Practical Exercise: Invoking the API

Training Objective

Learn how to test the API in Developer Portal and build and deploy the web application and test the application using cURL.

Business Scenario

After subscribing to the API the partners can access the API through the web application which leverages the WSO2 API Manager token API to generate JWT access tokens on demand.

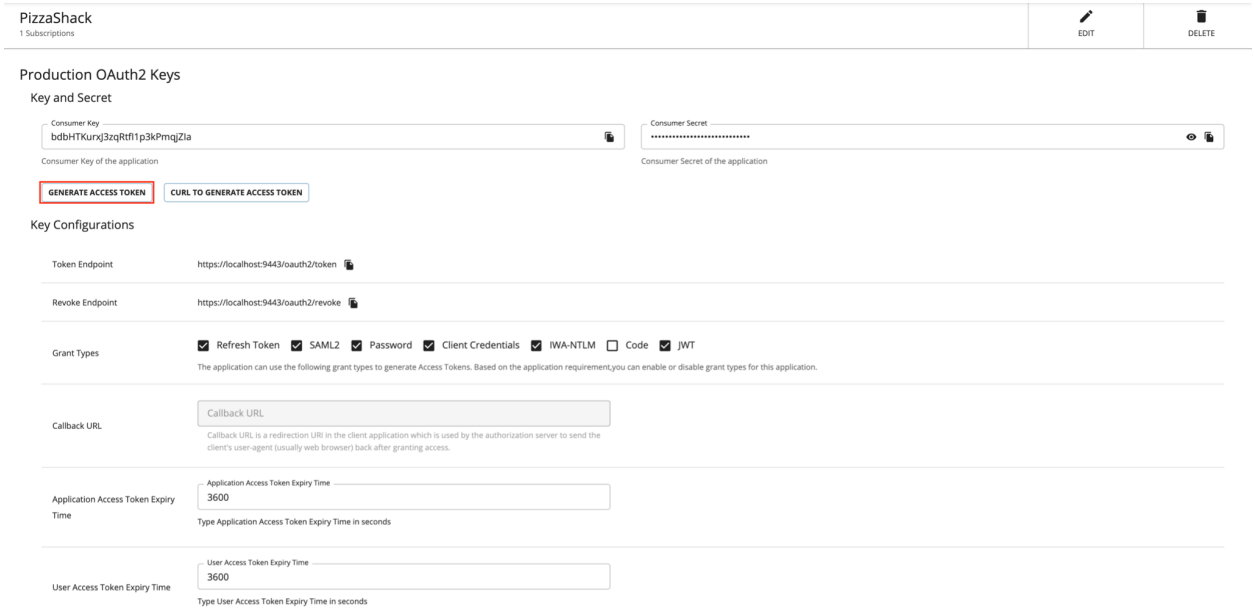
High-Level Steps

- Test the API using Developer Portal
- Test the API using cURL
- Deploy and test the PizzaShack Application

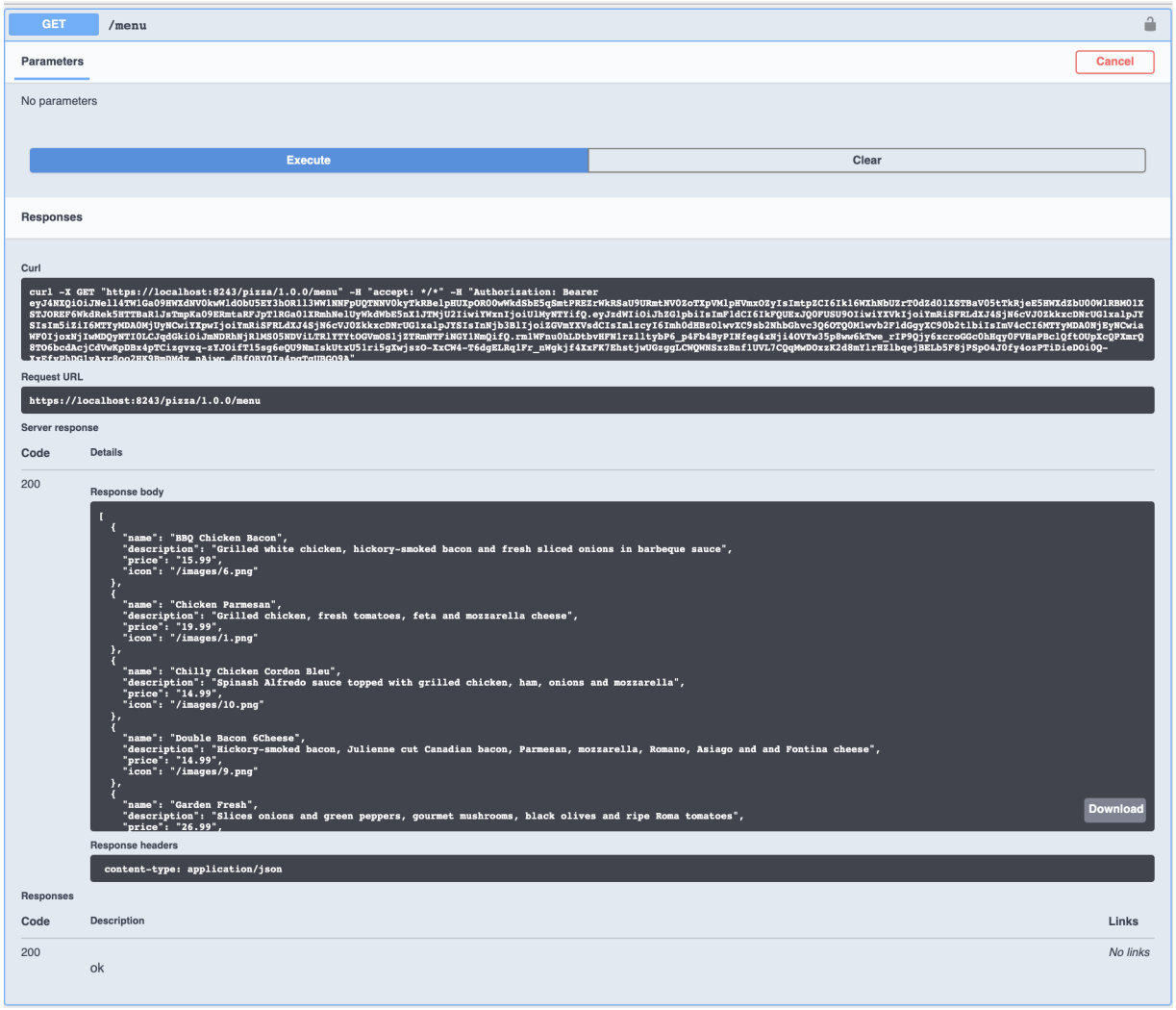
Detailed Instructions

Test the PizzaShack Application from the Developer Portal

1. Login to the Developer Portal as the **subscriber** user.
2. Click on the Pizzashack API.
3. Subscribe to the Pizzashack API by clicking on the **Subscriptions** tab of the API.
4. Once the subscription is added to the Subscriptions table, go to **Applications** at the Top Menu > Click on the **PizzaShack** Application > Then click on the **Production Keys** > **OAuth2 Tokens** tab from the Left Menu and click on **Generate Keys.**
5. Once the Keys are successfully generated, click on the **Generate Access Token** button and select the necessary scopes (order_pizza) and click on **GENERATE** and copy the generated Access Token.



1. Go back to the API > Click on the **Try Out** tab.
2. Paste the copied access token in the access token text box or click on **GET TEST KEY**. Make sure that the PizzaShack application is selected from the Applications drop down.
3. Expand the GET /menu resource and click on **Try it out** > **Execute**. You will now be able to see the response of the GET /menu resource.



Test the API using cURL

To test the API using cURL, we need to pass the correct API key or the Access Token. The API Key must be passed inside an Authorization HTTP Header:

e.g.,
`Authorization: Bearer vMxNW6ILwNrWvnKJyewejSlHZFka`
It's very simple to use cURL - Let's exercise the Menu API.
Note: You can also use a REST API Client of your preference instead of cURL.

1. Open a new Command Line Interface
2. Type `curl -v http://localhost:8280/pizzashack/1.0.0/menu`

OR

Method: GET

URL: `http://localhost:8280/pizzashack/1.0.0/menu`

You will see the following message if the cURL command is used.

```
* Connected to localhost (127.0.0.1) port 8280 (#0)
> GET /pizzashack/1.0.0/menu HTTP/1.1
> Host: localhost:8280
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< activityid: ed7e2cea-8eac-4fef-ba8b-6da508e4c4ac
< Access-Control-Expose-Headers:
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET
< WWW-Authenticate: null Bearer realm="WSO2 API Manager", error="invalid_token", error_description="The
provided token is invalid"
< Access-Control-Allow-Headers:
authorization,Access-Control-Allow-Origin,Content-Type,SOAPAction,apikey,Internal-Key,Authorization
< Content-Type: application/json; charset=UTF-8
< Date: Mon, 03 May 2021 11:51:26 GMT
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
{"code":"900902","message":"Missing Credentials","description":"Invalid Credentials. Make sure your API
invocation call has a header: 'Authorization : Bearer ACCESS_TOKEN' or 'Authorization : Basic ACCESS_TOKEN'
or 'apikey: API_KEY'"}* Closing connection 0
```

3. Now use the **Access Token** you previously generated and add it as the Authorization Bearer. (Make sure you generate the access token with **order_pizza** scope)

   ```
   curl -H "Authorization: Bearer XXXXXXXX" -v http://localhost:8280/pizzashack/1.0.0/menu
   ```

   where XXXXXXXX is the access token generated through the application. You should get a response similar to the one below:

   ```
   *   Trying ::1...
   * TCP_NODELAY set
   * Connection failed
   * connect to ::1 port 8243 failed: Connection refused
   *   Trying 127.0.0.1...
   * TCP_NODELAY set
   * Connected to localhost (127.0.0.1) port 8243 (#0)
   * ALPN, offering h2
   * ALPN, offering http/1.1
   ```

```
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: C=US; ST=CA; L=Mountain View; O=WSO2; OU=WSO2; CN=localhost
*  start date: Oct 23 07:30:43 2019 GMT
*  expire date: Jan 25 07:30:43 2022 GMT
*  issuer: C=US; ST=CA; L=Mountain View; O=WSO2; OU=WSO2; CN=localhost
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing anyway.
> GET /pizza/1.0.0/menu HTTP/1.1
> Host: localhost:8243
> User-Agent: curl/7.64.1
> accept: */*
> Authorization: Bearer
eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWk
dSbE5qSmtPREZrWkRSaU9URmtNV0ZoTXpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRj
eE5HWXdZbU00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelUyWkdWb
E5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJhZG1pbiIsImF1dCI6IkFQUExJQ0FUSU9OIiwiYXVkIjoiYmRiS
FRLdXJ4SjN6cVJ0ZkkxcDNrUG1xalpJYSIsIm5iZiI6MTYyMDA0MjUyNCwiYXpwIjoiYmRiSFRLdXJ4SjN6cVJ0Zkkx
cDNrUG1xalpJYSIsInNjb3BlIjoiZGVmYXVsdCIsImlzcyI6Imh0dHBzOlwvXC9sb2NhbGhvc3Q6OTQ0M1wvb2F1dG
gyXC90b2tlbiIsImV4cCI6MTYyMDA0NjEyNCwiaWF0IjoxNjIwMDQyNTI0LCJqdGkiOiJmNDRhNjRlMS05NDViLTRl
YTYtOGVmOS1jZTRmNTFiNGY1NmQifQ.rm1WFnu0hLDtbvHFN1rzlltybP6_p4Fb4ByPINfeg4xNji4OVYw35p8ww
6kTwe_rIP9Qjy6xcroGGc0hHqy0FVHaPBclQftOUpXcQPXmrQ8TO6bcdAcjCdVwKpDBx4pTCizgvxq-zYJOifT15sg
6eQU9NmIskUtxU51ri5gXwjszO-XxCW4-T6dgELRq1Fr_nWgkjf4XxFK7EhstjwUGzggLCWQWNSxzBnf1UVL7CQ
qMwDOxzK2d8mYlrHZlbqejBELb5F8jPSpO4J0fy4ozPTiDieDOi0Q-XxEfyPhDG1yAxr8oo2HK9BmDMdv_nAiwc_d
BfOBY0Ia4pqTgUBGO9A
>
< HTTP/1.1 200
< activityid: 217ddb00-7e4f-4955-a686-c1e4561f9008
< Access-Control-Expose-Headers:
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET
< Access-Control-Allow-Headers:
authorization,Access-Control-Allow-Origin,Content-Type,SOAPAction,apikey,Internal-Key,Authorization
< Content-Type: application/json
< Date: Mon, 03 May 2021 11:55:43 GMT
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
```

[{"name":"BBQ Chicken Bacon","description":"Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbeque sauce","price":"15.99","icon":"/images/6.png"},{"name":"Chicken Parmesan","description":"Grilled chicken, fresh tomatoes, feta and mozzarella cheese","price":"19.99","icon":"/images/1.png"},{"name":"Chilly Chicken Cordon Bleu","description":"Spinash Alfredo sauce topped with grilled chicken, ham, onions and mozzarella","price":"14.99","icon":"/images/10.png"},{"name":"Double Bacon 6Cheese","description":"Hickory-smoked bacon, Julienne cut Canadian bacon, Parmesan, mozzarella, Romano, Asiago and and Fontina cheese","price":"14.99","icon":"/images/9.png"},{"name":"Garden Fresh","description":"Slices onions and green peppers, gourmet mushrooms, black olives and ripe Roma tomatoes","price":"26.99","icon":"/images/3.png"},{"name":"Grilled Chicken Club","description":"Grilled white chicken, hickory-smoked bacon and fresh sliced onions topped with Roma tomatoes","price":"27.99","icon":"/images/8.png"},{"name":"Hawaiian BBQ Chicken","description":"Grilled white chicken, hickory-smoked bacon, barbeque sauce topped with sweet pine-apple","price":"27.99","icon":"/images/7.png"},{"name":"Spicy Italian","description":"Pepperoni and a double portion of spicy Italian sausage","price":"20.99","icon":"/images/2.png"},{"name":"Spinach Alfredo","description":"Rich and creamy blend of spinach and garlic Parmesan with Alfredo sauce","price":"16.99","icon":"/images/5.png"},{"name":"Tuscan Six Cheese","description":"Six cheese blend of mozzarella, Parmesan, Romano, Asiago and Fontina","price":"15.99","icon":"/images/4.png"}]

Deploy and Test the PizzaShack Application

To test the application, do the following.

Note: Make sure the **pizzashack.war** file is already deployed under <APIM_HOME>/repository/deployment/server/webapps directory before starting.

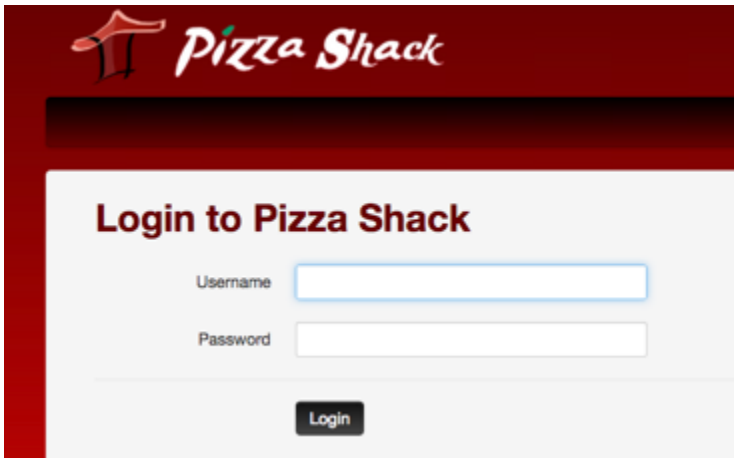
1. Download the **pizzashack.war** file from the link and copy it to <APIM_HOME>/repository/deployment/server/webapps so that it is deployed.
2. The service will be deployed after a few seconds.

To edit the file and build the app,

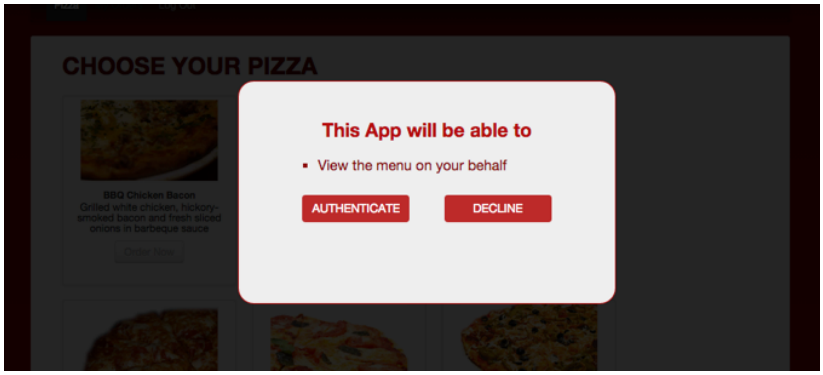
3. Open <APIM_HOME>/repository/deployment/server/webapps/pizzashack/WEB-INF/web.xml.
4. Update the consumer key and client secret with the values obtained when generating keys above for the PizzaShack application and save the file.

```
</context-param>
<context-param>
        <param-name>consumerKey</param-name>
        <param-value>XXXXXXXXXXXXXXXXXXXXXXXXXXXX</param-value>
</context-param>
<context-param>
        <param-name>consumerSecret</param-name>
        <param-value>YYYYYYYYYYYYYYYYYYYYYYYYYYYY</param-value>
 </context-param>
```

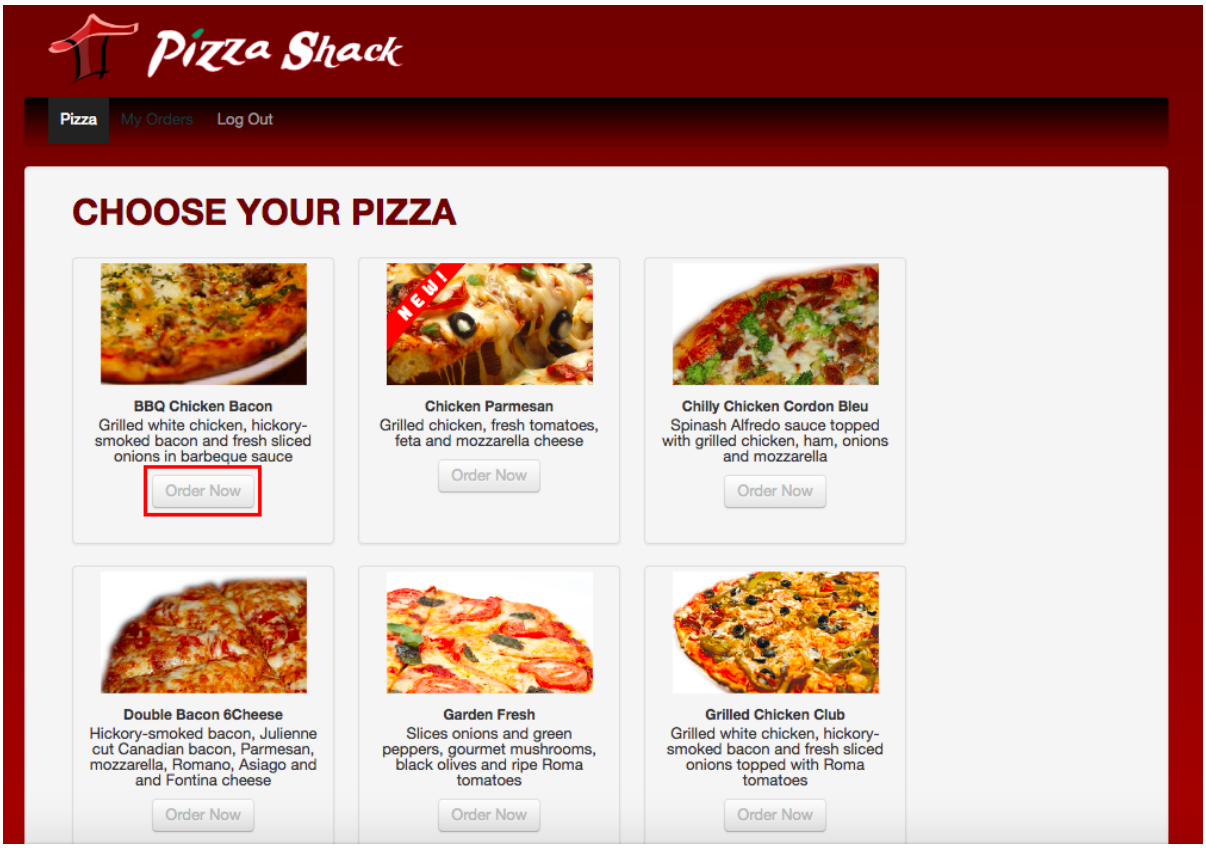
5. Go to: https://localhost:9443/pizzashack/login.jsp and log in as user **mike**.



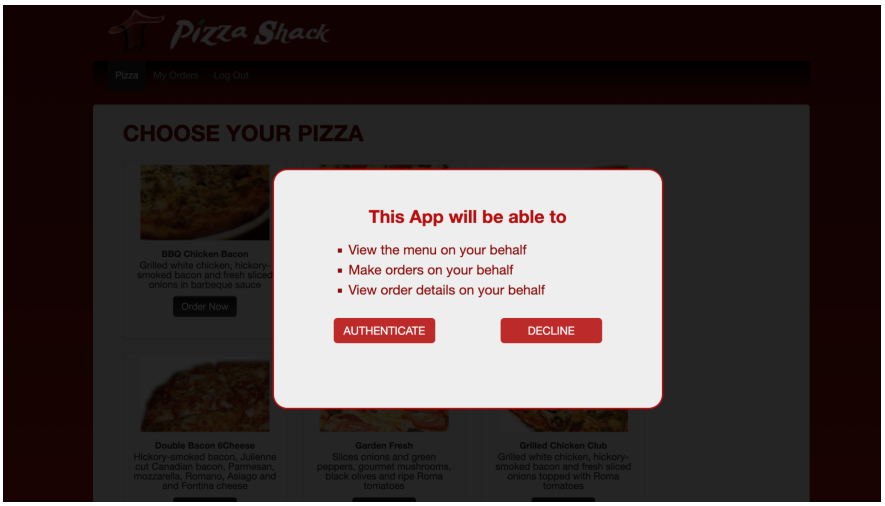
When mike logs in, he will not be able to get a token having the **order_pizza** scope since he doesn't have the **webuser** role. As a result, you will see the screen below.



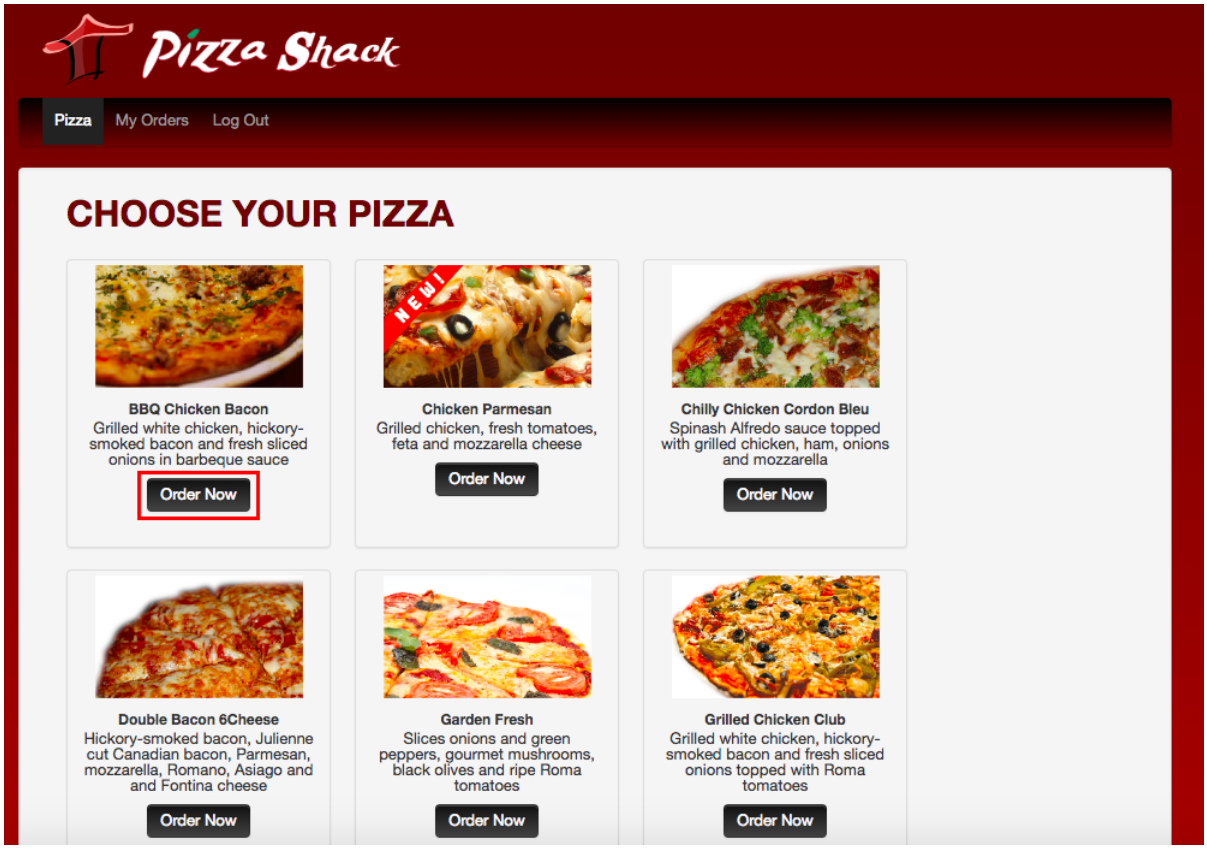
Note that the **Order Now** button is disabled in the image below.



6. Log in as a subscriber user. Since the user ***subscriber*** has the **Internal/subscriber** role, he is capable of getting an access token which has the **order_pizza** scope and can invoke the /order resource of the PizzaShackAPI. When you log in as a subscriber, you will see the screen below.



Note that the **Order Now** button is enabled.



Expected Outcome

In this exercise, the API was tested using Developer Portal and cURL and the PizzaShack web application was built and deployed in WSO2 API Manager. The web application was tested using 2 users with different levels of permission.