we use them
2. Create a Template

Explain:

Templates for Cars



Definition



Template

Designed to **contain all possible paths** of a particular TestSheet

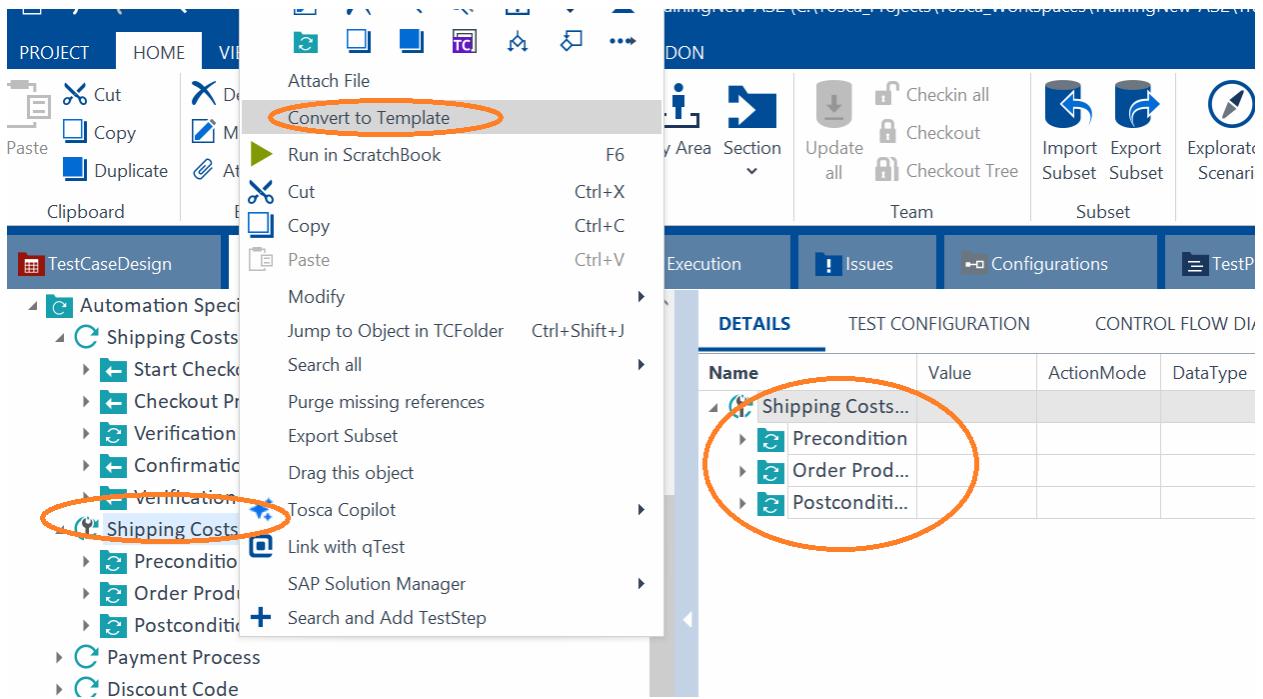
Allows to quickly **add, modify** or **remove** TestCases

Modifications will feed through to **all of the TestCases**

Covers the **risk areas** of our SUT in a time-efficient way

Now create a Test case and call References form the library or from Automation specialist.

Then you can convert this Test Case in to Template and vice versa (Remove references or not your choice)



Key Points

1. **Templates** are a **blueprint** for the **TestCases** which are built according to a **TestSheet**
2. **Templates** are created by **converting** existing **TestCases**
3. TestCases should be checked for **References before** being converted to Templates

Now create TCP at the Testcase level:

Name	Value	DataType
Browser	Chrome	String
Email	p.monikanaid.u@gmail.com	String
Password	Ugandhar@1975	String

Now you need to link Testsheet to Testcase and values from testsheet also like username and pwd---Please cross check it in Schema name-TESTSHEET name you can see.

Name	User1	User2	User3
Log In Data	User1	User2	User3
Instances	User1	User2	User3
Email	j.ray@tester1.test	b@wayne.test	s.ker@tester...
Instances	j.ray@tester1.test	b@wayne.test	s.ker@tester2.test
Password	Tosca123!	Tosca1234!	Tosca12345!
Instances	Tosca123!	Tosca1234!	Tosca12345!

Username and pwd dragged—show them

You can directly write the formula too.

Email ID--{XL[Value - Email]}

Password--{XL[PWD]}

Note—only valide Data it will take but not invalid data.

The screenshot shows a software interface for managing test cases. On the left, there's a sidebar with categories like TestCases, Library, Automation Specialist 1, Recovery Scenarios, Template, and Log In Process. The main area has tabs for Details, Test configuration, and Control Flow Diagram. Under Details, there's a table with columns Name, Value, ActionMode, and DataType. A row for 'Log In Process' is selected, showing its Precondition, Workflow, and Log In steps. The Log In step has fields for Email (containing {XL[Email]}), Password (containing {XL[Password]}), and Remember me (checkbox checked). Below these are Postcondition_Refer... and Postcondition_Reference... sections. To the right, there's a 'Relations' tab showing a grid of User1, User2, and User3 instances. Another section shows Email and Password data with their respective instances and values.

Then you need to Executable instances with the help of



Your Testcase:

TestCases

- Library
- Automation Specialist 1 TestCases
- *** Recovery Scenarios ***
- Template
 - 2a Log In Process
 - 2b Resolve References
 - 2c Convert to Template
 - 3a Link Template and Values
 - Log In Process
 - Precondition
 - Workflow
 - Postcondition_Refere...
 - 3b Instanti...

Log In Process

Precondition

Workflow

Postcondition

Create WHILE Statement (Ctrl+N,Ctrl+W)

Create Recovery Scenario Collection (Ctrl+N,Ctrl+R)

Create TemplateInstance (Ctrl+N,Ctrl+I)

Create TestCase (after this) (Ctrl+,)

The screenshot shows a software interface with a left sidebar and a main content area. The sidebar contains a tree view of test cases:

- Library
- Automation Specialist 1 TestCases
- *** Recovery Scenarios ***
- Template
 - 2a Log In Process
 - 2b Resolve References
 - 2c Convert to Template
 - 3a Link Template and Values
 - Log In Process
 - Precondition
 - Workflow
 - Postcondition_Reference
 - 3b Instantiation
 - Log In Process
 - TemplateInstance of Log In Process
 - User1
 - User2
 - User3

The 'TemplateInstance of Log In Process' node under '3b Instantiation' is circled in red.

The main content area shows a 'TEST CONFIGURATION' table:

Name	Value	Action
TemplateInstance of Log I...		
User1		
User2		
User3		

How many user you have that many can be seen here.

And Run this in the Scratch Book and view results.

Lesson 04-Modifications of Template

New Function



SUT

Shows a message after a **number** of shirts has been **entered** and the **add to the cart button** has been clicked

Shows the user if they entered a **valid quantity** or an **invalid** one



Add to Cart

This feature adds to a part of the SUT that was already tested

You have Modules, Templates, TestCases and TemplateInstances

A new TestSheet, including the **Verification Attributes** that are needed to verify the message from the SUT

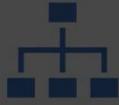
Lesson Objectives

1. Define what a **Verification Attribute** is
2. Modify an existing Template
3. Restantiate a modified Template

The screenshot shows the TestCaseDesign interface with a TestSheet open. The left sidebar lists modules like 'Shirts_sizes' and 'Shirts_quantity_with_validation'. The main area has tabs for 'Details' and 'Relations'. A table titled 'Shirts_quantity_with_validation' is displayed with columns for 'Name', 'Valid quantity', and 'Invalid quantity'. The table contains rows for 'Valid quantity', 'Invalid quantity', 'Quantity' (with instances 4 and -4), and 'Verification' (with instances 'The product has been added to your shopping cart' and 'Quantity should be positive').

Name	Valid quantity	Invalid quantity
Valid quantity		
Invalid quantity		
Quantity	4	-4
Instances	4	-4
Verification	The product has been added to your shopping cart	Quantity should be positive
Instances	The product has been added to your shopping cart	Quantity should be positive

Remember



XL Links

If the **Attributes** and **Instances** have been named **exactly the same way** in both the old and the new TestSheet, then Tosca will **update** the links **automatically**

These should, however, be **carefully checked**

The screenshot shows the Test Case Designer interface with two open test cases:

- TestCases** (Left pane): A tree view of test cases, including "TestCases", "Order Shirts", "Order Product", "Top menu", "Postcondition_Reference", and "TemplateInstance of Order Shirts".
- TemplateInstance of Order Shirts** (Center pane):
 - Details** tab: Shows a table with columns: Name, Value, ActionMode, and DataType. The table has one row for "Quantity" with Value "5", ActionMode "Input", and DataType "String".
 - Test configuration** tab: Shows a detailed configuration for "Quantity":
 - Valid quantity: 5
 - Precondition_Reference: Add to cart
 - Order Product: Navigate to Apparel and Shoes, Navigate to Shirt
 - Order Shirt: Validate Add To Cart Message
 - Top menu
- Shirts_quantity_with_validation** (Right pane):
 - Details** tab: Shows a table with columns: Name, Valid quantity, Invalid quantity, and Quantity 6. The table has three rows:
 - Valid quantity: 5
 - Invalid quantity: -4
 - Quantity 6
 - Relations** tab: Shows a table with columns: Instances, Valid quantity, Invalid quantity, and Quantity 6. The table has three rows:
 - Instances: 5, Valid quantity: 5, Invalid quantity: -4, Quantity 6: 6
 - Instances: -4, Valid quantity: 5, Invalid quantity: -4, Quantity 6: 6
 - Instances: 6, Valid quantity: 5, Invalid quantity: -4, Quantity 6: 6
 - Validation** tab: Shows a table with columns: Instances, The product has..., Quantity should..., and The product has.... The table has two rows:
 - Instances: The product has bee..., The product has been...
 - Instances: Quantity should be p...

Lesson -05 Conditions

Conditions

Conditions instruct Tosca on when to use (or not to use) certain **TestStepFolders**, **TestSteps** or **Values**

A Condition on a **TestStepFolder** acts as a rule for when that **entire Folder** would be run

A Condition on a **TestStepValue** is applicable on that **specific Value**

Name	Valid Values	Invalid UserName	Invalid Password
Log In Data - Including Inv...	Valid Values	Invalid UserName	Invalid Password
Instances			
Valid Values			
Invalid UserName			
Invalid Password			
Value - Email	brunda987@...	WrongUserN...	s.ker@tester...
Instances			
WrongUserName			
s.ker@tester2.te st			
brunda987@gma il.com			
Value - Password	Brunda@987	Tosca1234!	Wrong Pass...
Instances			
Tosca1234!			
Wrong Password			
Brunda@987			

The screenshot shows a software interface for test case management. On the left, there is a tree view under 'TestCases' containing various scenarios and conditions. On the right, there are three tabs: 'TEST CONFIGURATION' and 'CONTROL FLOW DIAGRAM' (both currently inactive), and 'DETAILS'. The 'DETAILS' tab displays a table with columns: Name, Value, ActionMode, DataType, WorkState, and Condition. A red oval highlights a specific row in the 'Validate login Error Messages' section, which contains two entries: 'Wrong Email Error' and 'Wrong Password'. The 'Validation' column for both entries shows the value 'Success'.

Name	ActionMode	DataType	WorkState	Condition
Register		String		
Login	Input	String		
Shopping cart		String		
Wishlist		String		
Log out		String		
Process				
Login				
Email:	(XL[Value - Email])	Input	String	
Password:	(XL[Value - Password])	Input	String	
RememberMe		String		
Forgot password?		String		
Log in	Input	String		
Validate login Error Messages				Validation == "Success"
Wrong Email Error	InnerText == (XL[Validation])	Verify	String	Validation == "Please ..."
Wrong Password	InnerText == (XL[Validation])	Verify	String	Validation == "Login ..."
PostCondition				

Lesson -06 Run & Report Automated

The screenshot shows a software interface for requirements management. On the left, there is a tree view under 'Requirements' containing a folder 'DemoWebShop' with sub-items like 'WebShop Frontend', 'Customer tasks', 'Log In', and 'Handle products'. On the right, there are several tabs: PROJECT, HOME, VIEW, TOOLS, REQUIREMENTS, API TESTING, and ADDON. The 'REQUIREMENTS' tab is active and displays a table with columns: Name, Description, Freq..., Dam..., Weight, Contribution (%), Coverage Specified (%), and Execution State. A red arrow points from the 'Customer tasks' node in the tree to the 'Valid Values' entry in the table, which has a status of '0'.

Name	Description	Freq...	Dam...	Weight	Contribution (%)	Coverage Specified (%)	Execution State
WebShop Frontend						100	
Customer tasks		4	3	128	0	100	
Register		3	4	128	0	100	
Log In		5	4	512	0	100	
Valid Values				1	0		
Invalid UserNa...				1	0		
Invalid Password				1	0		
Modify customer d...		3	2	32	0	100	
Check orders		2	3	32	0	100	
Handle products		4	4	256	0	100	
Shopping cart		5	5	1024	0	100	
Order process		5	5	1024	0	100	

Lesson -07 Build Templates and links ----Similar we do

(Like Earlier we Link Test Cases → Requirement

And Execution→Requirement)

Name	Value	ActionMode	LoginInfo	Accessibility
ActualLog		3		
TemplateInstance of L...		3		
Valid Values				
Invalid UserName				
Invalid Password				

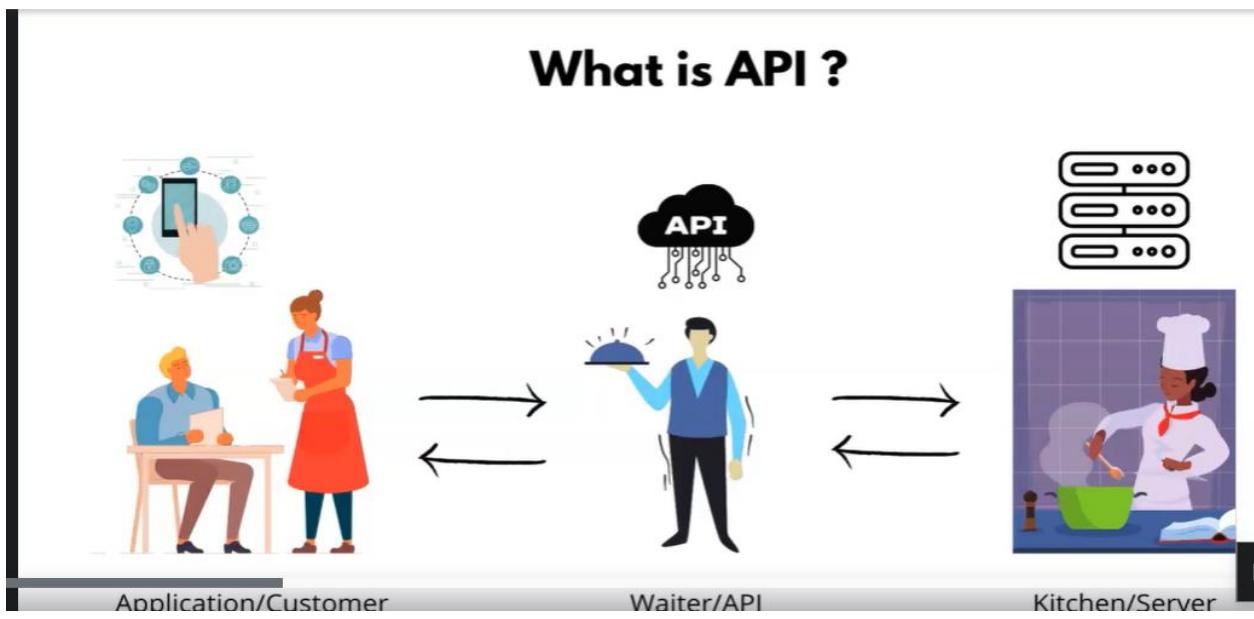
Name	Value	ActionMode	LoginInfo	Accessibility
ActualLog		3		05
TemplateInstance of L...		3		05
Valid Values				05
Invalid UserName				05
Invalid Password				05

Assignment for AS2:=

<https://thinking-tester-contact-list.herokuapp.com/>

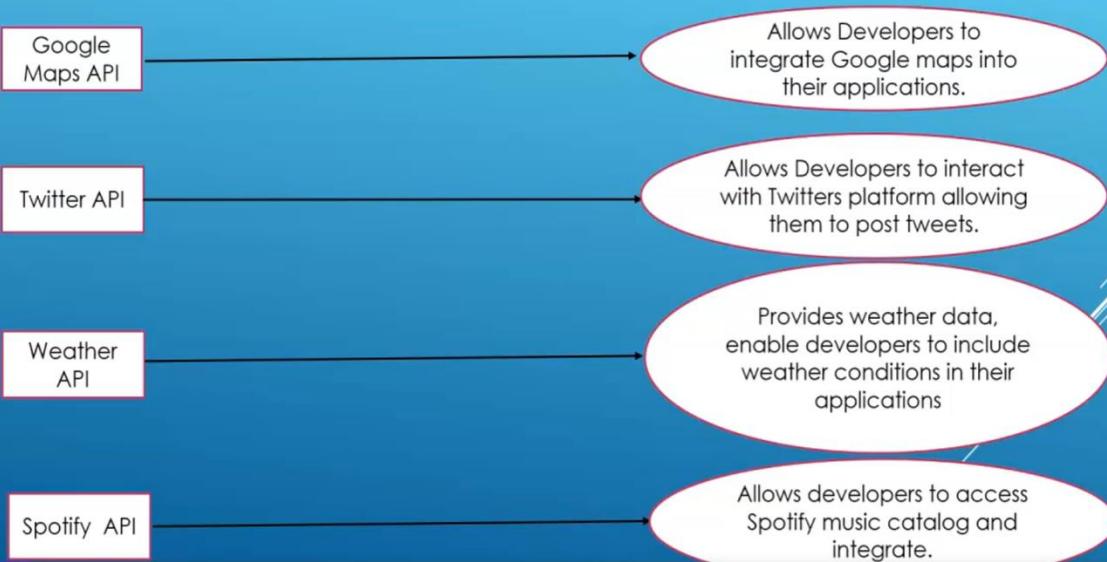
Lesson -08 API's

II)API TESTING(Application Programming Interface)---Full Basic to Advance:



Examples of API's and it's benefits

API EXAMPLES



ENHANCED FUNCTIONALITY

IMPROVED EFFICIENCY

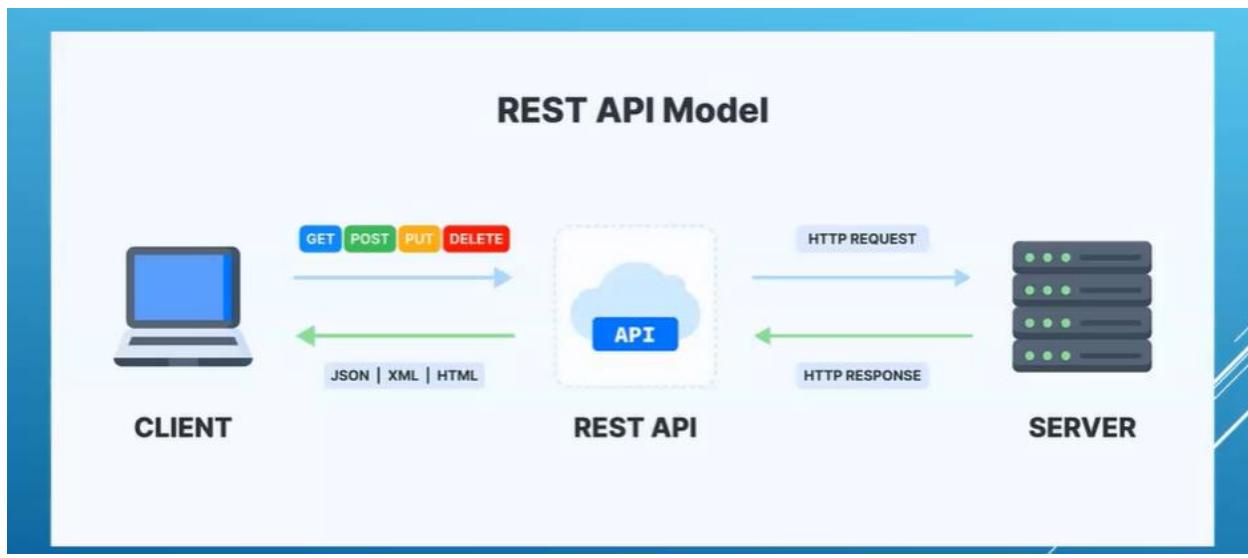
COST SAVINGS

BENEFITS OF
API'S

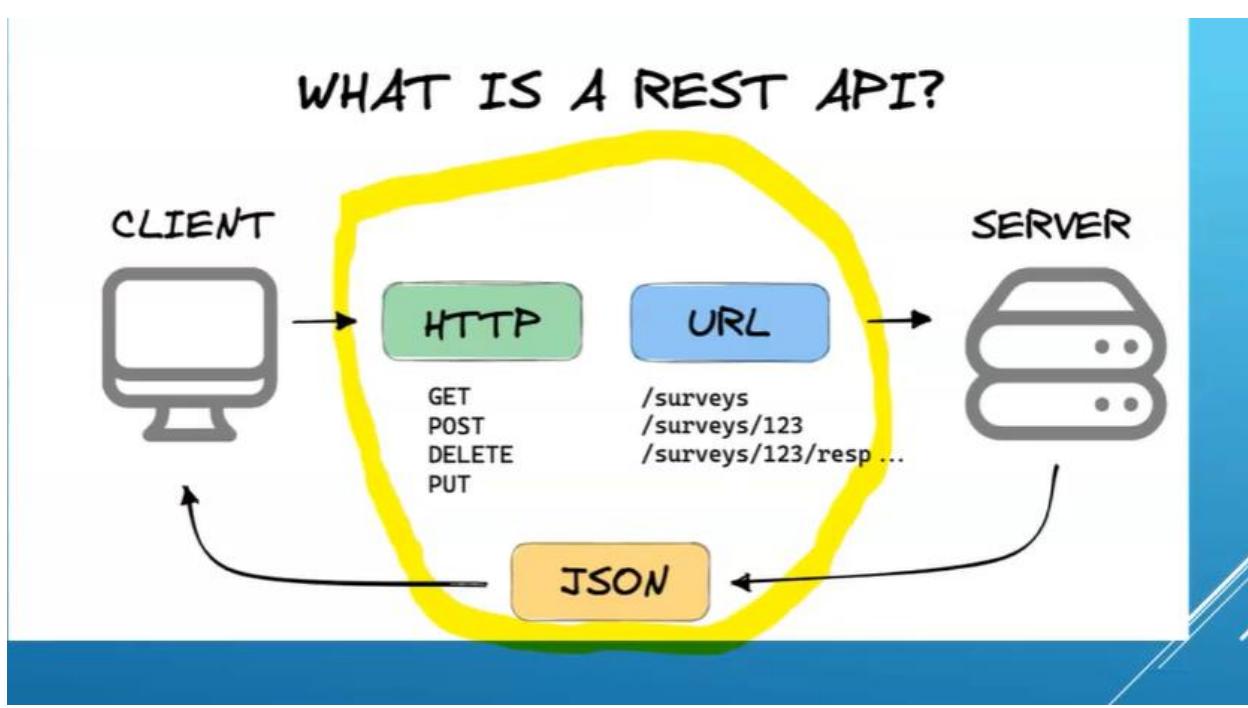
SCALABILITY

RELIABILITY AND
PERFORMANCE

CUSTOMIZATION AND
FLEXIBILITY



REST is a type of API. Not all APIs are REST, but all REST services are APIs.



KEY COMPONENTS- REST API

Resources

HTTP Methods

Headers

Status Codes

End Points

Parameters

Request Body

Response Body

Authentication and
Authorization

Documentation

Resources

Represents a piece of
information or object in
an API

Each resource is uniquely
identifiable.

Examples:

Resources

- **Definition:** The main entities that the API interacts with, such as users, posts, comments, etc.
- **URI:** Resources are identified using Uniform Resource Identifiers (URIs), typically in the form of `/resource_name`.

Example of a Resource

Let's consider an example of a "User" resource in a REST API.

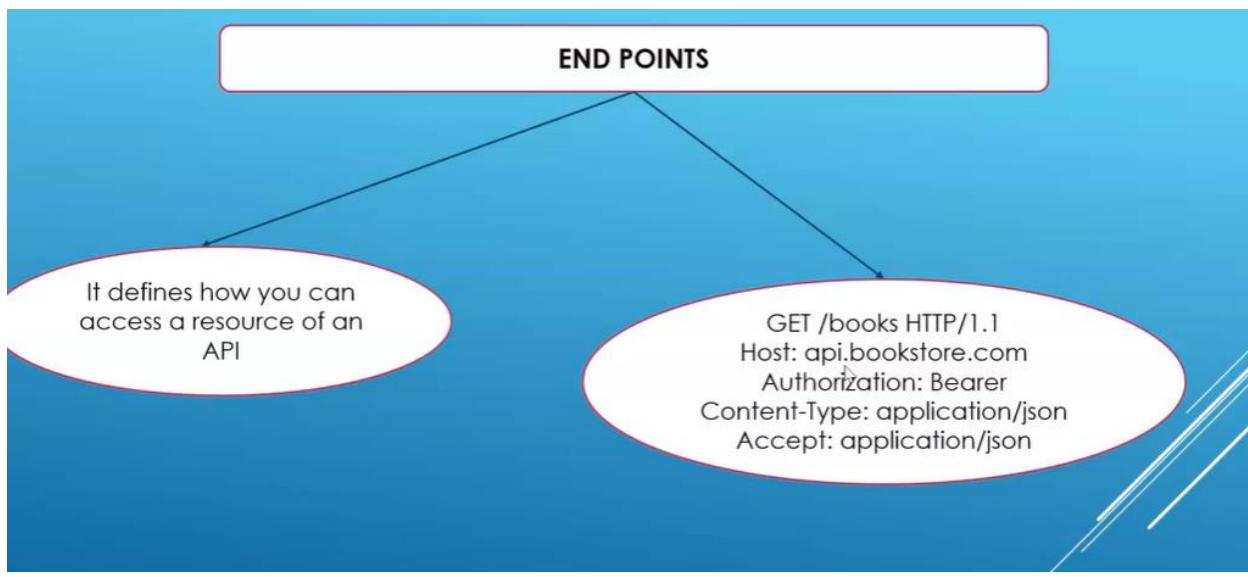
Resource URL

- **Collection of Users:** `https://api.example.com/users`
- **Single User:** `https://api.example.com/users/123`

One more Example:

2. HTTP Methods

- **GET:** Retrieve data from the server (e.g., fetch a list of users).
- **POST:** Submit data to the server to create a new resource (e.g., create a new user).
- **PUT:** Update an existing resource with new data (e.g., update user details).
- **DELETE:** Remove a resource from the server (e.g., delete a user).



Example Endpoints

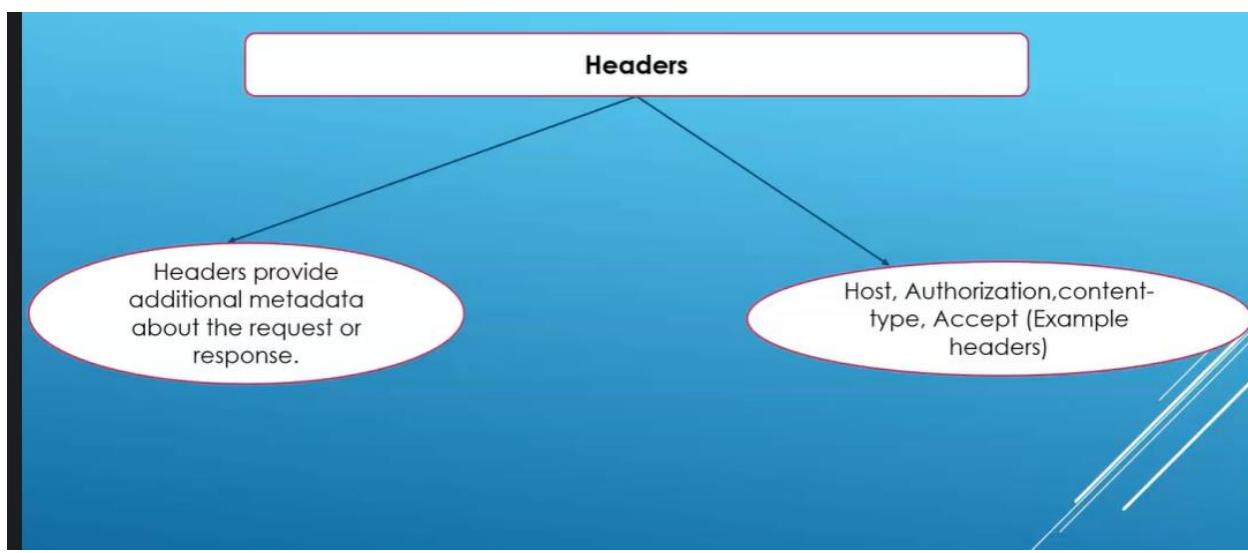
Get all books: GET /books <https://api.bookstore.com/books>

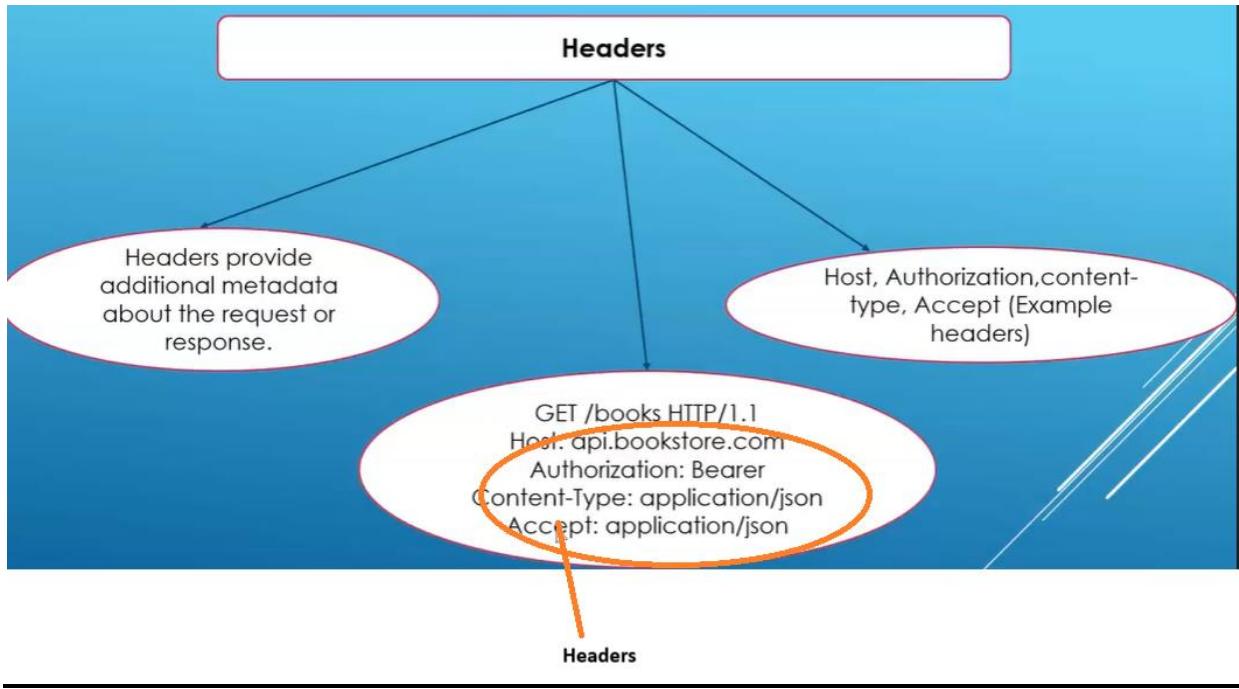
Get a specific book: GET /books/{bookId} <https://api.bookstore.com/books/1234>

Create a new book: POST /books <https://api.bookstore.com/books/7896>

Update a book: PUT /books/{bookId}

Delete a book: DELETE /books/{bookId}





Request:

Sample POST Request Message to create a new user in an API

POST /users HTTP/1.1

Host: api.example.com

Authorization: Bearer your_token_here

Content-Type: application/json

Accept: application/json

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.124 Safari/537.36

{"name": "John Doe", "email": "john.doe@example.com" }

Now—Server Response:

Sample Response message from the Server

HTTP/1.1 201 Created Date: Mon, 24 Jun 2024 12:00:00 GMT

Content-Type: application/json

Content-Length: 123

Location: <https://api.example.com/users/1>

Server: Apache/2.4.41 (Ubuntu)

```
{"id": 1, "name": "John Doe", "email": "john.doe@example.com", "created_at": "2024-06-24T12:00:00Z" }
```

Path Parameters:--single or multiple say her abc/345

Path Parameters:

Example

Get a specific book: GET /books/{bookId}

URL: <https://api.bookstore.com/books/abc/345>



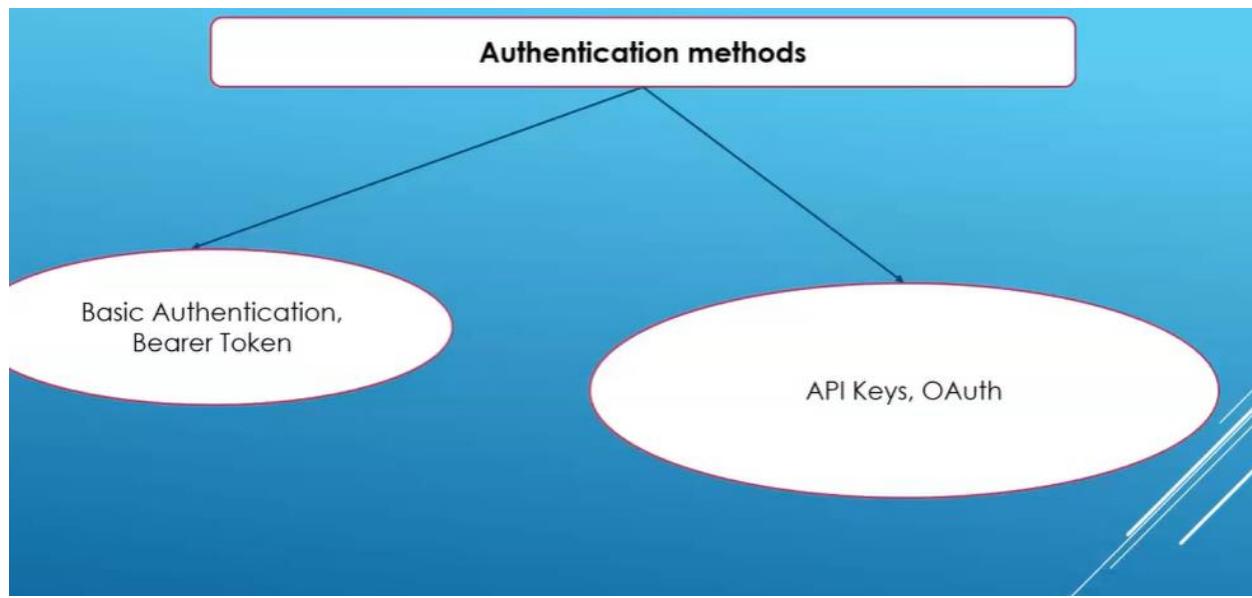
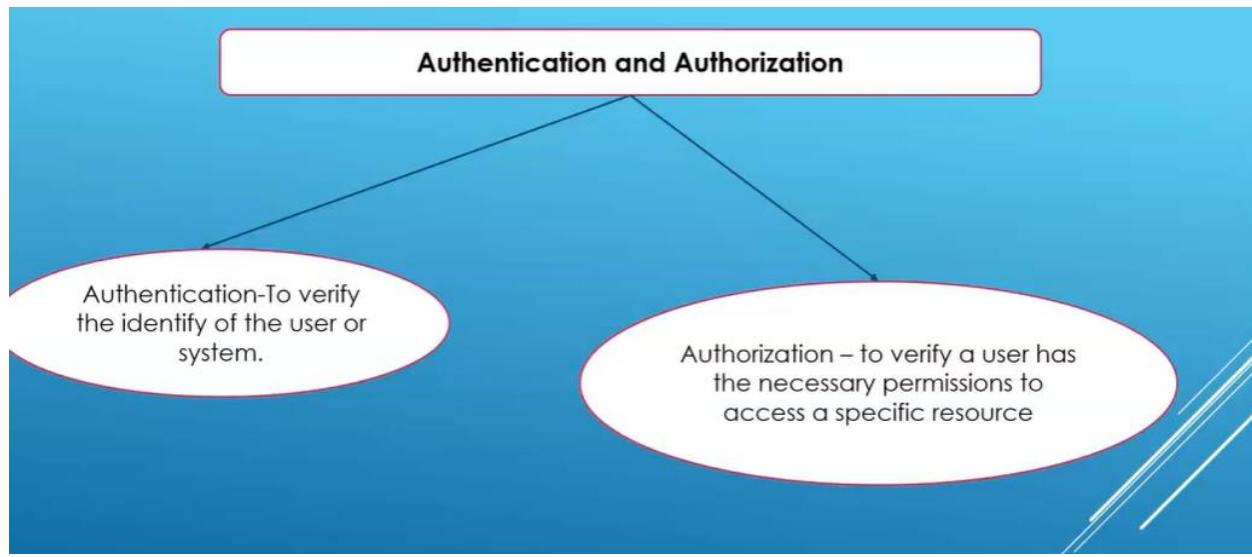
Query Parameters

Example

Get all books with optional filters: GET /books

URL: <https://api.bookstore.com/books?page=2&limit=10&sort=title&filter=available>

Here, page, limit, sort, and filter are query parameters.



Status codes

201 status code –created a new resource and the request is successful.

200 status –successful request.

400 status code in the response message

404 –not found

415 –unsupported media type

500 –Internal server error—

501 –Not Implemented—the server doesn't support the functionality

503—service unavailable—the server is not ready to handle the request due to overloading

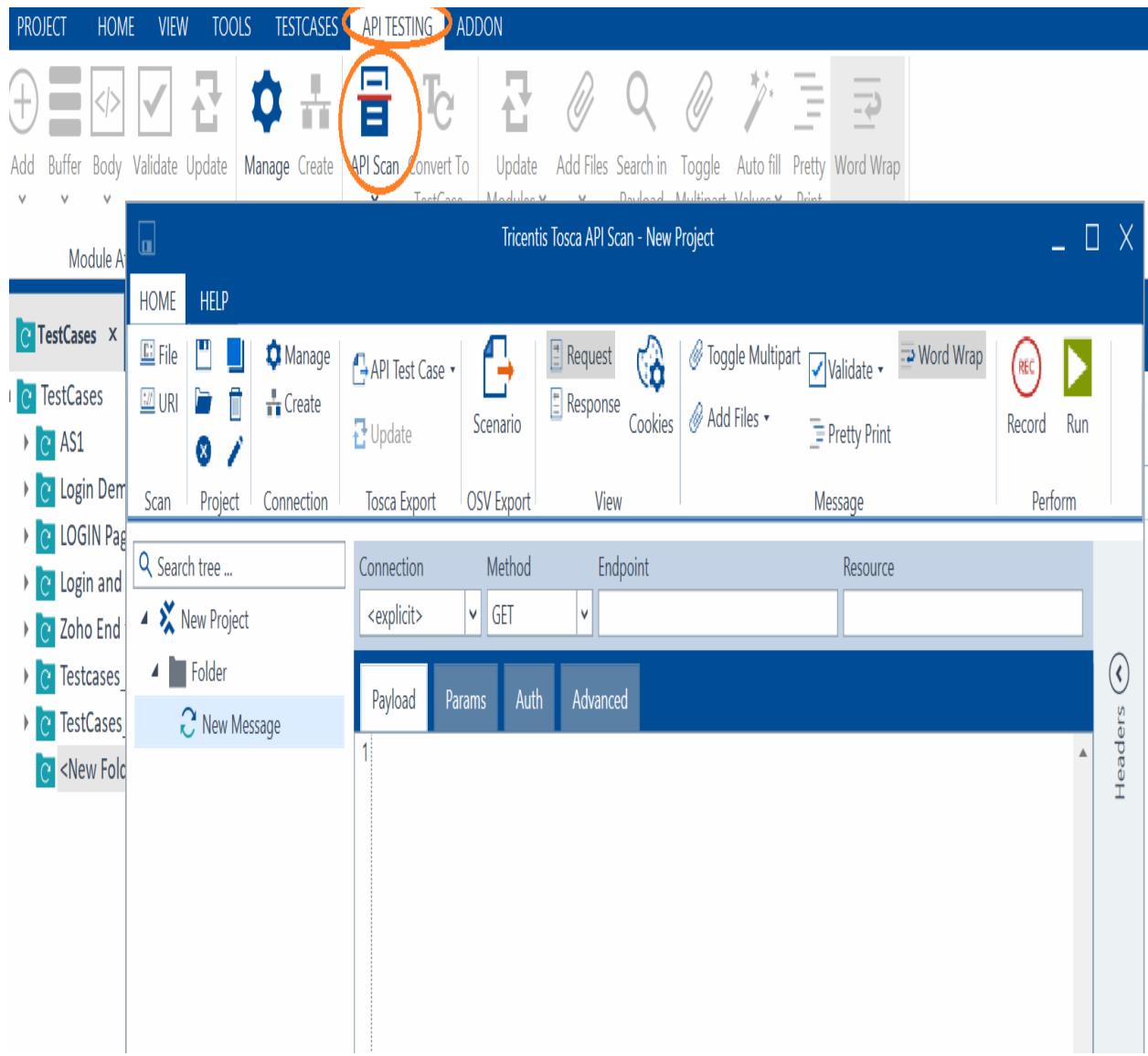
Now we start TOSCA-API TESTING:

Sample API Testing:--You can search in Goole:

Best site:--Use it

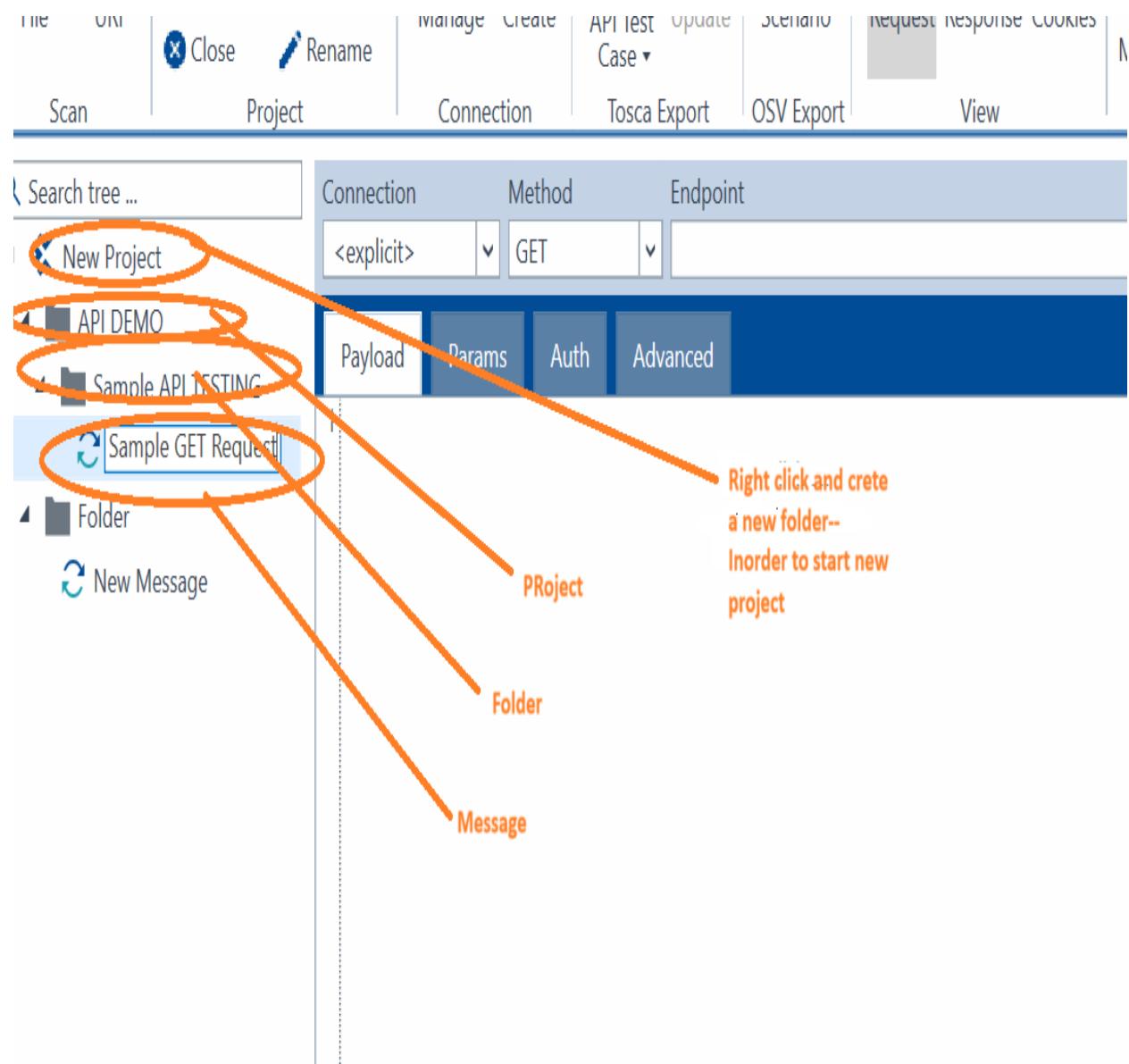
<https://reqres.in/>

Now in TOSCA: Go to API TESTING and select API Scan we see the below window.



Now start create in it:

1) Sample-GET



Example this:

The screenshot shows a user interface for making API requests. At the top right is a purple button labeled "SUPPORT REQRES". Below it is a horizontal line. To the left is a sidebar with icons for home, refresh, and navigation. The main area has two sections: "Request" and "Response".

Request: GET /api/users?page=2

Response: 200

The "Request" section shows a "LIST USERS" endpoint with a sub-section for "SIMPLE USER". It includes a placeholder text: "Design and Development tips in your inbox. Every weekday." and a "NOT FOUND" message. Below the endpoint name are "ADS VIA CARBON" and "SOURCE>" links.

The "Response" section displays a JSON object representing the API data. The JSON is as follows:

```
{  
    "page": 2,  
    "per_page": 6,  
    "total": 12,  
    "total_pages": 2,  
    "data": [  
        {  
            "id": 7,  
            "email": "michael.lawson@reqres.in"  
        }  
    ]  
}
```

<https://reqres.in/>----From this page

/api/users?page=2-----how to pass in TOSCA and run see

The screenshot shows the SoapUI application interface. In the top navigation bar, the 'HOME' tab is selected. The toolbar contains various icons for file operations (Save As, Duplicate, Open, Delete, Close, Rename), project management (Manage, Create, API Test Case, Update, Scenario), and tooling (Request, Response, Cookies, Toggle Multipart, Add Files, Validate, Pretty Print, Word Wrap). The 'Perform' section includes 'Record' and 'Run' buttons, with 'Run' being circled in orange.

The main workspace displays a project structure on the left with items like 'New Project', 'API DEMO', 'Sample API TESTING', and 'Sample GET Request'. The 'Sample GET Request' item is selected and expanded. The configuration pane shows a 'Connection' table with 'Method' set to 'GET' and 'Endpoint' set to '<https://reqres.in/>'. The 'Resource' field is set to '/api/users'. Below this, under the 'Params' tab, there is a table with a single row: 'Name' is 'page' and 'Value' is '2'. The 'Type' column for this row is 'Query'.

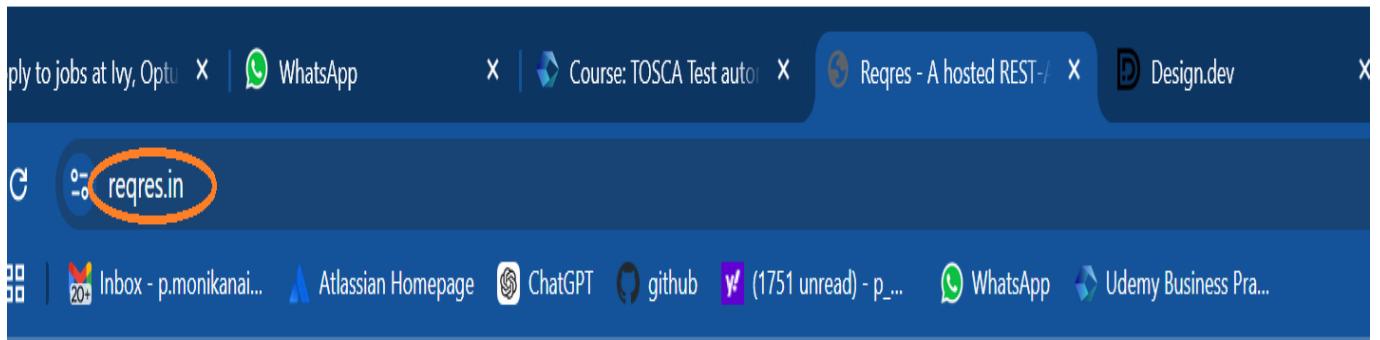
And Run and verify the results.

you can save it on Desktop .

All the rest the same way---Explain them in Detail.

2) Sample-POST

Now get Post Request and perform the same as GET:



SUPPORT REQRES

Request
`/api/users`

GET

LIST USERS

GET

SINGLE USER

GET

SINGLE USER NOT FOUND

GET

LIST <RESOURCE>

GET

SINGLE <RESOURCE>

GET

SINGLE <RESOURCE> NOT FOUND

Response
`201`

```
{  
  "name": "morpheus",  
  "job": "leader"  
}
```

```
{  
  "name": "morpheus",  
  "job": "leader",  
  "id": "906",  
  "createdAt": "2024-12-01T23:45:59.941Z"  
}
```

POST

CREATE

Design and Development tips

In TOSCA:

The screenshot shows the TOSCA application interface with several UI elements highlighted by orange circles:

- Toolbar:** Includes File, URI, Scan, Project, Connection, Tosca Export, OSV Export, View, Message, Perform, and various tool buttons like API Test Case, Scenario, Request, Response, Cookies, Toggle Multipart, Validate, Word Wrap, Record, and Run.
- Left Sidebar:** A tree view with "Search tree ..." at the top, followed by "New Project", "APISample1", "Folder", "APISample-GET", and "APISample-POST". The "APISample-POST" item is selected and highlighted with an orange circle.
- Request Configuration Area:** Contains fields for Connection (set to <explicit>), Method (set to POST), Endpoint (set to https://reqres.in), Resource (/api/users), and a Payload editor. The Payload editor shows the following JSON:

```
1 {  
2   "name": "morpheus",  
3   "job": "leader"  
4 }
```

- Bottom Buttons:** Save (highlighted with an orange circle) and Load buttons.

See the response

The screenshot shows a user interface for an API testing tool. On the left, there's a sidebar with a search bar and several project/folder entries, including 'APIsample1' and 'APISample-POST' which is currently selected. The main area has tabs for 'TOSCA EXPORT', 'CSV EXPORT', and 'VIEW'. In the 'VIEW' tab, the 'Status Code' is set to '201 Created' and the 'Response Time (ms)' is '638'. Below this, a 'Payload' section displays the JSON response:

```
1 {  
2   "id": "652",  
3   "createdAt": "2024-12-01T23:48:11.655Z"  
4 }
```

3) Sample PUT and DELETE

The screenshot shows a list of API operations. The 'PUT' method for the 'UPDATE' endpoint is highlighted with a large orange oval. To the right, a detailed view of the 'UPDATE' operation shows the 'Request' URL as '/api/users/2' and the 'Response' status code as '201'. The request payload is shown as a JSON object:

```
{  
  "name": "morpheus",  
  "job": "zion resident"  
}
```

The screenshot shows a REST API test tool interface. At the top, there are two main sections: "Request" and "Response". The Request section displays the URL "/api/users/2". The Response section shows the status code "204". Below these sections is a detailed list of test cases:

- GET**: LIST USERS
- GET**: SINGLE USER
- GET**: SINGLE USER NOT FOUND
- GET**: LIST <RESOURCE>
- GET**: SINGLE <RESOURCE>
- GET**: SINGLE <RESOURCE> NOT FOUND
- POST**: CREATE
- PUT**: UPDATE
- PATCH**: UPDATE
- DELETE**: DELETE
- POST**: REGISTER - SUCCESSFUL

4) Automation of sample test cases- response message validation—Under your Tosca Modules

Once done we Export our folder

The screenshot shows a test case management tool interface. The toolbar has several buttons: File, URI, Open, Delete, Close, Rename, Manage, Create, API Test Case (highlighted with a red circle), Update, Scenario, Cookies, Toggle Multipart, Add Files, Validate, Pretty Print, Word Wrap, Record, Run, and Perform. Below the toolbar, there is a dropdown menu for API Test Case with options: OSV Scenario, Export to Component Folders, and Resource. The main panel shows a list of projects: New Project, APIsample-POST, APIsample1, and Sample -APIs-TOSCA (highlighted with a red circle). The Sample -APIs-TOSCA project is expanded, showing sub-items: APIsample-Delete, APIsample-GET, APIsample-POST, APIsample-PUT, New Message_2, and New Message_3. The right side of the interface shows tabs for Payload, Params, Auth, and Advanced, and a Headers table.

WE get this---Explain Request and Response

The screenshot shows a software interface with a navigation bar at the top containing 'TestCases', 'Apiscan_Import', and 'Apiscan_Export'. Below the navigation bar is a tree view of API definitions:

- ApiScan_Import_4
 - Modules
 - Sample -APIs-TOSCA
 - APISample-Delete Request
 - APISample-Delete Response
 - APISample-GET Request
 - APISample-GET Response** (highlighted with a blue selection bar)
 - APISample-POST Request
 - APISample-POST Response
 - APISample-PUT Request
 - APISample-PUT Response
 - New Message_2 Request
 - New Message_2 Response
 - New Message_3 Request
 - New Message_3 Response
 - TestCases
 - Sample -APIs-TOSCA
 - APISample-Delete Request
 - APISample-Delete Response

See Status code is being added:

See Status code is being added:

Now you can see Status code under Test case –Show them both shortcut method and video method

The screenshot shows a software interface with a left sidebar containing a tree view of test cases. The tree includes nodes for 'New Message_2 Response', 'New Message_3 Request', 'New Message_3 Response', and a 'TestCases' folder which contains 'Sample -APIs-TOSCA' and several sub-nodes under it, each ending in 'StatusCode'. To the right of the sidebar is a 'DETAILS' table with columns for Name, Value, ActionMode, DataType, and WorkState. The table has two rows: one for 'APISample-POST Response' and another for 'StatusCode' with a value of '201 Created' and an 'ActionMode' of 'Verify'.

DETAILS		TEST CONFIGURATION	TECHNICAL VIEW	CONTROL FLOW DIAGRAM
Name	Value	ActionMode	DataType	WorkState
APISample-POST Response				
StatusCode	201 Created	Verify	String	

Now you can run and see the results as your previous TC how we were running till now:

The screenshot shows a software interface with a left sidebar containing a tree view of test cases. The tree includes nodes for 'APISample-POST R', 'APISample-PUT R', 'APISample-PUT Re...', 'APISample-PUT Re...', and a 'TestCases' folder which contains 'Sample -APIs-TOSCA' and several sub-nodes under it, each ending in 'StatusCode'. To the right of the sidebar is a 'DETAILS' table with columns for Name, Value, and Act. The table has two rows: one for 'Sample -APIs-TOSCA' and another for 'StatusCode' with a value of '204 No Co...' and an 'Act' status. On the far right, there is a 'ScratchBook' section showing a list of checked items corresponding to the test cases in the tree.

DETAILS		TEST CONFIGURATION
Name	Value	Act
Sample -APIs-TOSCA		
StatusCode	204 No Co...	

DETAILS		TEST CONFIGURATION
Name	Value	Act
ScratchBook		
APISample-Delete Re...		
APISample-Delete Re...		
APISample-GET Re...		
APISample-GET Re...		
APISample-POST Re...		
APISample-POST Re...		
APISample-PUT Re...		
APISample-PUT Re...		

Explain in detail:

The screenshot shows a test configuration interface with two main sections. On the left is a tree view of API endpoints under a 'ScratchBook' category. The endpoints listed are: APISample-Delete Request, APISample-Delete Response, APISample-GET Request, APISample-GET Response, APISample-POST Request, APISample-POST Response, APISample-PUT Request, and APISample-PUT Response. On the right is a detailed table of configuration settings for the 'ScratchBook' category. The table has columns for Name, Value, ActionMode, LoginInfo, and Accessibility.

Name	Value	ActionMode	LoginInfo	Accessibility
APISample-Delete Re...				
APISample-Delete Re...			Server Response Time: 414 ms	
APISample-GET Requ...				
APISample-GET Respo...			Server Response Time: 421 ms	
Status Code	201 Created	Verify	Verification was successful.	
APISample-POST Requ...			Verification was successful. Expected value == "201 Created" Actual value: "201 Created"	
APISample-PUT Requ...				
APISample-PUT Respo...			Server Response Time: 402 ms	

Additional Things — like you want to add few more fields and validate them you can do it.

Say ID or email etc add it in the similar way and validate it in the test case.

The screenshot shows a test configuration interface with two main sections. On the left is a tree view of API endpoints under a 'Status Code' category. The endpoints listed are: APISample-POST Request, APISample-POST Response, APISample-PUT Request, APISample-PUT Response, and TestCases. Under 'TestCases', there is a 'Sample - APIs-TOSCA' entry which contains: APISample-Delete Request, APISample-Delete Response, APISample-GET Request, and APISample-GET Response. On the right is a detailed table of configuration settings for the 'Status Code' category. The table has columns for Status Code and Response Time (ms). Below the table is a 'Payload' section showing a JSON response with an 'id' field highlighted.

Status Code	Response Time (ms)
201 Created	541

Payload

```
1 [  
2   {  
3     "id": "687",  
4     "createdAt": "2024-12-02T09:07:15.207Z"  
5   }  
6 ]
```

Now you can verify this way and run and check too.

The screenshot shows the Test Case Designer interface with the following details:

- TestCases** tab is selected.
- A project named **ApiScan_Import_5** is open.
- In the left sidebar, under **TestCases**, there are several API samples listed:
 - APISample-POST Response
 - APISample-PUT Request
 - APISample-PUT Response
 - TestCases
 - Sample -APIs-TOSCA
 - APISample-Delete Request
 - APISample-Delete Response
 - APISample-GET Request
 - APISample-GET Response
 - APISample-POST Request
 - APISample-POST Response
 - APISample-PUT Request
 - APISample-PUT Response
- DETAILS** tab is selected in the top navigation bar.
- TEST CONFIGURATION** tab is also visible.
- APISample-POST Response** is selected in the list.
- Value** column contains:
 - StatusCode: 201 Created
 - id: 687
- ActionMode** column contains:
 - StatusCode: Verify
 - id: Verify
- DataType** column contains:
 - StatusCode: String
 - id: String
- WorkState** column contains:
 - StatusCode:
 - id:

Tell them what happens Action Mode—Verify as it is Dynamic value---so we need to change it to Buffer.

You can validate anything or whole response body can also be validated in case you need it.

The screenshot shows the Test Case Designer interface with the following details:

- TestCases** tab is selected.
- A project named **ApiScan_Import_5** is open.
- In the left sidebar, under **TestCases**, there are several API samples listed:
 - APISample-POST Response
 - APISample-PUT Request
 - TestCases
 - Sample -APIs-TOSCA
 - APISample-Delete Request
 - APISample-Delete Response
 - APISample-GET Request
 - APISample-GET Response
 - APISample-POST Request
 - APISample-POST Response
 - APISample-PUT Request
 - APISample-PUT Response
- DETAILS** tab is selected in the top navigation bar.
- TEST CONFIGURATION** tab is also visible.
- APISample-Delete Response** is selected in the list.
- Value** column contains:
 - StatusCode: 204 No Content
- ActionMode** column contains:
 - StatusCode: Verify
- DataType** column contains:
 - StatusCode: String
- APISample-GET Request** is selected in the list.
- Value** column contains:
 - StatusCode: 201 Created
- ActionMode** column contains:
 - StatusCode: Verify
- DataType** column contains:
 - StatusCode: String
- APISample-GET Response** is selected in the list.
- Value** column contains:
 - StatusCode: 201 Created
- ActionMode** column contains:
 - StatusCode: Verify
- DataType** column contains:
 - StatusCode: String
- APISample-POST Request** is selected in the list.
- Value** column contains:
 - StatusCode: 201 Created
- ActionMode** column contains:
 - StatusCode: Verify
- DataType** column contains:
 - StatusCode: String
- APISample-POST Response** is selected in the list.
- Value** column contains:
 - id: 687
- ActionMode** column contains:
 - id: Buffer
- DataType** column contains:
 - id: String
- APISample-PUT Request** is selected in the list.

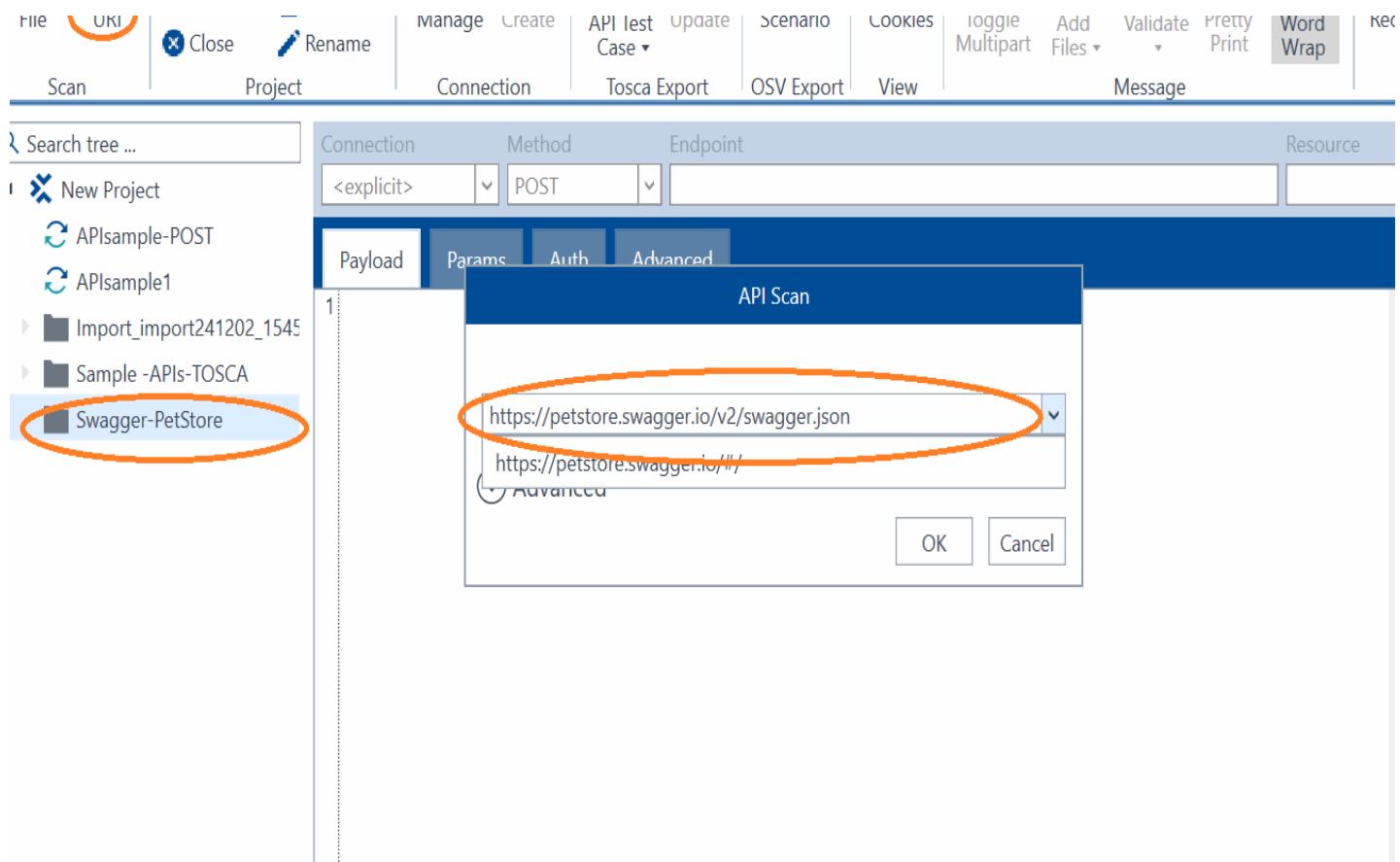
5)Scan API through Swagger URI - Automation of API tests

Login to :

<https://petstore.swagger.io/#/>

OR

you have a Swagger.json file you can call directly and perform the same action.



Once you done then same Run All API's and Export to Tosca and validate it etc same are above.

Note: Where every the body as per your API's you need to add you must add

The screenshot shows a REST API testing tool interface. At the top, there's a toolbar with various icons for file operations (Save As, Open, Duplicate, Close, Delete, Rename, Manage, Create, API Test Case, Scenario, OSV Export, View, Request, Response, Cookies, Toggle Multipart, Add Files, Validate, Pretty Print, Word Wrap, Record, Run). Below the toolbar, there's a search bar labeled "Search tree ..." and a status bar showing "Status Code: 200 OK" and "Response Time (ms): 986".

The main area has three tabs: "Payload" (selected), "Headers", and "Cookies". The "Payload" tab displays a JSON object with numbered lines:

```

1: {
2:   "available": 1,
3:   "sold": 31,
4:   "placed": 1,
5:   "string": 468,
6:   "available_TEST_339": 1,
7:   "pending": 18,
8:   "available_TEST_326": 1,
9:   "available": 406,
10:  "Not Available": 1,
11:  "PENDING": 1,
12:  "available": 1,
13:  "available_TEST_2": 2,
14:  "available_TEST_382": 1,
15:  "available_TEST_22": 1,
16:  "sold out": 1,
17:  "available_TEST_303": 1,
18:  "available_TEST_302": 1,
19:  "available_TEST_388": 1,
20:  "available_TEST_376": 1,
21:  "available_TEST_397": 1,
22:  "available_TEST_374": 2,
23:  "available_TEST_340": 1
24:

```

The "Headers" tab shows the following table:

Name	Value
Date	Mon, 02 Dec 2024 13:19
Transfer-Encoding	chunked
Connection	keep-alive
Access-Control-Allow-Origin	*
Access-Control-Allow-Methods	GET, POST, DELETE, PUT
Access-Control-Allow-Headers	Content-Type, api_key, Authorization
Server	Jetty(9.2.9.v20150224)
Content-Type	application/json

At the bottom right of the main area are "Save" and "Load" buttons.

5)Real time with REST API's-In GITHUB

Github Authentication---Show them the Test case of Github and use it in TOSCA as above what we have done.

Explain Below Example:-

The screenshot shows the SoapUI interface. On the left, there's a project tree with a node 'Invalid authentication' highlighted and circled in orange. In the main panel, a request configuration window is open. The 'Connection' tab shows 'GET' method and endpoint 'https://api.github.com/user'. The 'Auth' tab is selected, displaying an 'Authorization' field with the value 'Bearer ksdfsdfsdfsadfyuee'. To the right of this field, a 'Type' dropdown is set to 'Header', also circled in orange. The 'Payload' tab is visible at the bottom.

When you run invalid message see—

The screenshot shows the SoapUI interface after running the invalid authentication message. The project tree on the left still shows the 'Invalid authentication' node. In the main panel, the response status code is displayed as '401 Unauthorized' with a response time of '1047'. The response payload is shown in a code editor, containing JSON data: 'message': 'Bad credentials', 'documentation_url': 'https://docs.github.com/rest', 'status': '401'.

=====The End of AS2=====