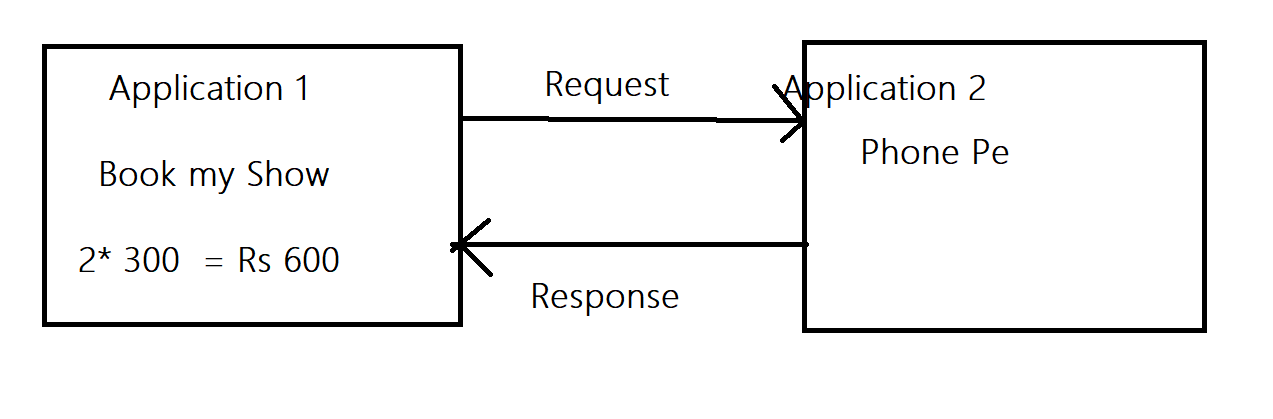
**API Testing**

(**Manually Test** / API Automation)

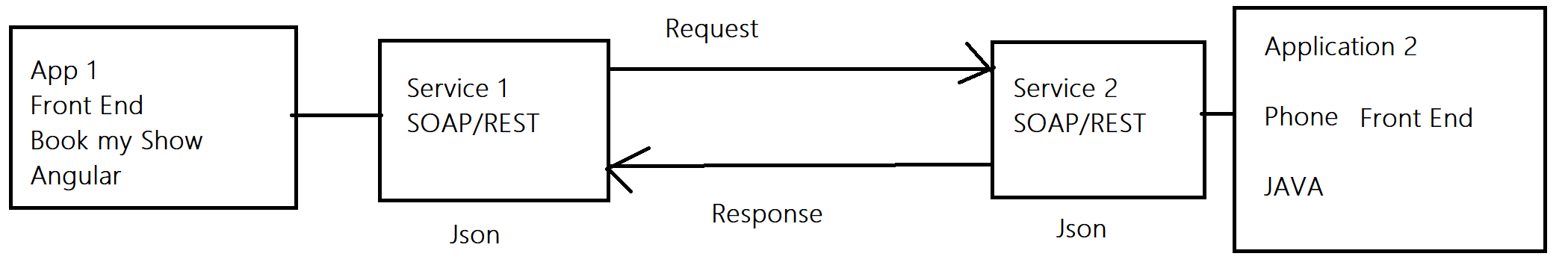
* API – Application Programming Interface.
* API is software intermediary that allows to communicate with each other.

**Web Service Communication**

* It is the communication between two applications.



Java language Angular Language

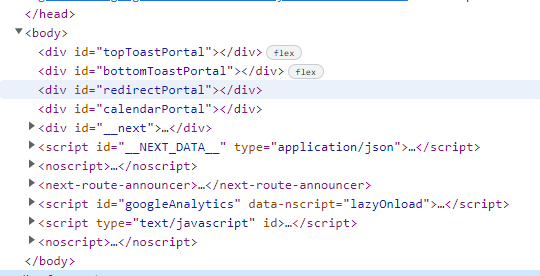


|  |  |
| --- | --- |
| **Manual Testing** | **API Testing** |
| 1. UI Required | 1. UI – Not required |
| 1. System and Functionality / Regression/retesting etc Performed | 1. Only functional Testing (Request and Response) |
| 1. Build / Application required | 1. URL/URI(Rest Services Testing)   WSDL (SOAP Service Testing) |

**Important Terms.**

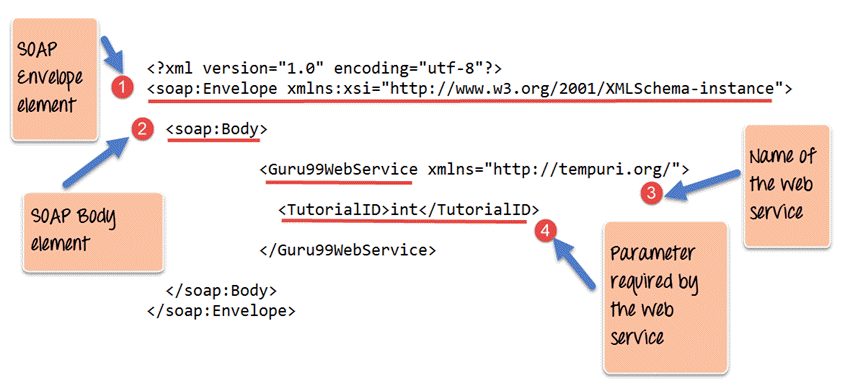
1. **XML – Extreme Mark Up language.**

**(HTML, Header, Body, div, fault)**

****

1. **SOAP – Simple Object Access Protocol**

**(Envelope, body, Parameters, Header, fault, div)**



1. **WSDL – Web Service Descriptive Language.**

**(Data type, Elements, message, binding)**

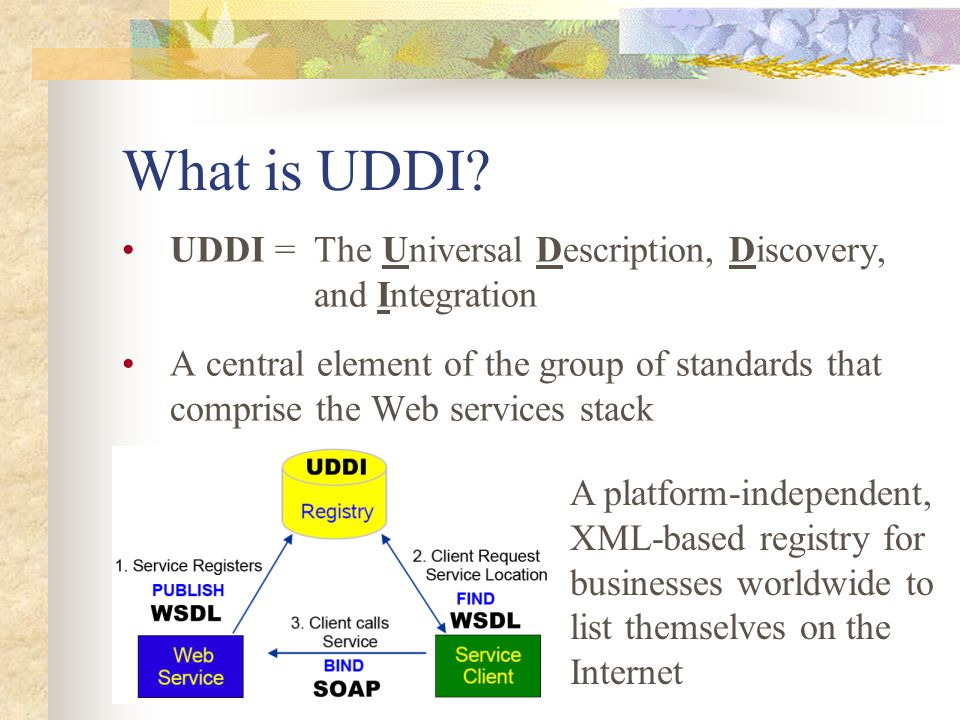
* WSDL is **an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information**

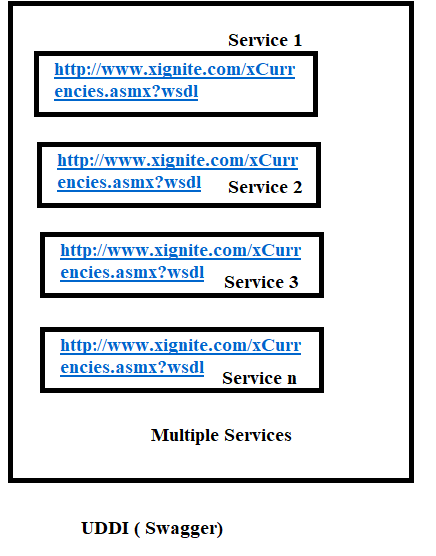
**Eg -** [**http://www.xignite.com/xCurrencies.asmx?wsdl**](http://www.xignite.com/xCurrencies.asmx?wsdl)

1. **UDDI – Universal Descriptive, Discovery and Integration**

**(WSDL file Repository)**

[**https://petstore.swagger.io/**](https://petstore.swagger.io/)



****

**----------------------------------------------------------------------------------------------------**

**SOAP**

**Envelope – SOAP Message (Either Request or Response)**

**Header – Authorization token, Access, i.e Username Password**

**Body – main parts of the SOAP message (E.g – When Search for the Travel then all the Travel display that names are present in body)**

**Fault – error message.**

**---------------------------------------------------------------------------------------------------**

**WSDL**

**Data Type – Type of Data which is used e.g int, float, string, etc**

**Elements – Operation perform on the data.**

**Message – Functionality of the Service**

**Binding – Combine all the functionality**

**Types of Services: (Used for API Testing)**

1. **SOAP Services**
2. **REST Services ( SOAP UI / POSTMAN)**

|  |  |
| --- | --- |
| **SOAP Services** | **REST Services** |
| 1. SOAP is Protocol | 1. REST is an Architecture (Special Design of Group of Protocols) |
| 1. **SOAP services are used for only Web Based Application.** | 1. **REST services are used for Web based, Mobile Based and Standalone application.** |
| 1. Language – Required only XML | 1. Language – HTTP, HTTPS, URL, URI, JSON |
| 1. Required – WSDL file for Testing | 1. Required – URI, URL, URN |
| 1. SOAP services are slow | 1. REST services are fast as compared to SOAP. |

--------------------------------------------------------------------------------------------------------------------

Common Test Cases for API Testing.

1. To Validate the Rest/SOAP Response
2. To Validate the Data and Count of Response
3. To validate the Tagname/Attribute present in the response
4. To validate the Status code in the response
5. To validate the Time taken for the Response.
6. To validate the Assertions applied for the Verification.
7. To validate the functionality by passing Test Data.
8. To Validate the Functionality by Negative Test Data

---------------------------------------------------------------------------------------------------------------------

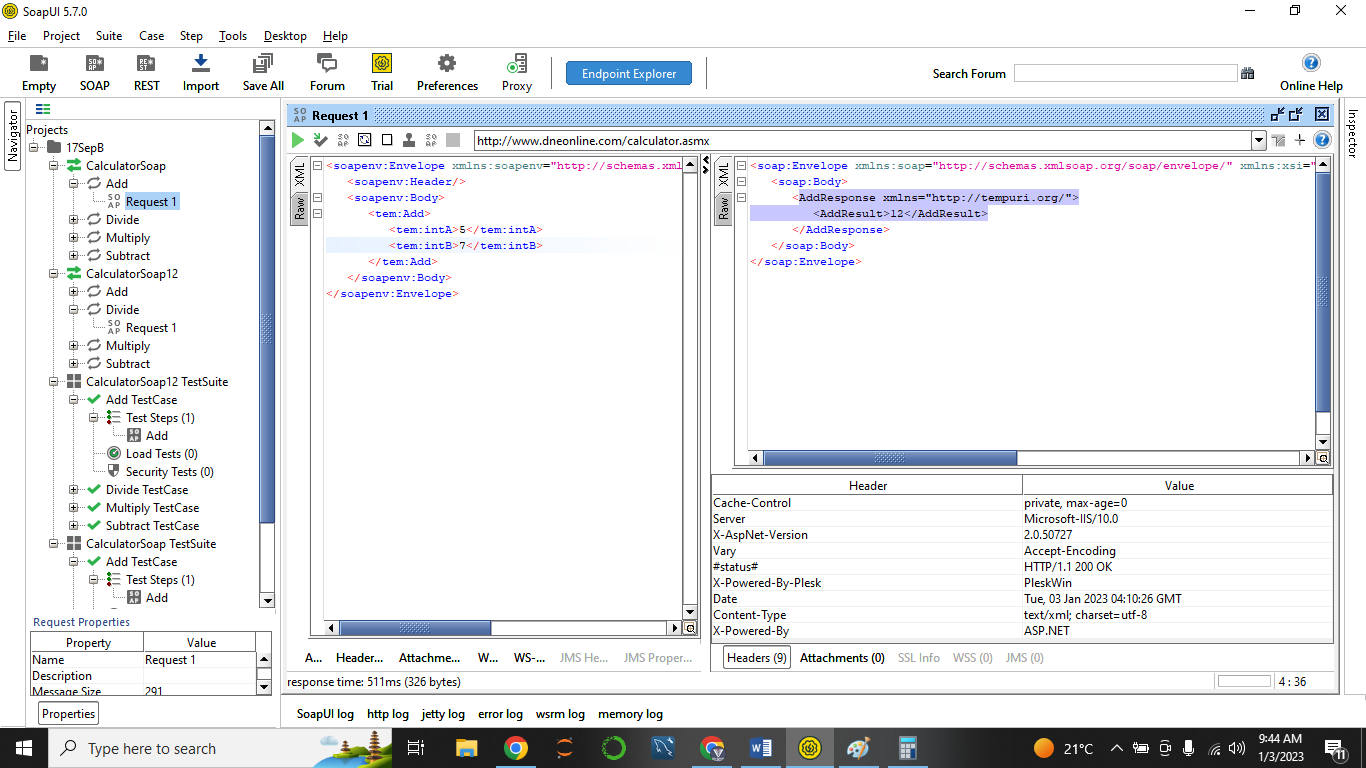
**SOAP UI Tool**

**For Testing SOAP Services.**

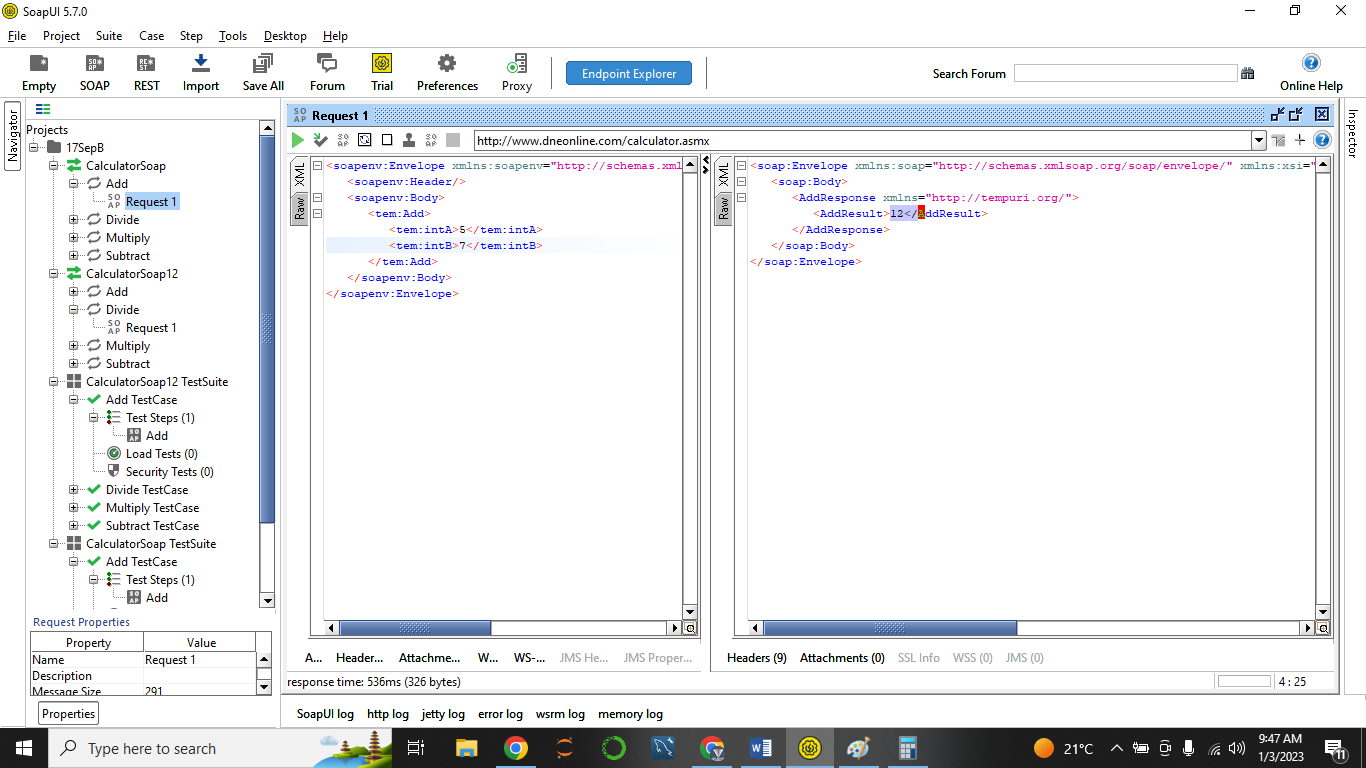
**Dev will provide**

* **SOAP Services – WSDL file :** [**http://www.dneonline.com/calculator.asmx?wsdl**](http://www.dneonline.com/calculator.asmx?wsdl)
* **Authorization Token, Auth Key, User Name Password**
* **Time for Response < 30 Sec.**
* **Unit Testing Document – Steps for Testing, URL, Table Name.**

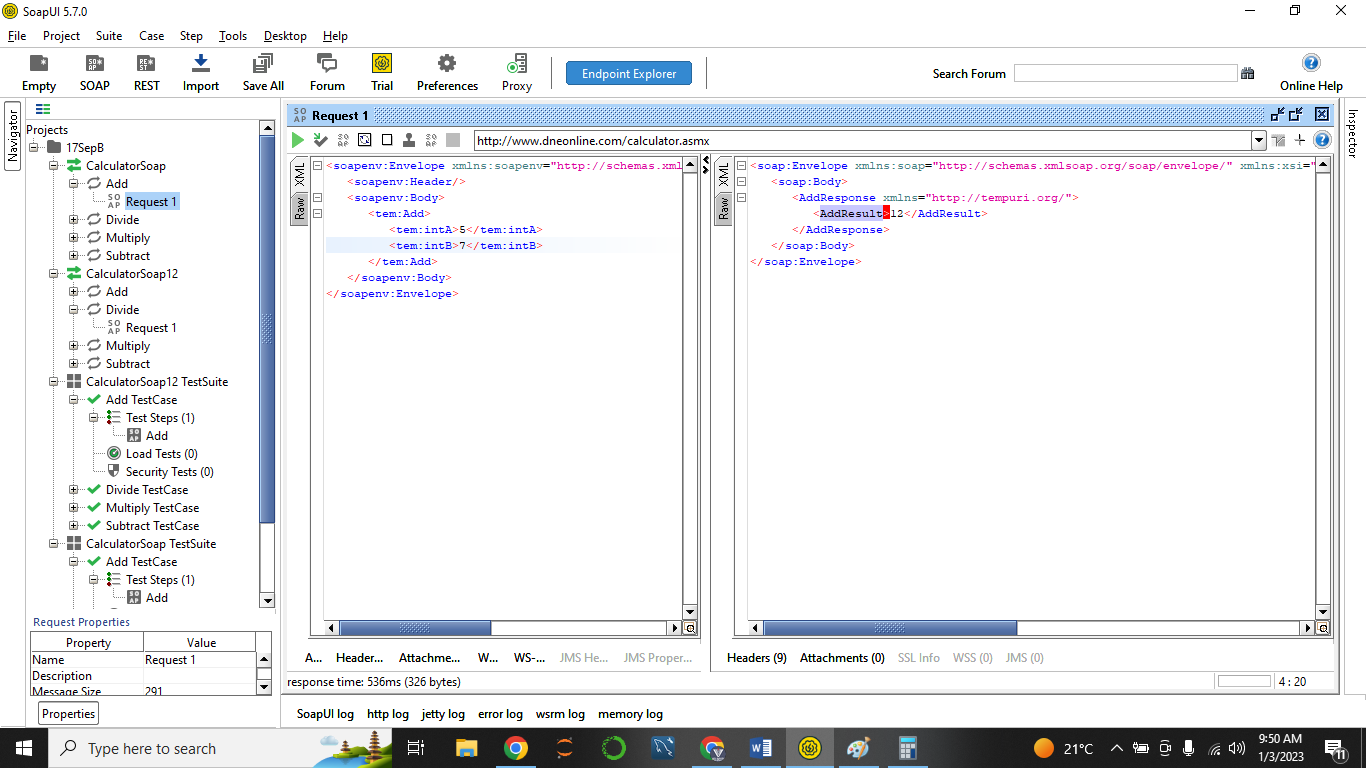
1. To Validate the Rest Response = PASS



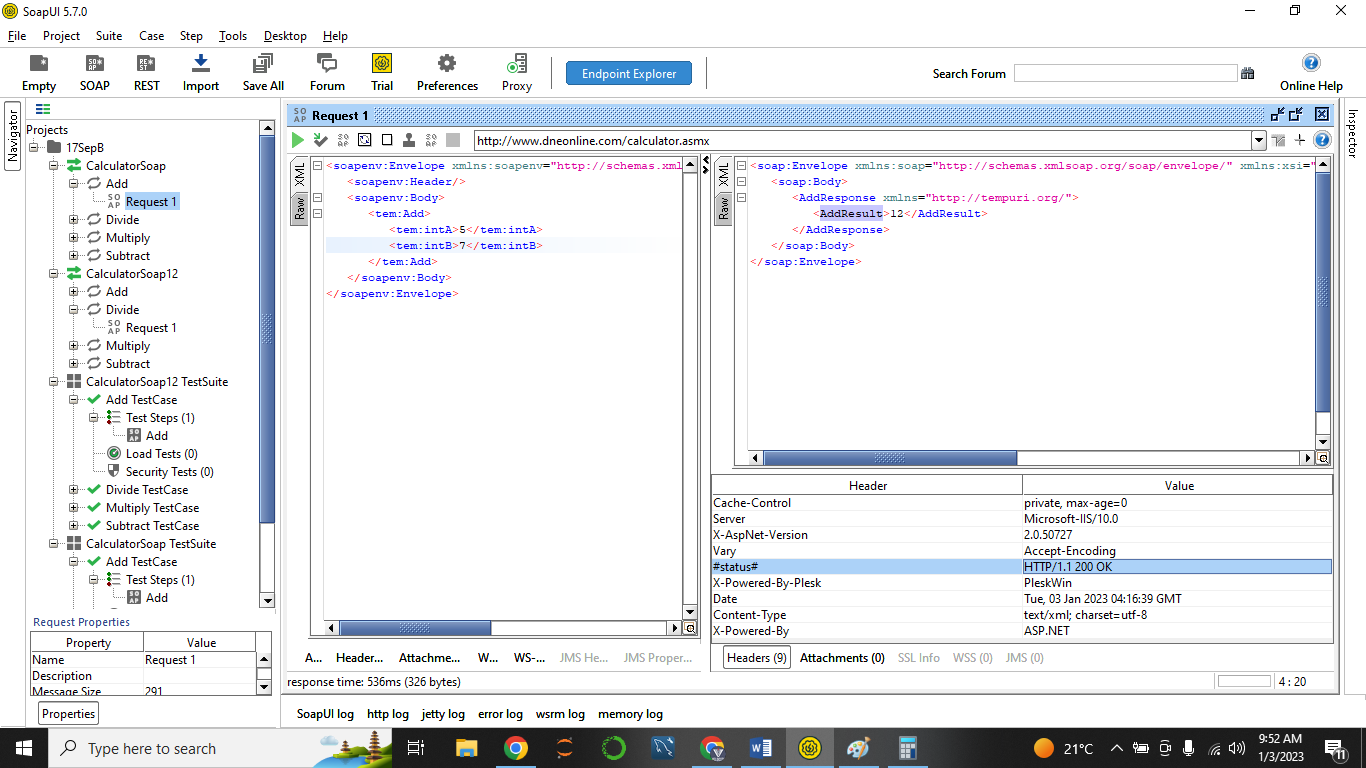
1. To Validate the Data and Count of Response = PASS



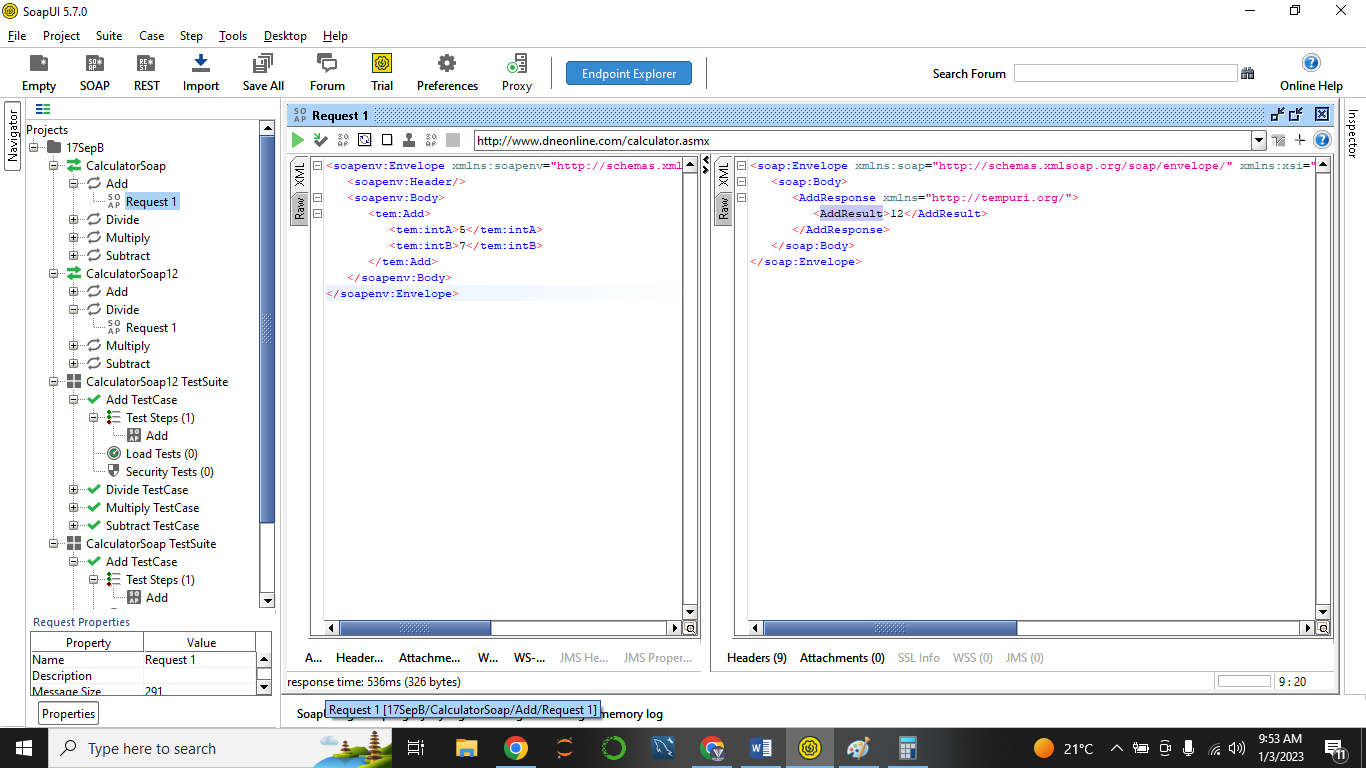
1. To validate the Tagname/Attribute present in the response = Pass (Tagname – **AddResult**)



1. To validate the Status code in the response = PASS (Status Code – 200 OK)

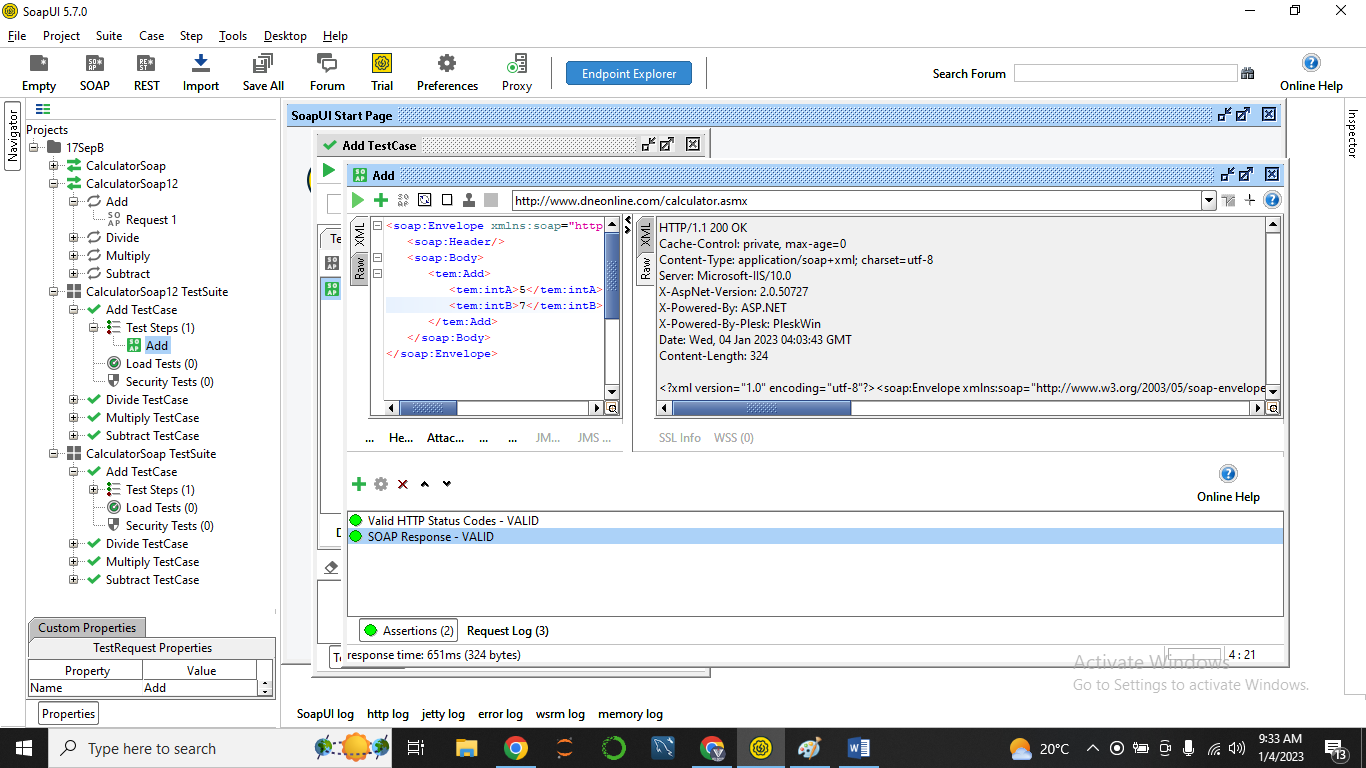


1. To validate the Time taken for the Response. = PASS ( Time Taken – 536 msec)



1. To validate the Assertions applied for the Verification. = PASS

( Status Code Assertion, Response)

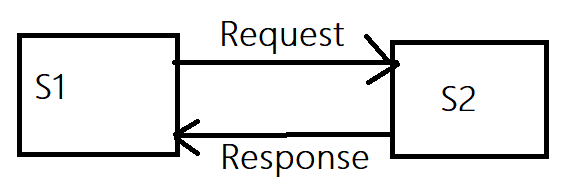


1. To validate the functionality by passing Test Data.

1 Digit, 2 Digit, 3 Digit, 4 Digit

-5 -10 0

1. To Validate the Functionality by Scenarios (Negative Test Data)

**REST SERVICES.**

**Types of Request Methods.**

1. GET: When S1 Request S2 🡪 to get the data of the S2.

E.g – Phone Pe (S1) 🡪 Bank Account HDFC (S2): View Balance -🡪 Respose - RS. 1200 (UI of Phone Pe).

(SQL – Parallel Command: Select)

1. POST : When S1 Request S2 🡪 to Create a data in service 2. (to create)

E.g – Facebook Profile Pic Upload/Post.

(SQL – Parallel Command: Insert / Create)

1. PATCH : When S1 Request S2 🡪 to update the Data of S2 (Which already stored)  
   (SQL – Parallel Command: Update)
2. PUT : When S1 Request S2 🡪 to update the Data of S2 (Which already stored)

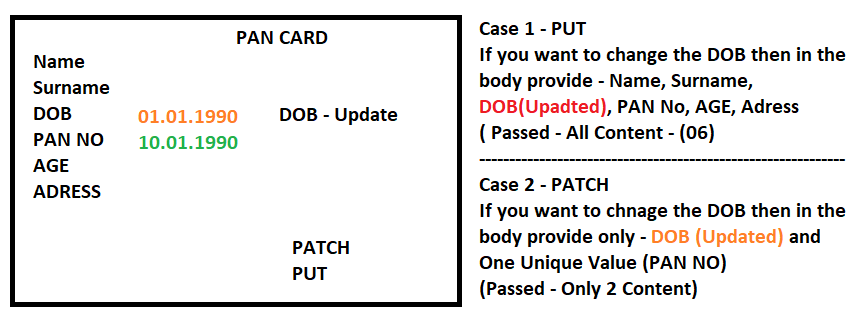
(SQL – Parallel Command: Update)

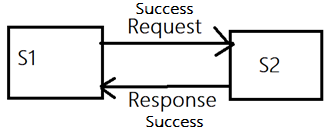
1. DELETE

When S1 Request S2 🡪 Delete the record / data from s2.

(SQL – Parallel Command: DELETE)

(PATCH method is Fast one)

****

**Status Code**

**Developer Defines the Status Code:**

1. **Successful Response – 2XX (200, 201, 202, 204)**
2. **Server Error – 4XX (400, 401, 403, 404, 405)**
3. **Internal Server Error - 5XX (500, 501, 503)**
4. **Successful Response – 2XX (200, 201, 202, 204)**
5. **200: OK – GET: Request Successful and GET method is used.**
6. **201: Created – POST: Request Successful and POST method is used.**
7. **202: Accepted – PATCH/PUT: Request Successful and PATCH/PUT method is used.**
8. **204: No Content - DELETE: Request Successful and DELETE method is used.**
9. **Server Error**

**S1 send request to S2 but there is Issue at S2**

1. **400: Bad Request : When S1 request S2 🡪 S2 not responded correctly**
2. **401: Unauthorized: When S1 request S2 🡪 S1 has not access to use any method/Wrong method used/ Access token is wrong.**
3. **403 : Forbidden :**
4. **404 : Not Found :**
5. **405 : Method Not Allowed :**

