**STEP BY STEP TEST FRAMEWORK SET UP**

**To set up a test framework I would consider the following steps;**

**Step 1- I would download the “Visual Studio IDE- Community 2017” free version.**

**Step 2- I would then integrate Specflow Extension with Visual Studio IDE under Tools**, then Extension and updates (by default I should have some extensions and updates already installed) and make sure I’m online with internet access. I would need to search for Specflow in the search box provided and then download Specflow (if not already downloaded). If I get a message at the bottom of the window that says “your changes will be scheduled” meaning I need to close Visual studio and re-open it. A window should pop up and ask me to modify visual studio, and I would then modify visual studio. Once it has finished modifying I would then close the box. All I need to do is re-open visual studio. I need to go back and check on Tools- extensions and updates to see if it has installed Specflow (with green tick).

**Step 3- Create a New Test-Project. At this stage i would need to create a folder for the new test project I am about to work on.**

On visual studio, Go to New-Project, I would select Test framework- then select “Unit Test Project (.Net framework)”. I can then write the name of my new project. I also need to create a folder for this on e.g Desktop by browsing for the folder location. And I then save by clicking OK. Now my “Solution- under Solution Explorer” is now created on Visual Studio. Under solution explorer, I should have my new test file, properties, references, packages configuration and unit test, all created (even though I may not be needing the Unit test file- which is usually used by Developers, I can delete unit test). I should have the “Test Explorer” as well which shows what test I would be running on that project. I need to click “build solution” in order to refresh my code, the status would be displayed on output.

**Step 4- Adding Nugets Packages**

All I need to do is go to Tools- select “Nuget package manager”- “Manage nuget packages for solution”. And There would already be 2 nuget packages installed by default (MSTest.Test adapter by Microsoft and MSTest.TestFramework by Microsoft).

Note- I can also add my nuget packages under “solution”.

The following Nugets packages would be added one after the other;

1. NUnit
2. NUnit.Console
3. NUnit.Runner
4. NUnit3TestAdapter
5. SpecFlow
6. SpecFlow. Nunit
7. Selenium.WebDriver
8. Selenium.Support
9. Selenium.WebDriver.IEDriver
10. Selenium. Firefox.WebDriver
11. Selenium. WebDriver.ChromeDriver

To install the above packages, I need to click on browse and type each nuget packages till all is being installed. I would then see the download button turned green under each nuget packages, to show it has been installed.

Note- We can check Packages.configuration to ensure all our nuget packages have been installed or added.

**Step 5a- Creating Folders Structure. Folders to be created are; Feature file, Step Definition, Page Objects and Utilities.**

Go to the coursework and add new folder and name as below;

Feature file

Page Objects

Step Definitions

Utilities

**Step 5b- Creating Features file, step definition, page object and Hooks**

For me to create feature file, I need to ensure the specflow has been installed after adding new item. Then I can save with the name of the feature file I am creating e.g Homepage Product Information, Login or Registration etc. On the feature file, I need to write my test scripts. I then need to generate step definition for each test steps and save it in my folder created under step definition. Also, I added hooks (Hooks has before and after scenarios) under utility folder.

**Writing Test codes**

Codes were written for the feature steps created under step definitions and the tests were ran and passed accordingly without error.

**Challenges**

1. I couldn’t launch into visual studio without it downloading/installing all over.
2. The output at the bottom of the window is not coming up. Even though I could see my Test explorer, solution explorer. The Output tab was positioned vertically instead of being positioned horizontally.
3. I mistakenly saved the steps (test scripts) generated from visual studio, just under my coursework folder instead of saving it directly under step definitions.

**Overcoming Challenges**

1. I had to allow time for it to download
2. With the help of one of my team members, I was able to restore and positioned the tools at the appropriate places by following the instructions given to me via telephone call.
3. I had to delete the step definitions folder on visual studio and re-add it, so that I can generate step definitions and re-save.

**Summary of API Testing**

**What is an API?**

API stands for Application Programming Interface. An API is a software intermediary that allows two applications to talk to each other.  In other words, an API is the messenger that delivers your request to the provider that you’re requesting it from and then delivers the response back to you.

**How do APIs work?**

Imagine a waiter in a restaurant.  You, the customer, are sitting at the table with a menu of choices to order from, and the kitchen is the provider who will fulfill your order.

You need a link to communicate your order to the kitchen and then to deliver your food back to your table. It can’t be the chef because she’s cooking in the kitchen. You need something to connect the customer who’s ordering food and the chef who prepares it.  That’s where the waiter — or the API — enters the picture.

The waiter takes your order, delivers it to the kitchen, telling the kitchen what to do. It then delivers the response, in this case, the food, back to you. Moreover, if it’s designed correctly, hopefully, your order won’t crash!

**A real example of an API**

How are APIs used in the real world? Here’s a very common scenario – booking a flight.

When you search for flights online, you have a menu of options to choose from. You choose a departure city and date, a return city and date, cabin class, and other variables like your meal, your seat, or baggage requests.

To book your flight, you need to interact with the airline’s website to access the airline’s database to see if any seats are available on those dates, and what the cost might be based on the date, flight time, route popularity, etc.

You need access to that information from the airline’s database, whether you’re interacting with it from the website or an online travel service that aggregates information from multiple airlines. Alternatively, you might be accessing the information from a mobile phone. In any case, you need to get the information, and so the application must interact with the airline’s API, giving it access to the airline’s data.

The API is the interface that, like your helpful waiter, runs and delivers the data from the application you’re using to the airline’s systems over the Internet. It also then takes the airline’s response to your request and delivers right back to the travel application you’re using. Moreover, through each step of the process, it facilitates the interaction between the application and the airline’s systems – from seat selection to payment and booking.

APIs do the same for all interactions between applications, data, and devices. They allow the transmission of data from system to system, creating connectivity. APIs provide a standard way of accessing any application data, or device, whether it’s accessing cloud applications like [Salesforce](http://mulesoft.com/integration-solutions/saas/salesforce), or shopping from your mobile phone.

**Types of APIs**

There are numerous types of APIs. For example, you may have heard of Java APIs, or interfaces within classes that let objects talk to each other in the Java programming language. Along with program-centric APIs, there are also Web APIs such as the Simple Object Access Protocol (SOAP), Remote Procedure Call (RPC), and perhaps the most popular—at least in name— Representational State Transfer (REST). There are 15,000 publicly available APIs, according to [Programmable Web](http://snip.ly/wwnim), and many thousands of more private APIs that companies use to expand their internal and external capabilities.

**API Testing**

An API or Application Programming Interface is a set of programming instructions for accessing a web-based software application. There are mainly 4 methods involve in API Testinglike GET, POST, Delete, and PUT. We need to check response code, response message and response body inAPIi Testing.

**How is API testing done**

API testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. Since APIs lack a GUI, API testing is performed at the message layer.

**Tools used for API testing**

SoapUI is an automation testing tool for SOAP and REST API. You can perform functional testing, performance testing, security testing, and data-driven testing. It will provide you with the reports for testing and will allow you to export the data.

**API testing with postman**

Postman is an interactive and automatic tool for verifying the APIs of your project. Postman is a Google Chrome app for interacting with HTTP APIs. It presents you with a friendly GUI for constructing requests and reading responses. It works on the backend, and makes sure that each API is working as intended.

A test in Postman is fundamentally a JavaScript code, which run after a request is sent and a response has been received from the server. POSTMAN is very easy to use. It provides collection of API calls, and one has to follow that collection of APIcalls for testing APIs of application

**How to run API in postman**

Enable authorization

In Postman, select an API method.

Click the Authorization tab.

Choose OAuth 2.0 and add the following information from the table below.

Click Get access token.

Postman starts the authentication flow and prompts you to save the access token.

Select Add token to header.

**How to test an API**

API Testing Best Practices

Test for the typical or expected results first.

Add stress to the system through a series of API load tests.

Test for failure. ...

Group test cases by test category.

Prioritize API function calls so that it will be easy for testers to test quickly and easily.

**Summary of SQL**

Structured Query Language (SQL) is a standard computer language for relational database management and data manipulation. SQL is used to query, insert, update and modify data stored in a relational database.

SQL statements are used to perform tasks such as updatedata on a database, or retrieve data from a database

Database

A database consists of one or more tables. A table is identified by its name. A table is made up of columns and rows. Columns contain the column name and data type. Rows contain the records or data for the columns.

Basic SQL

Each record has a unique identifier or primary key. SQL, which stands for Structured Query Language, is used to communicate with a database.

SQL has five important syntax to create table, select, update, insert, and delete records.

* SELECT - get data from a database table
* UPDATE - change data in a database table
* DELETE - remove data from a database table
* INSERT INTO - insert new data in a database table
* CREATE TABLE – create database table

DELETE, UPDATE, and SELECT statements can use a WHERE clause to limit the rows that are affected or returned and to capture foreign reference relationships.

SELECT

The SELECT is used to query the database and retrieve selected data that match the specific criteria that you specify:  
  
SELECT column1 [, column2, ...]   
FROM tablename   
WHERE condition   
  
The conditional clause can include these operators

* = Equal
* > Greater than
* < Less than
* >= Greater than or equal
* <= Less than or equal
* <> Not equal to
* LIKE pattern matching operator

SELECT \* FROM tablename   
  
returns all the data from the table.

Use single quotes around text values (most database systems will also accept double quotes). Numerical values should not be enclosed in quotes.

LIKE matches a pattern. The wildcard % is used to denote 0 or more characters.

* 'A%' : matches all strings that start with A
* '%a' : matches all strings that end with a
* '%a%' : matches all strings that contain an a

CREATE TABLE

The CREATE TABLE statement is used to create a new table. The format is:   
  
CREATE TABLE tablename   
(column1 data type,   
column2 data type,   
column3 data type);

* char(size): Fixed length character string.
* varchar(size): Variable-length character string. Max size is specified in parenthesis.
* number(size): Number value with a max number of columns specified in parenthesis
* date: Date value
* number(size,d): A number with a maximum number of digits of "size" and a maximum number of "d" digits to the right of the decimal

INSERT VALUES

Once a table has been created data can be inserted using INSERT INTO command.  
  
INSERT INTO tablename   
(col1, ... , coln)   
VALUES (val1, ... , valn)

UPDATE

To change the data values in a pre-existing table, the UPDATE command can be used.   
  
UPDATE tablename   
SET colX = valX [, colY = valY, ...]   
WHERE condition

DELETE

The DELETE command can be used to remove a record(s) from a table.   
  
DELETE FROM tablename   
WHERE condition   
  
To delete all the records from a table without deleting the table do   
  
DELETE \* FROM tablename

DROP

To remove an entire table from the database use the DROP command.  
  
DROP TABLE tablename

ORDER BY

ORDER BY clause can order column name in either ascending (ASC) or descending (DESC) order.  
  
ORDER BY col\_name ASC

AND / OR

AND and OR can join two or more conditions in a WHERE clause. AND will return data when all the conditions are true. OR will return data when any one of the conditions is true.

IN

IN operator is used when you know the exact value you want to return for at least one of the columns   
  
SELECT \* FROM table\_name WHERE col\_name IN (val1, val2, ...)

BETWEEN / AND

The BETWEEN ... AND operator selects a range of data between two values. These values can be numbers, text, or dates.   
  
SELECT \* FROM table\_name WHERE col\_name BETWEEN val1 AND val2

JOIN

There are times when we need to collate data from two or more tables. That is called a join. Tables in a database are related to each other through their keys. We can associate data in various tables without repeating them. Example we could have a table called Customers which could have information about customers like their name, address, phone numbers. We could have another table called Products that has information regarding the products like part number, product name, manufacturer, number in stock, unit price. A third table called Orders could have information regarding what product was ordered, by whom, the date the order was placed, and quantity. Here are the tables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers | | | | |
| Cust\_ID | FirstName | LastName | Address | Phone |
| 01 | Mickey | Mouse | 123 Gouda St. | 456-7890 |
| 02 | Donald | Duck | 325 Eider Ln. | 786-2365 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Products | | | | |
| Part\_No | Name | Manufacturer | In\_Stock | Price |
| 20-45 | Hammer | Stanley | 57 | 3.50 |
| 21-68 | ScrewDriver | DeVries | 84 | 2.75 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Orders | | | | |
| Order\_No | Part\_No | Cust\_ID | Date | Quantity |
| 2005-27 | 21-68 | 02 | 31 Oct 2005 | 2 |
| 2005-34 | 20-45 | 01 | 02 Nov 2005 | 3 |

We can obtain information on who has ordered what:   
  
SELECT Customers.FirstName, Customers.LastName, Products.Name   
FROM Customers, Products, Orders   
WHERE Customers.Cust\_ID = Orders.Cust\_ID AND Products.Part\_No = Orders.Part\_No

We can select data from two tables with INNER JOIN. The INNER JOIN returns all rows from both tables where there is a match. If there are rows in Customers that do not have matches in Orders, those rows will not be listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
INNER JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

The LEFT JOIN returns all the rows from the first table (Customers), even if there are no matches in the second table (Orders). If there are rows in Customers that do not have matches in Orders, those rows also will be listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
LEFT JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

The RIGHT JOIN returns all the rows from the second table (Orders), even if there are no matches in the first table (Customers). If there had been any rows in Orders that did not have matches Customers, those rows also would have been listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
RIGHT JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

ALTER TABLE

With ALTER TABLE you can add or delete columns in an existing table. When you add a column you must specify a data type.   
ALTER TABLE table\_name   
ADD col\_name datatype   
  
ALTER TABLE table\_name   
DROP COLUMN col\_name

UNION

The UNION command is used to select data from two tables very similar to the JOIN command. But the UNION command can be used only with columns having the same datatype. With UNION only distinct values are selected, i.e. if there are common data in the two tables only one instance of that data is returned.   
  
SELECT Name FROM Customers\_USA   
UNION   
SELECT Name FROM Customers\_Asia   
  
This will select all the customers from USA and Asia but if there is a name that occurs in both the tables it will return only one such name. To get all the names use UNION ALL instead.

GROUP BY

The GROUP BY was added to SQL so that aggregate functions could return a result grouped by column values.   
  
SELECT col\_name, function (col\_name) FROM table\_name GROUP BY col\_name

HAVING keyword was introduced because the WHERE keyword could not be used. HAVING states a condition.   
  
SELECT clo\_name, function (col\_name) FROM table\_name   
GROUP BY col\_name   
HAVING function (col\_name) condition value