**Developer Details:-**

Name:- Nilesh Rajak

Date:- 09th May 2021

**Project Details:-**

Name:- ShopBridge

Tool:- Visual Studio 2019

Architecture: - Web API

DB:- SQL server 2012

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

1:- Creation of database with table and store procedure

--Create Database ShopBridgeDB

--use ShopBridgeDB

--Create Table tblShopBridgeDetails

--(

-- Id int identity(1,1) primary key,

-- Name Varchar(50) Not null,

-- Description Varchar(500) Not null,

-- Price decimal(10, 2) Not Null,

-- IsActive varchar(10) Not Null,

-- CreatedDate datetime Not Null,

-- ModifiedDate datetime

--)

--Select \*from tblShopBridgeDetails

--ALTER PROCEDURE SP\_ShopBridge @Task VARCHAR(50) = NULL

-- ,@Id INT = NULL

-- ,@Name VARCHAR(50) = NULL

-- ,@Description VARCHAR(500) = NULL

-- ,@Price DECIMAL(10, 2) = NULL

--AS

--BEGIN

-- IF (@Task = 'Get')

-- BEGIN

-- SELECT4 Id

-- ,Name

-- ,Description

-- ,Price

-- FROM tblShopBridgeDetails

-- WHERE IsActive = 'Y'

-- END

-- ELSE IF (@Task = 'GetById')

-- BEGIN

-- SELECT Id

-- ,Name

-- ,Description

-- ,Price

-- FROM tblShopBridgeDetails

-- WHERE Id = @Id

-- AND IsActive = 'Y'

-- END

-- ELSE IF (@Task = 'Insert')

-- BEGIN

-- IF EXISTS (

-- SELECT \*

-- FROM tblShopBridgeDetails

-- WHERE Name = @Name

-- AND IsActive = 'Y'

-- )

-- BEGIN

-- BEGIN TRAN

-- BEGIN TRY

-- INSERT INTO tblShopBridgeDetails

-- VALUES (

-- @Name

-- ,@Description

-- ,@Price

-- ,'Y'

-- ,GETDATE()

-- ,NULL

-- )

-- COMMIT

-- SELECT 1;--Record inserted seccessfully

-- END TRY

-- BEGIN CATCH

-- ROLLBACK

-- SELECT 0;--Record Not Inserted due to some error

-- END CATCH

-- END

-- ELSE

-- BEGIN

-- SELECT 2;--Duplicate records can not be allow to insert

-- END

-- END

-- ELSE IF (@Task = 'Update')

-- BEGIN

-- IF EXISTS (

-- SELECT \*

-- FROM tblShopBridgeDetails

-- WHERE Id = @Id

-- )

-- BEGIN

-- BEGIN TRAN

-- BEGIN TRY

-- UPDATE tblShopBridgeDetails

-- SET Name = @Name

-- ,Description = @Description

-- ,Price = @Price

-- ,ModifiedDate = getdate()

-- WHERE Id = @Id

-- COMMIT;

-- SELECT 1;--Record is updated

-- END TRY

-- BEGIN CATCH

-- ROLLBACK;

-- SELECT 0;--Record is not update

-- END CATCH

-- END

-- ELSE

-- BEGIN

-- SELECT 2;---Record is not available for update

-- END

-- END

-- ELSE IF (@Task = 'Delete')

-- BEGIN

-- IF EXISTS (

-- SELECT \*

-- FROM tblShopBridgeDetails

-- WHERE Id = @Id And IsActive='Y'

-- )

-- BEGIN

-- BEGIN TRAN

-- BEGIN TRY

-- UPDATE tblShopBridgeDetails

-- SET ModifiedDate = getdate()

-- ,IsActive = 'N'

-- WHERE Id = @Id

-- COMMIT;

-- SELECT 1;--Deleted successfully

-- END TRY

-- BEGIN CATCH

-- ROLLBACK;

-- SELECT 0;--record is not deleted due to some error

-- END CATCH

-- END

-- ELSE

-- BEGIN

-- SELECT 2;--Record is not available for delete

-- END

-- END

--END

--INSERT INTO tblShopBridgeDetails

-- VALUES (

-- 'Apple'

-- ,'Fruit'

-- ,'100'

-- ,'Y'

-- ,GETDATE()

-- ,NULL

-- )

--Create Table tblErrorLog

--(

-- Id Int identity(1,1) primary key,

-- Message Varchar(500) ,

-- Stacktrace Varchar(500),

-- Source Varchar(500),

-- Target Varchar(500),

-- LogDate Datetime

--)

--CREATE PROCEDURE SP\_ErrorLog @Message VARCHAR(500)

-- ,@Stacktrace VARCHAR(500)

-- ,@Source VARCHAR(500)

-- ,@Target VARCHAR(500)

--AS

--BEGIN

-- INSERT INTO ErrorLog

-- VALUES (

-- @Message

-- ,@Stacktrace

-- ,@Source

-- ,@Target

-- ,GETDATE()

-- )

--END

select \* from tblShopBridgeDetails

2)Create Class file under **model** folder

a) Entity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations;

namespace ShopBridge.Models

{

public class Entity

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public double Price { get; set; }

}

}

b)DAL.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

using ShopBridge.Models;

using Newtonsoft.Json;

namespace ShopBridge.Models

{

public class DAL

{

string Constr = ConfigurationManager.ConnectionStrings["conString"].ConnectionString;

public string GetDetails()

{

using (SqlConnection con = new SqlConnection(Constr))

{

try

{

if (con.State == ConnectionState.Closed)//Check for connection status

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ShopBridge", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Task", "Get");

SqlDataAdapter sda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows.Count > 0)

{

return JsonConvert.SerializeObject(dt);

}

else

{

return "No data found";

}

}

catch (Exception ex)

{

//// show error and store error in DB

//if (con.State == ConnectionState.Closed)

// con.Open();

//SqlCommand cmd1 = new SqlCommand("SP\_ErrorLog", con);

//cmd1.CommandType = CommandType.StoredProcedure;

//cmd1.Parameters.AddWithValue("@Message", ex.Message);

//cmd1.Parameters.AddWithValue("@Stacktrace", ex.StackTrace);

//cmd1.Parameters.AddWithValue("@Source", ex.Source);

//cmd1.Parameters.AddWithValue("@Target", ex.TargetSite);

//cmd1.ExecuteNonQuery();

ErrorLog(ex);//Store exception or error details

return "Some error occurs while fetching record";

}

}

}

public string GetDetails(int id)

{

using (SqlConnection con = new SqlConnection(Constr))

{

try

{

if (con.State == ConnectionState.Closed)

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ShopBridge", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Task", "GetById");

cmd.Parameters.AddWithValue("@Id", id);

SqlDataAdapter sda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows.Count > 0)

{

return JsonConvert.SerializeObject(dt);

}

else

{

return "No data found";

}

}

catch (Exception ex)

{

//// show error and store error in DB

//if (con.State == ConnectionState.Closed)

// con.Open();

//SqlCommand cmd1 = new SqlCommand("SP\_ErrorLog", con);

//cmd1.CommandType = CommandType.StoredProcedure;

//cmd1.Parameters.AddWithValue("@Message", ex.Message);

//cmd1.Parameters.AddWithValue("@Stacktrace", ex.StackTrace);

//cmd1.Parameters.AddWithValue("@Source", ex.Source);

//cmd1.Parameters.AddWithValue("@Target", ex.TargetSite);

//cmd1.ExecuteNonQuery();

ErrorLog(ex);//Store error details

return "Some error occurs while fetching record";

}

}

}

public string Post(Entity En)

{

using (SqlConnection con = new SqlConnection(Constr))

{

try

{

if (con.State == ConnectionState.Closed)

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ShopBridge", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Task", "Insert");

cmd.Parameters.AddWithValue("@Name", En.Name);

cmd.Parameters.AddWithValue("@Description", En.Description);

cmd.Parameters.AddWithValue("@Price", En.Price);

SqlDataAdapter sda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows.Count > 0)

{

if (dt.Rows[0][0].ToString() == "1")

return "Data Inserted Successfully";

else if (dt.Rows[0][0].ToString() == "0")

return "Data Not Inserted due to some error";

else

return "Duplicate records can not be allow to insert";

//return JsonConvert.SerializeObject(dt);

}

else

{

return "Data Not Inserted";

}

}

catch (Exception ex)

{

ErrorLog(ex);// store error details

return "Some error occurs while Inserting record";

}

}

}

public string Put(int id, Entity En)

{

using (SqlConnection con = new SqlConnection(Constr))

{

try

{

if (con.State == ConnectionState.Closed)

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ShopBridge", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Task", "Update");

cmd.Parameters.AddWithValue("@Id", id);

cmd.Parameters.AddWithValue("@Name", En.Name);

cmd.Parameters.AddWithValue("@Description", En.Description);

cmd.Parameters.AddWithValue("@Price", En.Price);

SqlDataAdapter sda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows.Count > 0)

{

if (dt.Rows[0][0].ToString() == "1")

return "Data Updated Successfully";

else if (dt.Rows[0][0].ToString() == "0")

return "Data Not Updated";

else

return "Record is not available for update";

//return JsonConvert.SerializeObject(dt);

}

else

{

return "Data Not Updated";

}

}

catch (Exception ex)

{

//// show error and store error in DB

ErrorLog(ex);//store error details

return "Some error occurs while updating record";

}

}

}

public string delete(int id)

{

using (SqlConnection con = new SqlConnection(Constr))

{

try

{

if (con.State == ConnectionState.Closed)

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ShopBridge", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Task", "Delete");

cmd.Parameters.AddWithValue("@Id", id);

SqlDataAdapter sda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows.Count > 0)

{

if (dt.Rows[0][0].ToString() == "1")

return "Data Deletd Successfully";

else if (dt.Rows[0][0].ToString() == "0")

return "Data Not Deleted";

else

return "Record is not available for delete";

//return JsonConvert.SerializeObject(dt);

}

else

{

return "Data Not Deleted";

}

}

catch (Exception ex)

{

//// show error and store error in DB

ErrorLog(ex);//store error details

return "Some error occurs while deleting record";

}

}

}

public void ErrorLog(Exception ex)

{

using (SqlConnection con = new SqlConnection(Constr))

{

if (con.State == ConnectionState.Closed)

con.Open();

SqlCommand cmd1 = new SqlCommand("SP\_ErrorLog", con);

cmd1.CommandType = CommandType.StoredProcedure;

cmd1.Parameters.AddWithValue("@Message", ex.Message);

cmd1.Parameters.AddWithValue("@Stacktrace", ex.StackTrace);

cmd1.Parameters.AddWithValue("@Source", ex.Source);

cmd1.Parameters.AddWithValue("@Target", ex.TargetSite);

cmd1.ExecuteNonQuery();

}

}

}

}

c)Validation.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

using System.Web;

namespace ShopBridge.Models

{

public class Validation

{

public string Validate(Entity En)

{

string result = "";

if (!string.IsNullOrEmpty(En.Name))

{

}

else

{ //

//Name is required

result = result + "Name is required;";

}

if (!string.IsNullOrEmpty(En.Description))

{

}

else

{ //

//Description is required

result = result + "Description is required;";

}

if (!string.IsNullOrEmpty(En.Price.ToString()))

{

if (En.Price == 0)

return result + "Enter Valid price";

if (!Regex.IsMatch(En.Price.ToString(), @"[\d]{1,4}([.][\d]{1,2})?"))

result = result + "Enter Valid price";

}

else

{ //

//Price is required

result = result + "Price is required";

}

return result;

}

}

}

3) write a connection string in web.config file, for global access

<connectionStrings>

<add name="conString"

providerName="System.Data.SqlClient"

connectionString="Data Source=localhost;Initial Catalog=ShopBridgeDB;Integrated Security=True;" />

</connectionStrings>

4)We have created ShopBridgeController.cs file unser controller folder.

In ShopBridgecontroller.cs folder have various methos like

Get(),Put(),Post(),Delete()

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

using ShopBridge.Models;

using Newtonsoft.Json;

namespace ShopBridge.Controllers

{

public class ShopBridgeController : ApiController

{

string Constr = ConfigurationManager.ConnectionStrings["conString"].ConnectionString;

// GET api/ShopBridge

public string Get()

{

string result = "";

DAL objDal = new DAL();

result=objDal.GetDetails();

return result;

}

// GET api/ShopBridge/5

public string Get(int id)

{

string result = "";

DAL objDal = new DAL();

result = objDal.GetDetails(id);

return result;

}

// POST api/ShopBridge

public string Post([FromBody] Entity En)

{

Validation objvalidation = new Validation();

string IsValidString = objvalidation.Validate(En);

if (string.IsNullOrEmpty(IsValidString))

{

string result = "";

DAL objDal = new DAL();

result = objDal.Post(En);

return result;

}

else

{

return IsValidString;

}

}

// PUT api/ShopBridge/5

public string Put(int id, [FromBody] Entity En)

{

Validation objvalidation = new Validation();

string IsValidString = objvalidation.Validate(En);

if (string.IsNullOrEmpty(IsValidString))

{

string result = "";

DAL objDal = new DAL();

result = objDal.Put(id,En);

return result;

}

else

{

return IsValidString;

}

}

// DELETE api/ShopBridge/5

public string Delete(int id)

{

string result = "";

DAL objDal = new DAL();

result = objDal.delete(id);

return result;

}

}

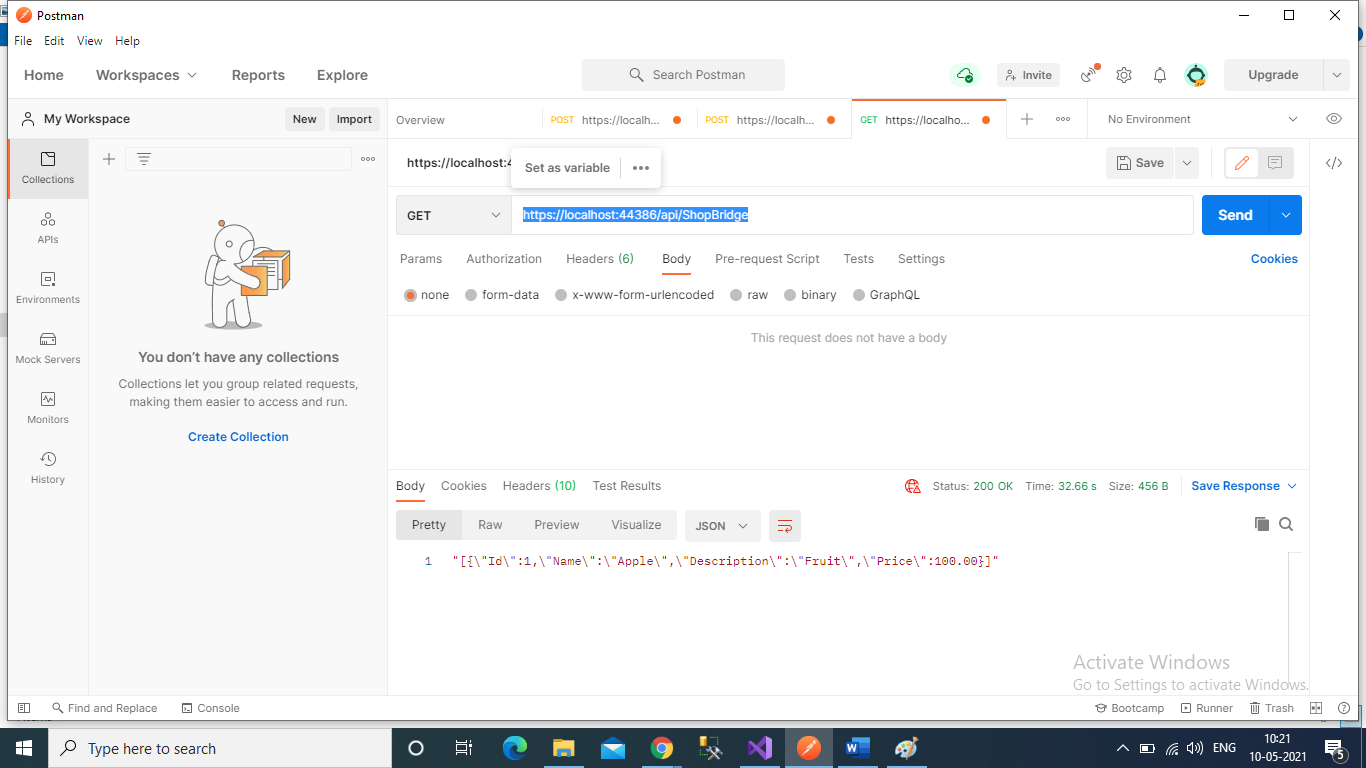
}

For Testing the Web API we have use the **Postman.**

**1:- GET method**

[**URL:-**](URL:-)<https://localhost:44386/api/ShopBridge>

JSON output:- "[{\"Id\":1,\"Name\":\"Apple\",\"Description\":\"Fruit\",\"Price\":100.00}]"

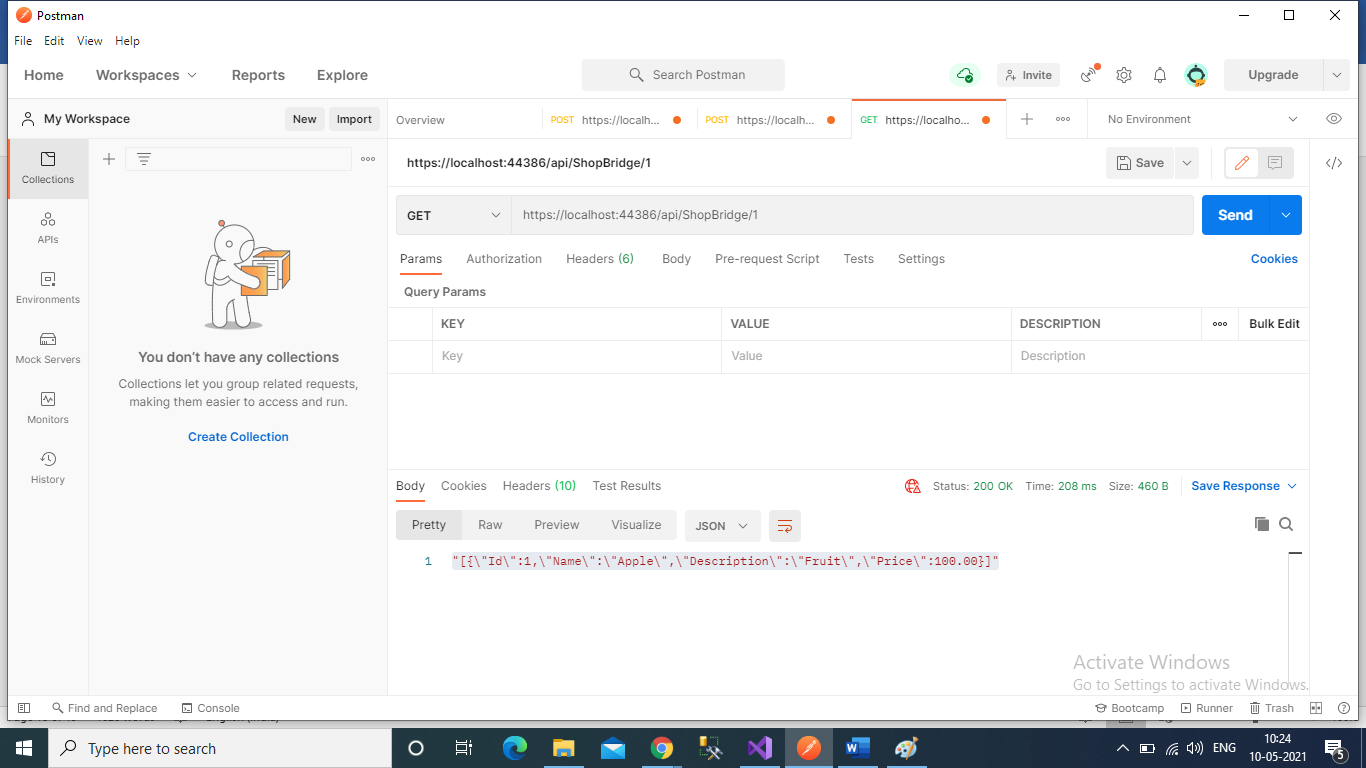


2:-GET method:- Get data by ID.

Test case 1:-

<URL:-> <https://localhost:44386/api/ShopBridge/1>

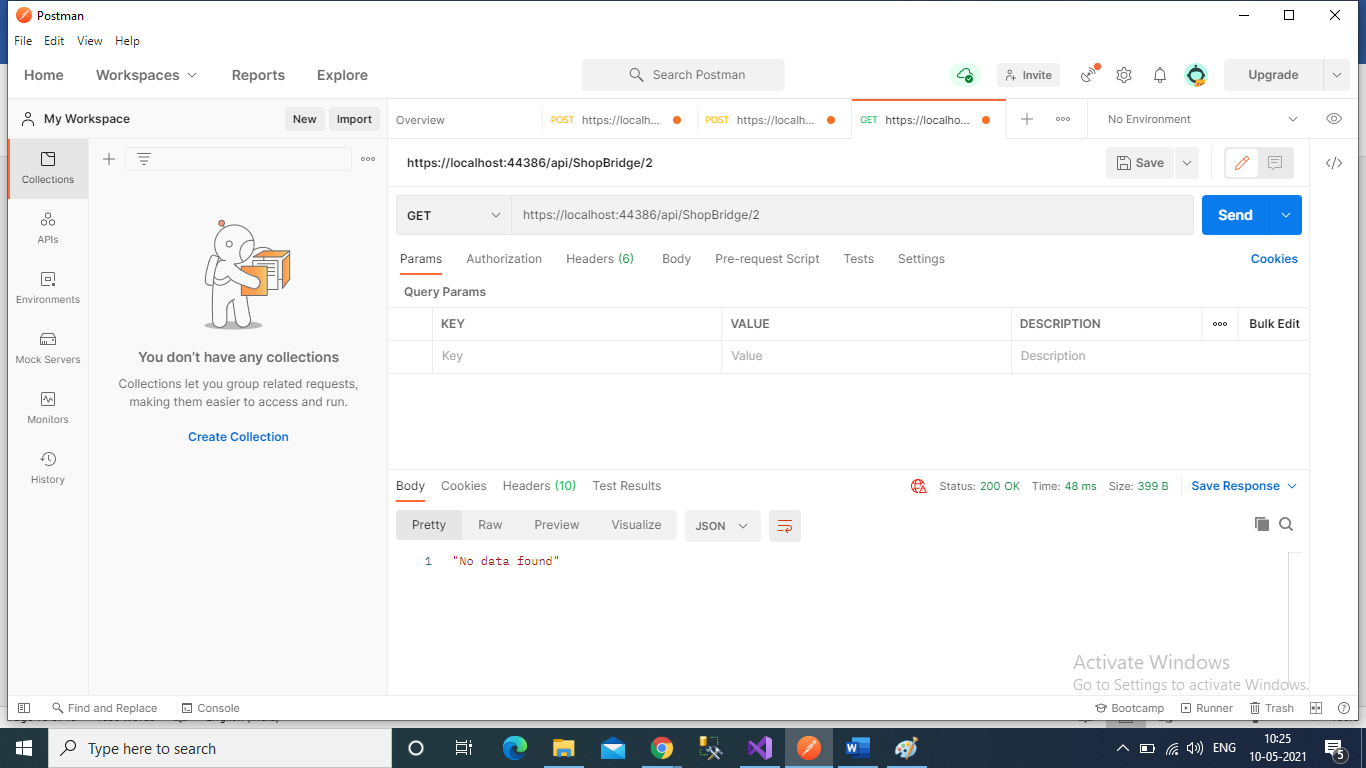
Json Data:- "[{\"Id\":1,\"Name\":\"Apple\",\"Description\":\"Fruit\",\"Price\":100.00}]"



Testcase2:-

<URL:-> <https://localhost:44386/api/ShopBridge/2>

Output:- "No data found"



2:- Post method:-

Testcase 1:

<URL:-> <https://localhost:44386/api/ShopBridge>

Body:-

{

    "Name":"Banana",

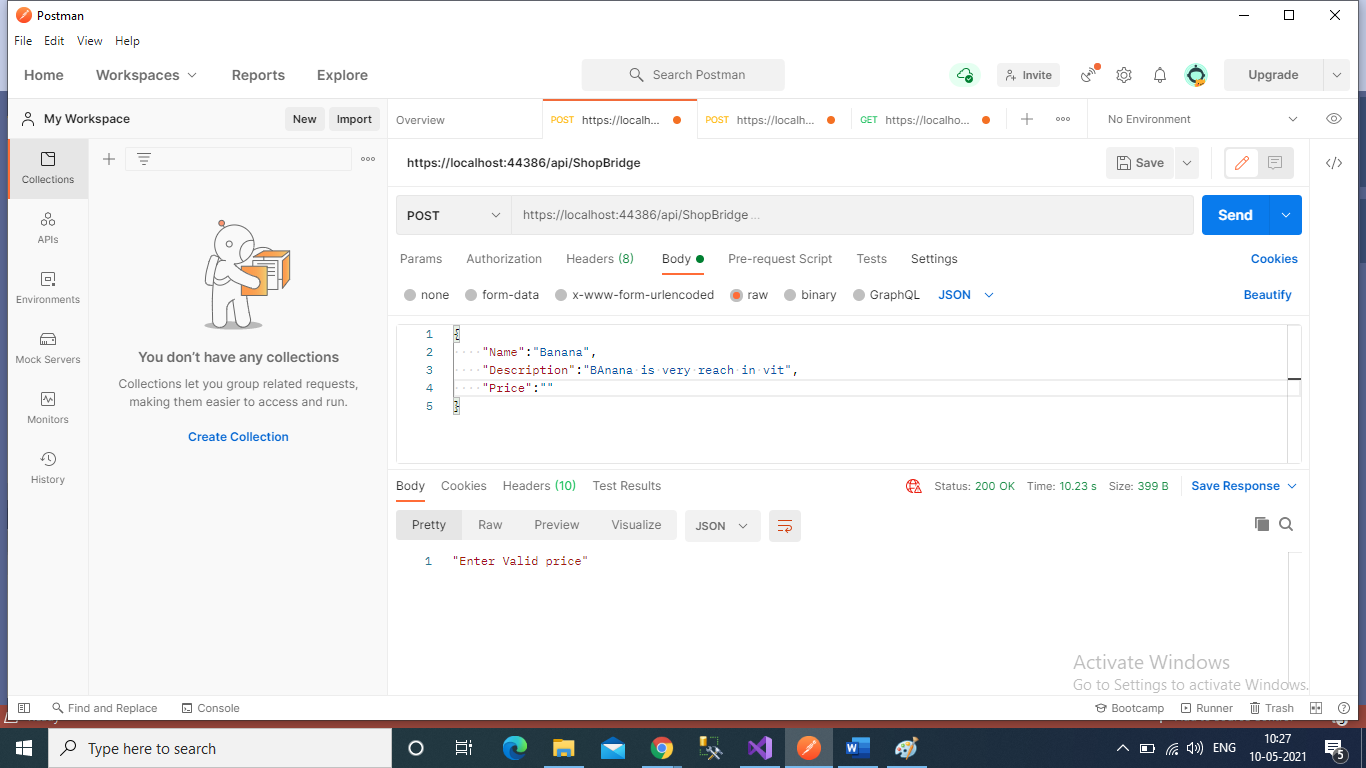
    "Description":"BAnana is very reach in vit",

    "Price":""

}

Output:-

"Enter Valid price"



Testcase 2:

URL:- <https://localhost:44386/api/ShopBridge>

Body:

{

    "Name":"apple",

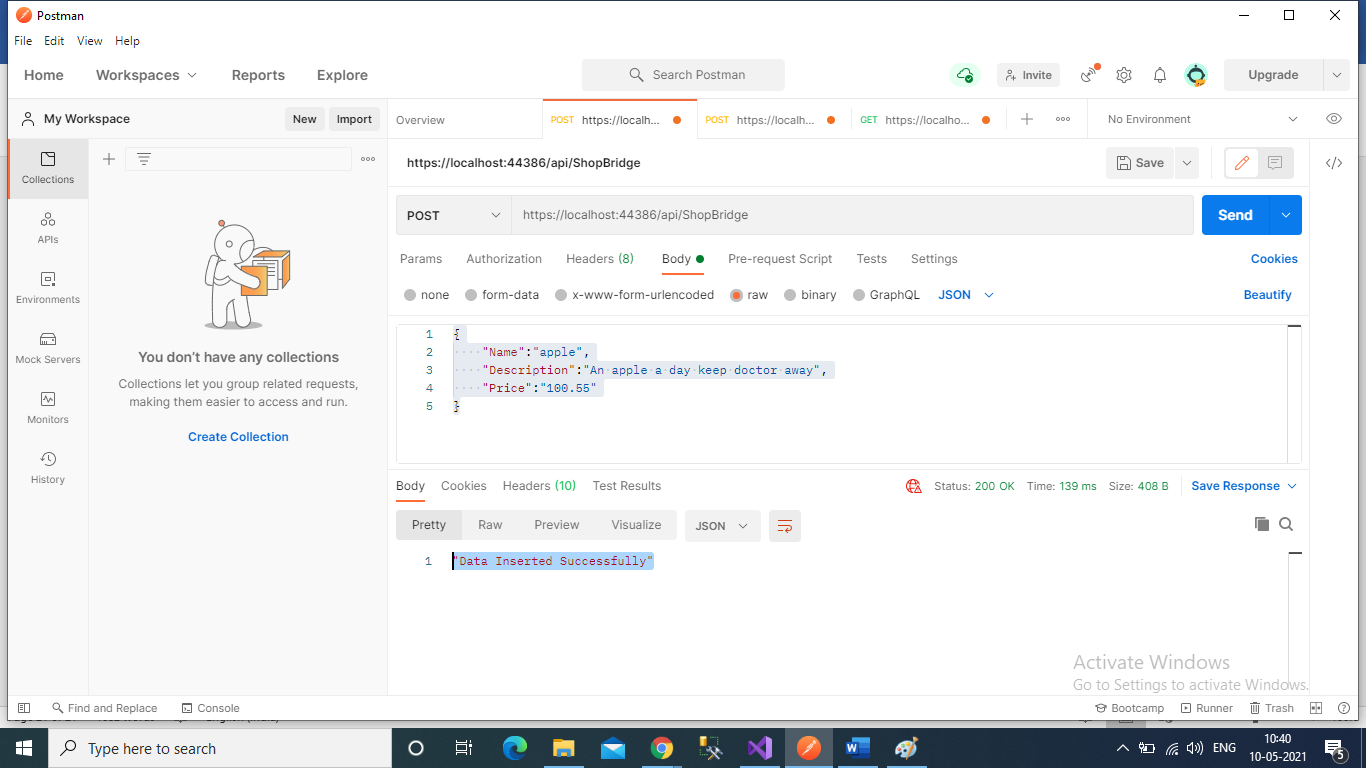
    "Description":"An apple a day keep doctor away",

    "Price":"100.55"

}

Output:

"Data Inserted Successfully"



Testcase 3:

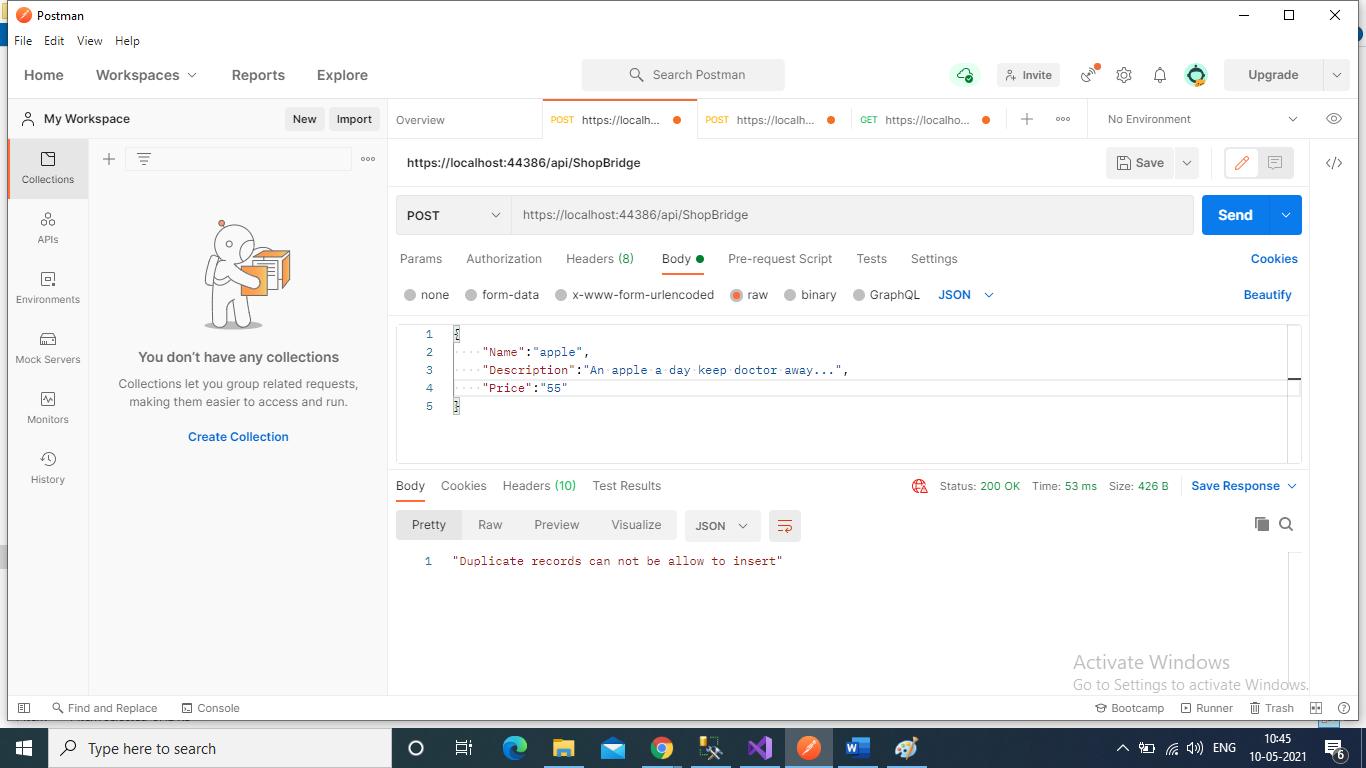
<URL:-> <https://localhost:44386/api/ShopBridge>

Body:-

    "Name":"apple",

    "Description":"An apple a day keep doctor away...",

    "Price":"55"



3) Put Method:-

Test case 1:-

URL:- <https://localhost:44386/api/ShopBridge/3>

Body:-

{

    "Name":"Pineapple",

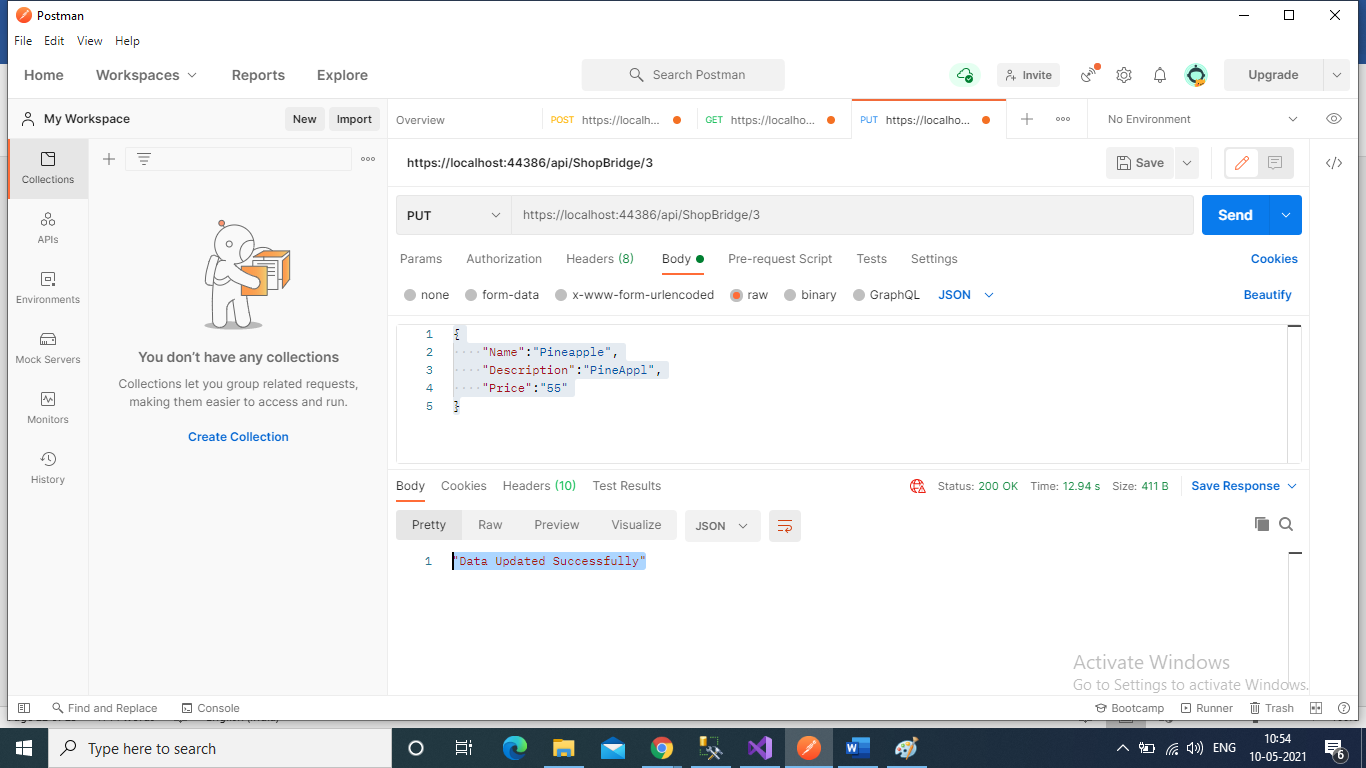
    "Description":"PineAppl",

    "Price":"55"

}

Output:

"Data Updated Successfully"



Testcase2:-

URL: <https://localhost:44386/api/ShopBridge/5>

Body:-

{

    "Name":"Pineapple",

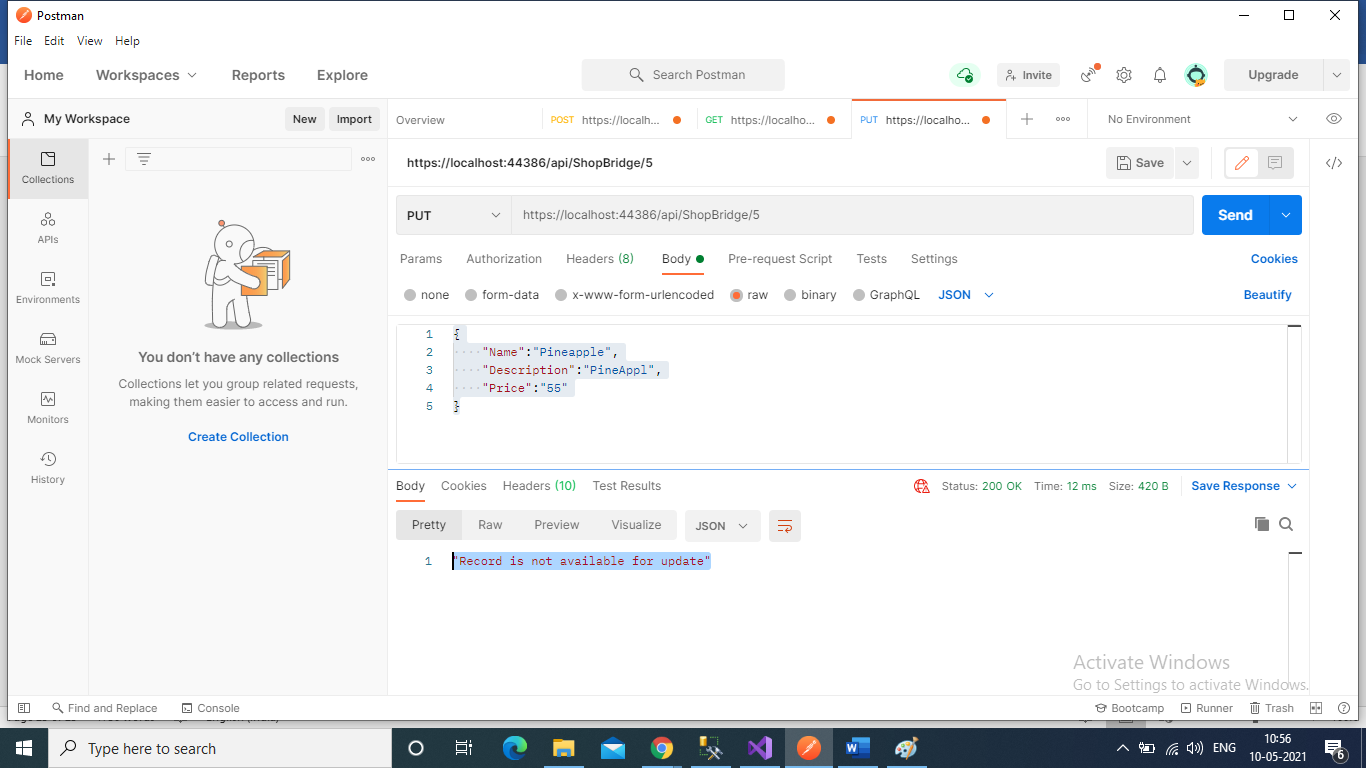
    "Description":"PineAppl",

    "Price":"55"

}

Output:-

"Record is not available for update"



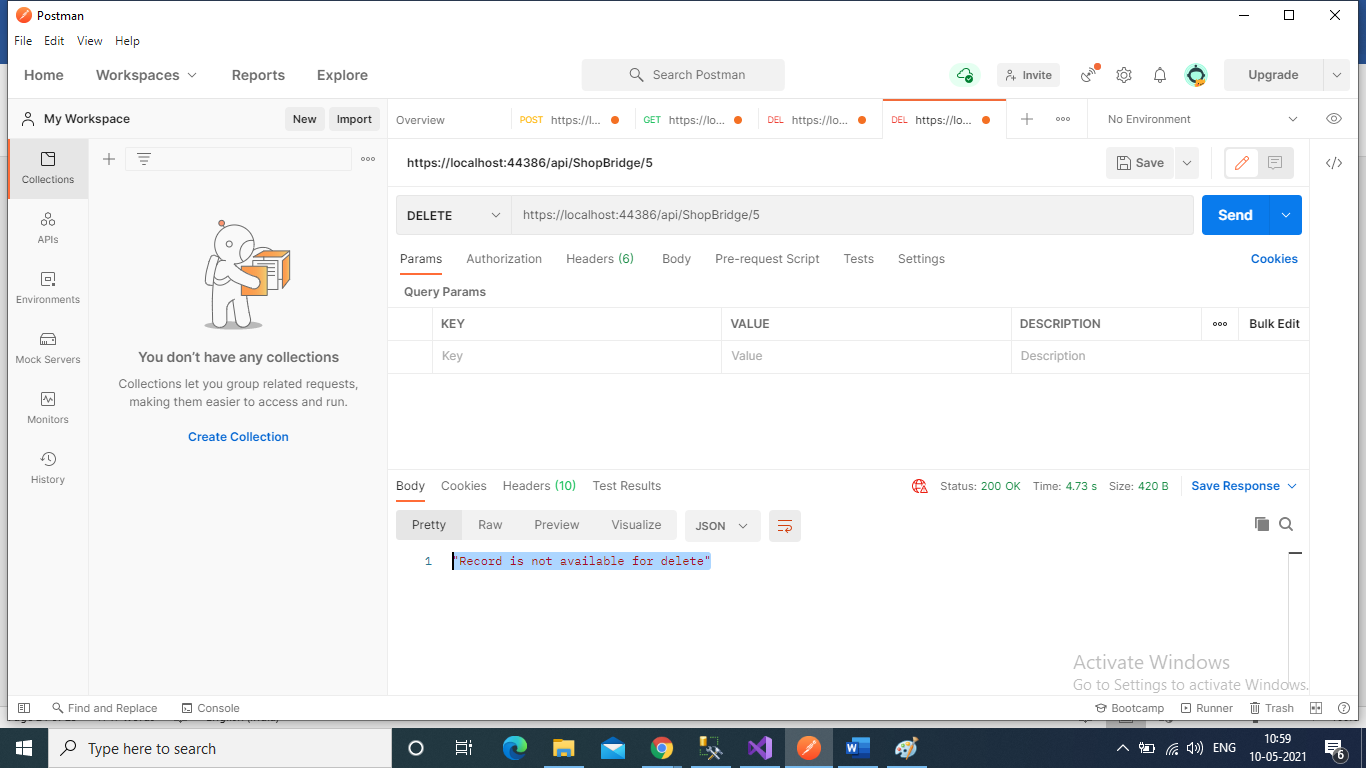
4:- Delete Method

Testcase 1:-

URL:- <https://localhost:44386/api/ShopBridge/5>

Output:-

"Record is not available for delete"



Testcase2:

URL:- https://localhost:44386/api/ShopBridge/3

Output:-

"Data Deletd Successfully"

