Technical Operation Guide

# **Introduction**

This document is to provide a thorough technical overview of the solution POC.

A client wants to build an API that allows users to submit reviews on their products. The client wants the users to submit their comments and a star rating in the review

A list of the following solutions:

* Web API to List all products
* Web API to Create a new product review
* Web API to Update product review
* Web API to Delete a product review
* Web API for List all reviews for a product base on title Get /products**/{productTitle}**/reviews

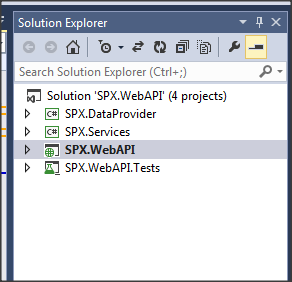
Bonus Part:

* Create unit tests for the API.
* Authentication and Security: Allow only authorized users to delete a review

Project Structure and design:

Project build with Layered architecture and Entity Framework Code First Model approach used for data access layer along with Generic Repository design pattern and UnitOfWork.

Web API’s are built using Web API 2 in Visual Studio 2013 and below is the screen grab of Layered Architecture project along with Unit Test project. Project Source is uploaded to GIT



# **WEB API DETAILS:**

1. List all products : URL(localhost): <http://localhost:54028/api/SPX>

Method Type : GET

Sample JSON Response looks like:

[{

"ProductId": 1,

"Title": "Titel1",

"ShortDescription": "10",

"Brand": "40",

"Reviews": [{

"Rating": "12 34 56",

"Comments": "8327 2837 4732 1234T",

"User": "12 34 56"

}]

}, {

"ProductId": 2,

"Title": "Titel1B",

"ShortDescription": "10B",

"Brand": "40B",

"Reviews": [{

"Rating": "12 34 56",

"Comments": "8327 2837 4732 1234",

"User": "12 34 56"

}, {

"Rating": "3",

"Comments": "8327 2837 4732 1234",

"User": "12 34 56"

}]

}]

1. Create a new product review: URL(localhost): <http://localhost:54028/api/spx/InsertProducts>

Method: POST

Sample JSON Request Payload:

[{

"ProductId": 1,

"Title": "Title1",

"ShortDescription": "Good to go ahead",

"Brand": "APV",

"Reviews": [{

"Rating": "3",

"Comments": "Looking forward",

"User": "HarshaRavi"

},

{

"Rating": "2",

"Comments": "Not recommended",

"User": "Shashi"

}

]

},

{

"ProductId": 2,

"Title": "Title Airpel",

"ShortDescription": "10B",

"Brand": "Airpel",

"Reviews": [{

"Rating": "2",

"Comments": "NOT GOOD",

"User": "Rai"

},

{

"Rating": "3",

"Comments": "Average",

"User": "sonwoh"

}

]

}]

Good Response Samples:

{

"Status": "SUCCESS",

"Response": "REQUEST PROCESS SUCCESSFULLY AND PRODUCT INSERTED"

}

Bad Response (Payload Validation) Samples:

Example 1:

{

"Status": "ERROR",

"Response": "REQUEST CONTENT IS NULL, JSON PAYLOAD IS MOST LIKELY IN BAD FORMAT."

}

Example 2:

{

"Status": "ERROR",

"Response": "BRAND TYPE IS INVALID IN REQUEST PAYLOAD."

}

Example 3:

{

"Status": "ERROR",

"Response": "RATING VALUE SHOULD BE BETWEEN 1 AND 5 IN REQUEST PAYLOAD."

}

1. Web API to Update product review

URL (localhost): [http://localhost:54028/api/SPX/{**ReviewId**}](http://localhost:54028/api/SPX/%7bReviewId%7d)

METHOD: PUT

Sample JSON Request Payload:

{

"Rating": "1",

"Comments": "8327 2837 4732 1234T",

"User": "12 34 56"

}

Sample JSON Good Response:

{

"Status": "SUCCESS",

"Response": "UPADATE REQUEST IS SUCCESS"

}

Sample JSON Bad Response:

{

"Status": "NOT FOUND",

"Response": "NO REVIEW TO UPADATE"

}

Example 2:

{

"Status": "ERROR",

"Response": "REQUEST FAILED"

}

1. Web API to Delete a product review

URL (localhost): [http://localhost:54028/api/SPX/{**ReviewId**}](http://localhost:54028/api/SPX/%7bReviewId%7d)

METHOD: DELETE

Sample JSON Request Payload:

Raw Headers: (Authorization : Base64 encode username (SPX) and pwd (Password)

Content-Type: application/json

Authorization: Basic U1BYOlBhc3N3b3Jk

Sample JSON Request Payload:

{

"Status": "SUCCESS",

"Response": "REVIEW DELETED SUCCESSFULLY"

}

Sample JSON Unauthorized Response

{

"Status": "ACCESS\_DENIED",

"Response": "UNAUTHORIZED ACCESS TO THE API"

}

Ex 2:

{

"Status": "NOT FOUND",

"Response": "REVIEW NOT FOUND TO DELETE"

}

1. Web API for List all reviews for a product base on title Get /products**/{productTitle}**/reviews

NOTE: **{productTitle}** is LIKE SEARCH

URL (localhost): http://localhost:54028/api/SPX/Get/products/**{ProductTitle}**/reviews

METHOD: GET

Sample JSON Response:

[{

"Rating": "3",

"Comments": "Good one!",

"User": "Sri"

},

{

"Rating": "4",

"Comments": "Nice one!",

"User": "HarshaRavi"

},

{

"Rating": "4",

"Comments": "Good!",

"User": "SonWoh"

}]

Exception handing is done by simple try catch with custom response for invalid request and all logs including request and response with URL (API Details) are done using Log4Net Nuget package.

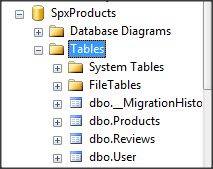
All logs are done using Log file Appender (datetime formatted.txt) in application root folder “Logs”.

Sample logs attached.



# **Database Details:**

1. Database Script is shared in GIT and below are the tables used



1. User Table should contain predefined value which is used to authentication. Below is the screen shot table “**User**”



NOTE: UI in ASP.NET using AJAX to call the REST API NOT covered due to less chance to work.