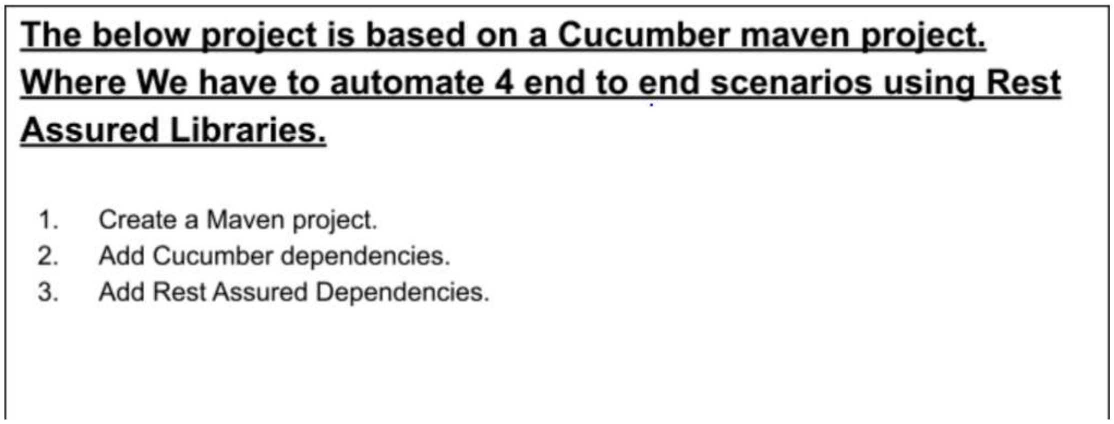
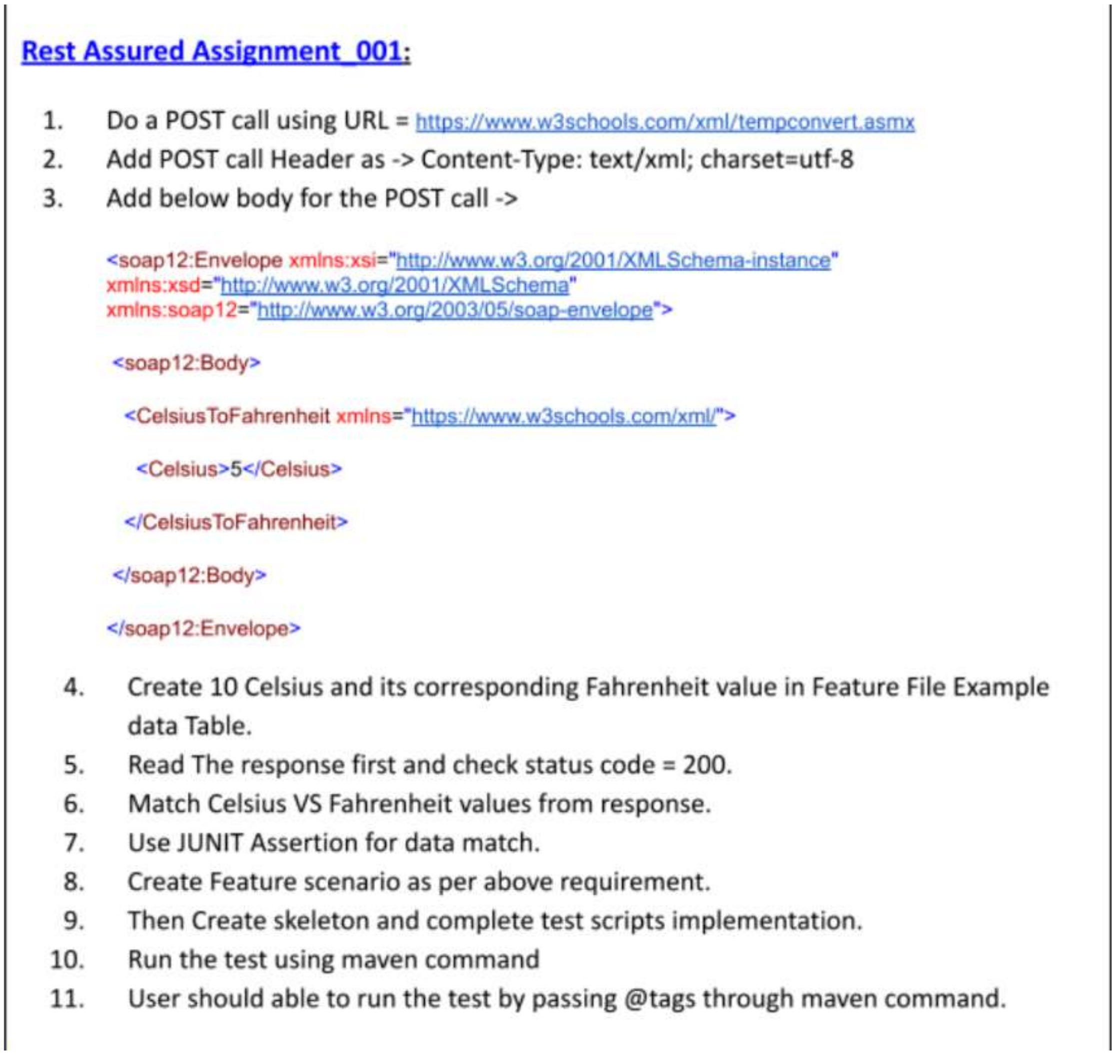
Non-Functional Testing Using Cucumber and Rest Assured



The below project is based on a Cucumber maven project. Where We have to automate 4 end to end scenarios using Rest Assured Libraries.

1. Create a Maven project.
2. Add Cucumber dependencies.
3. Add Rest Assured Dependencies.



Rest Assured Assignment_001:

1. Do a POST call using URL = https://www.w3schools.com/xml/tempconvert.asmx
2. Add POST call Header as -> Content-Type: text/xml; charset=utf-8
3. Add below body for the POST call ->

```
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

 <soap12:Body>

  <CelsiusToFahrenheit xmlns="https://www.w3schools.com/xml/">

   <Celsius>5</Celsius>

  </CelsiusToFahrenheit>

 </soap12:Body>

</soap12:Envelope>
```

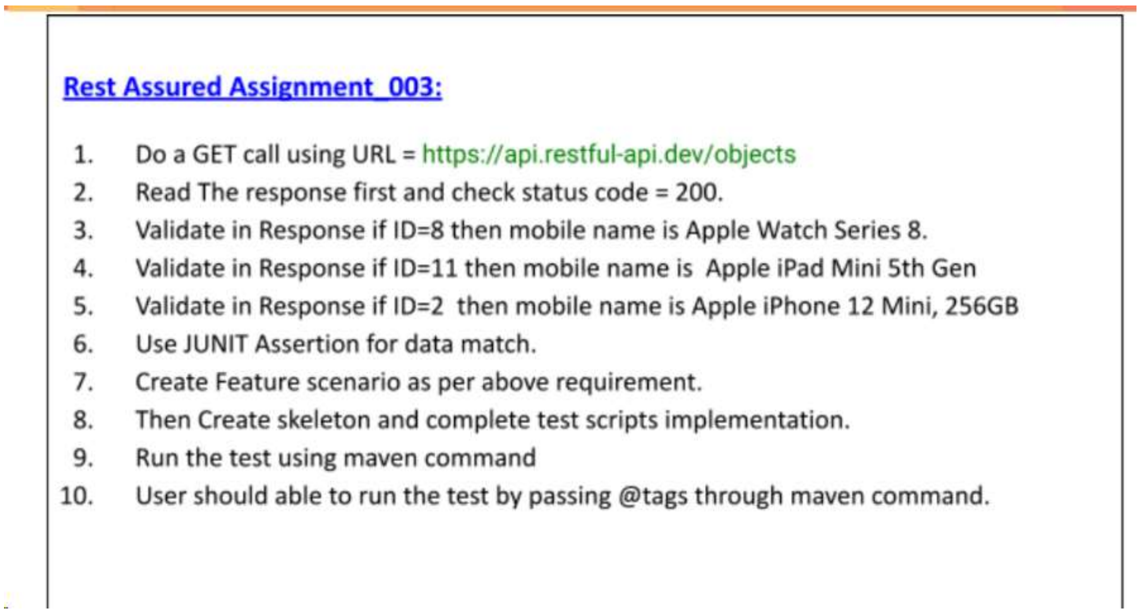
4. Create 10 Celsius and its corresponding Fahrenheit value in Feature File Example data Table.
5. Read The response first and check status code = 200.
6. Match Celsius VS Fahrenheit values from response.
7. Use JUNIT Assertion for data match.
8. Create Feature scenario as per above requirement.
9. Then Create skeleton and complete test scripts implementation.
10. Run the test using maven command
11. User should able to run the test by passing @tags through maven command.

Rest Assured Assignment_002:

1. Do a POST call using URL = https://api.restful-api.dev/objects
2. Add below body for the POST call ->

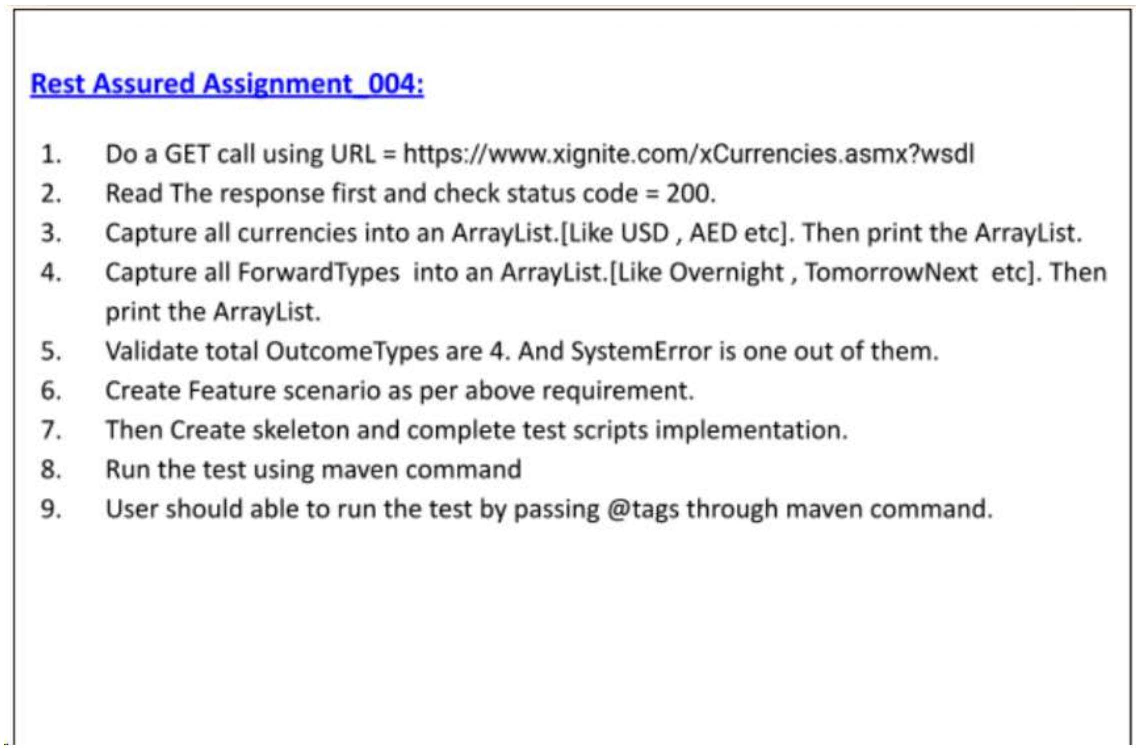
```
{
   "name": "Apple MacBook Pro 16",
   "data": {
      "year": 2019,
      "price": 1849.99,
      "CPU model": "Intel Core i9",
      "Hard disk size": "1 TB"
   }
}
```

3. Read name = 'Apple MacBook Pro 16' from example Table.
4. Read The response first and check status code = 200.
5. Make YEAR as Example Table data driven and check in Response Year is same.
6. Make price as Example Table data driven and check in Response Price is same.
7. Validate in response createdAt tag and its value should not be NULL.
8. Use JUNIT Assertion for data match.
9. Create Feature scenario as per above requirement.
10. Then Create skeleton and complete test scripts implementation.
11. Run the test using maven command
12. User should able to run the test by passing @tags through maven command.



Rest Assured Assignment_003:

1. Do a GET call using URL = https://api.restful-api.dev/objects
2. Read The response first and check status code = 200.
3. Validate in Response if ID=8 then mobile name is Apple Watch Series 8.
4. Validate in Response if ID=11 then mobile name is Apple iPad Mini 5th Gen
5. Validate in Response if ID=2 then mobile name is Apple iPhone 12 Mini, 256GB
6. Use JUNIT Assertion for data match.
7. Create Feature scenario as per above requirement.
8. Then Create skeleton and complete test scripts implementation.
9. Run the test using maven command
10. User should able to run the test by passing @tags through maven command.



Rest Assured Assignment_004:

1. Do a GET call using URL = https://www.xignite.com/xCurrencies.asmx?wsdl
2. Read The response first and check status code = 200.
3. Capture all currencies into an ArrayList.[Like USD , AED etc]. Then print the ArrayList.
4. Capture all ForwardTypes into an ArrayList.[Like Overnight , TomorrowNext etc]. Then print the ArrayList.
5. Validate total OutcomeTypes are 4. And SystemError is one out of them.
6. Create Feature scenario as per above requirement.
7. Then Create skeleton and complete test scripts implementation.
8. Run the test using maven command
9. User should able to run the test by passing @tags through maven command.