Muhammad Haider Ejaz

SPECFLOW HYBRID FRAMEWORK

v1.0

Contents

[1. Framework Introduction 2](#_Toc111663658)

[2. Specflow Introduction 2](#_Toc111663659)

[2.1 Why Specflow 2](#_Toc111663660)

[2.2 Specflow plus Living Doc 3](#_Toc111663661)

[3. Specflow Framework Packages 4](#_Toc111663662)

[4. Project Structure 5](#_Toc111663663)

[5. Assignment Detail 6](#_Toc111663664)

[ Solution 6](#_Toc111663665)

[5.1 Task 1 6](#_Toc111663666)

[5.2 Task 2 9](#_Toc111663667)

[5.3 Bonus Requirement 12](#_Toc111663668)

**SPECFLOW HYBRID FRAMEWORK DOCUMENTATION**

# Framework Introduction

This framework is designed for testing purposes so we can test our web application and API’s easily. Mainly This Framework has more focused on Web & API’s testing but yes we can add Mobile Testing as well with just one configuration file through which we can decide the devices, browsers etc.

# Specflow Introduction

SpecFlow is a test automation solution for .NET built upon the BDD paradigm. Use SpecFlow to define, manage and automatically execute human-readable acceptance tests in .NET projects (Full Framework and .NET Core).

SpecFlow tests are written using [Gherkin](https://cucumber.io/docs/gherkin/) Language, which allows you to write test cases using natural languages. SpecFlow uses the official Gherkin parser, which supports over 70 languages. These tests are then tied to your application code using [Bindings](https://docs.specflow.org/projects/specflow/en/latest/Bindings/Bindings.html), allowing you to execute the tests using the testing framework of your choice. You can also execute your tests using SpecFlow’s dedicated test runner, SpecFlow+ Runner.

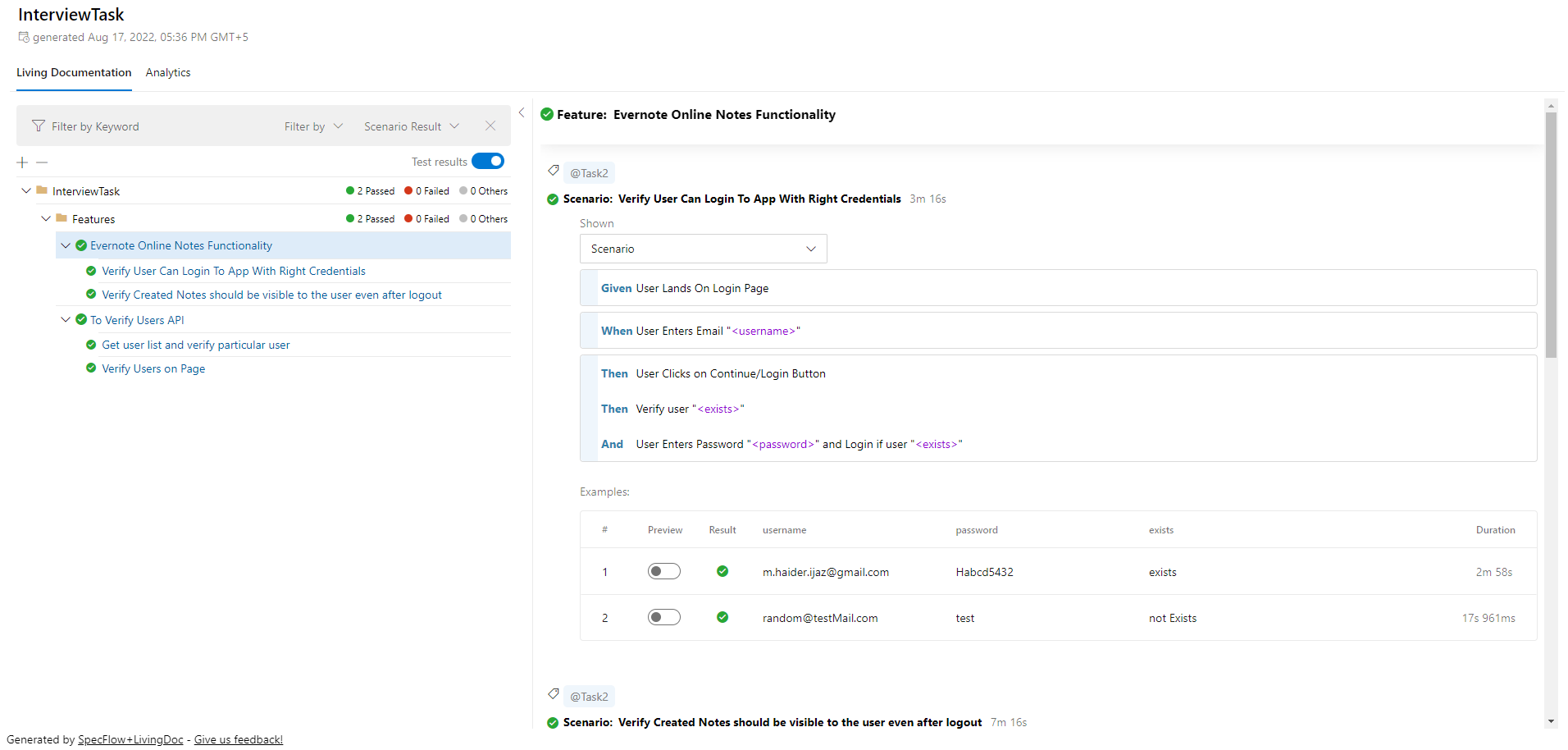
## Why Specflow

* **Step definitions** - Step definitions provide the connection between Gherkin feature specifications and application interfaces for test automation.
* **Navigation to Step definitions** - Don’t waste your time searching for the correct definition across your binding classes, just right-click and jump to the relevant code.
* **Hooks** - Hooks (event bindings) can be used to perform additional automation logic at specific times, such as any setup required before executing a scenario.
* **Context Injection** - SpecFlow supports a dependency injection framework that can instantiate and inject context for scenarios. This allows you to group the shared state in context classes, and inject them into every binding class that needs access to that shared state.

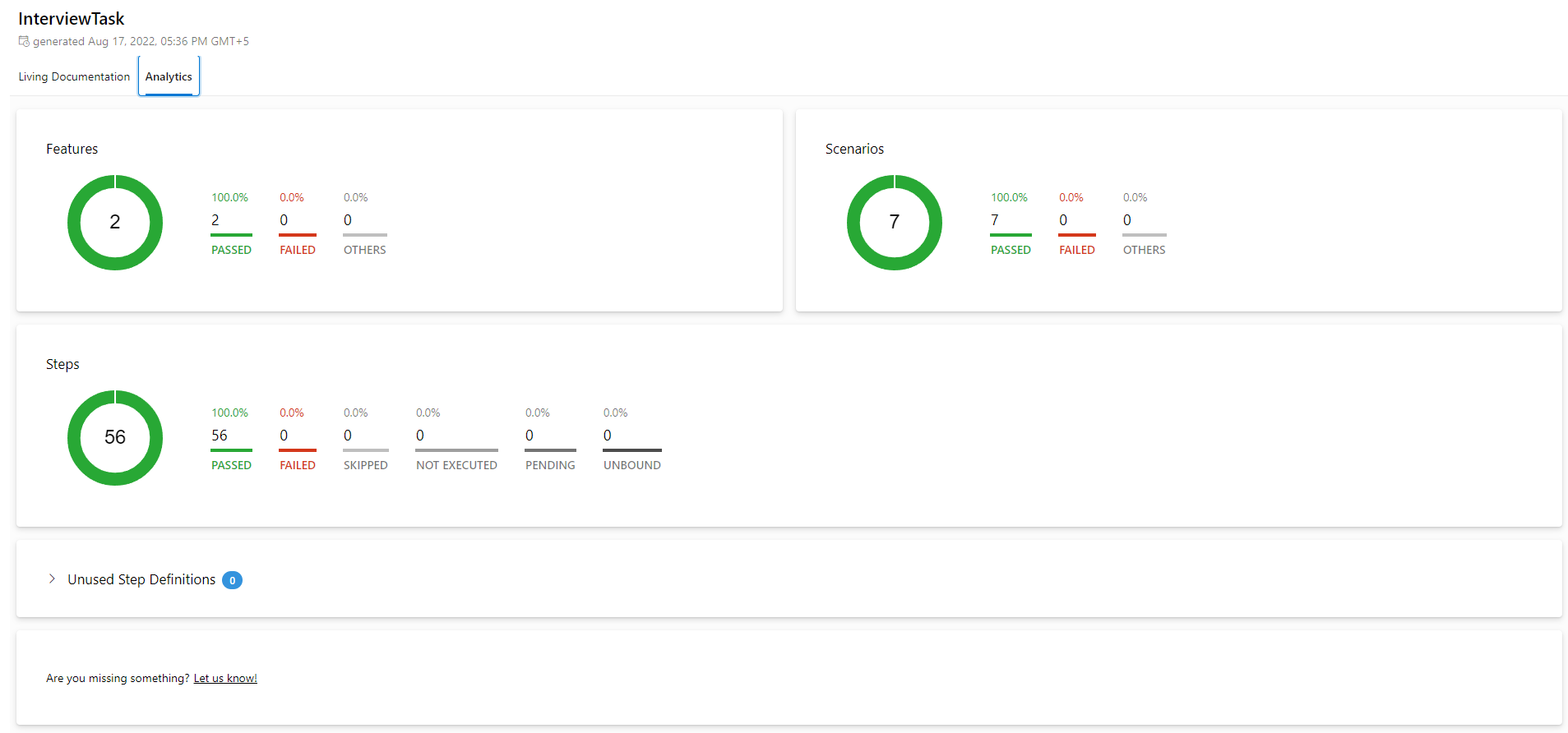
## Specflow plus Living Doc

SpecFlow+LivingDoc is a set of tools that allows you to share and collaborate on Gherkin Feature Files with stakeholders who may not be familiar with developer tools and for technical team as well.

Living Documentation:

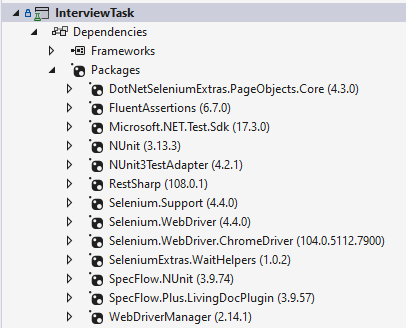


Analytics:



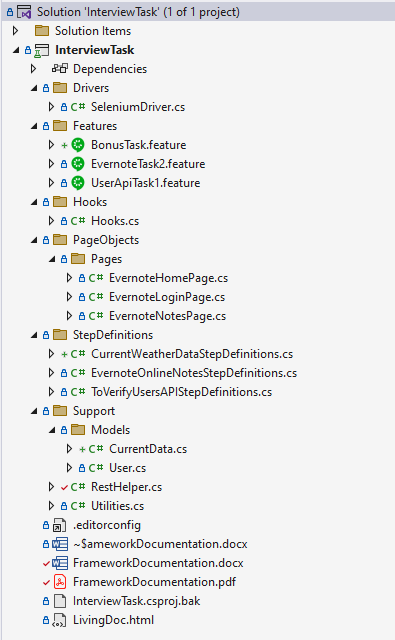
# Specflow Framework Packages

* **Selenium.Webdriver.ChromeDriver:** Install Chrome Driver for Selenium WebDriver into your Unit Test Project, "chromedriver(.exe)" is copied to the bin folder from the package folder when the build process NuGet package restoring ready, and no need to commit "chromedriver(.exe)" binary into source code control repository. (This package automatically downloads the latest version of chrome driver so we don’t need to use webdriver manager).
* **SeleniumExtras.PageObjects:** Page Factory is a class provided by Selenium WebDriver to support Page Object Design patterns and lazy initialization. In Page Factory, testers use FindsBy annotation. The initElements method is used to initialize web elements. FindsBy: An annotation used in Page Factory to locate and declare web elements using different locators.
* **NUnit**: NUnit features a fluent assert syntax, parameterized, generic, and theory tests and is user-extensible.This package includes the NUnit 3 framework assembly, which is referenced by your tests. You will need to install version 3 of the nunit3-console program or a third-party runner that supports NUnit 3 to execute tests. Runners intended for use with NUnit 2.x will not run NUnit 3 tests correctly.
* **RestSharp:** I’ve used RestSharp for API testing sinceRestSharp is probably the most popular HTTP client library for .NET. Featuring automatic serialization and deserialization, request and response type detection, variety of authentications and other useful features.



# Project Structure

Page object model design pattern is used in this framework and the complete structure of this framework is shown below.



# Assignment Detail

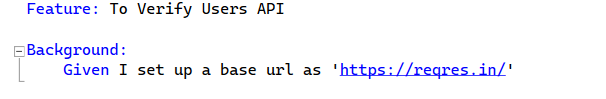
As QA Automation Engineer and as Solution Architect I’ve designed this Framework which can hold Web Browser Tests as well as API’s Test’s. So to achieve this target I’ve used above mentioned technology stack (Selenium, RestSharp with Specflow). Since as per requirement I’ve to implement it using BDD framework so every task has its own feature file and based on that feature we’ve different scenario’s implemented, let’s have a brief Introduction of each Scenario of both the Feature file.

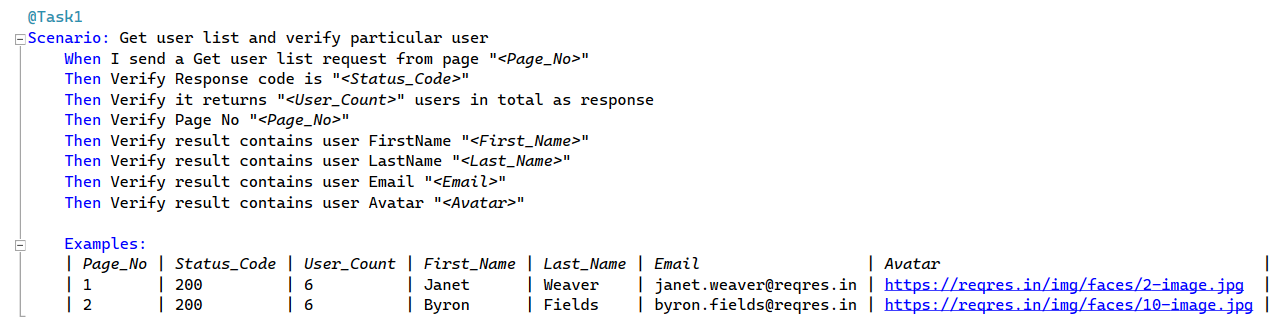
# Solution

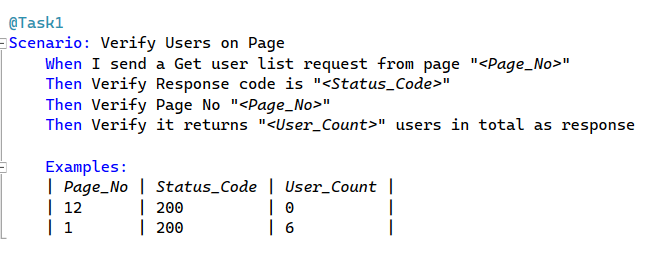
* I have created one feature file having two scenario’s each Scenario has a Background where I’m setting up the base URL. Steps are defined and implemented in a way that every step has reusable capabilities and it can be run through data driven approach means any one can change or add the data to test more scenarios.
* I have used Tags approach to deal different conditions, e.g. we can group them and even if you discussed Task2 so there we have to initialize chrome browser and that we don’t need in Task1, so created hooks with Tags means when Task2 trigger than only browser initializes else no need.
* Specflow Living Document and Detail Analysis also available view full execution report and also you can preview scenario behavior.

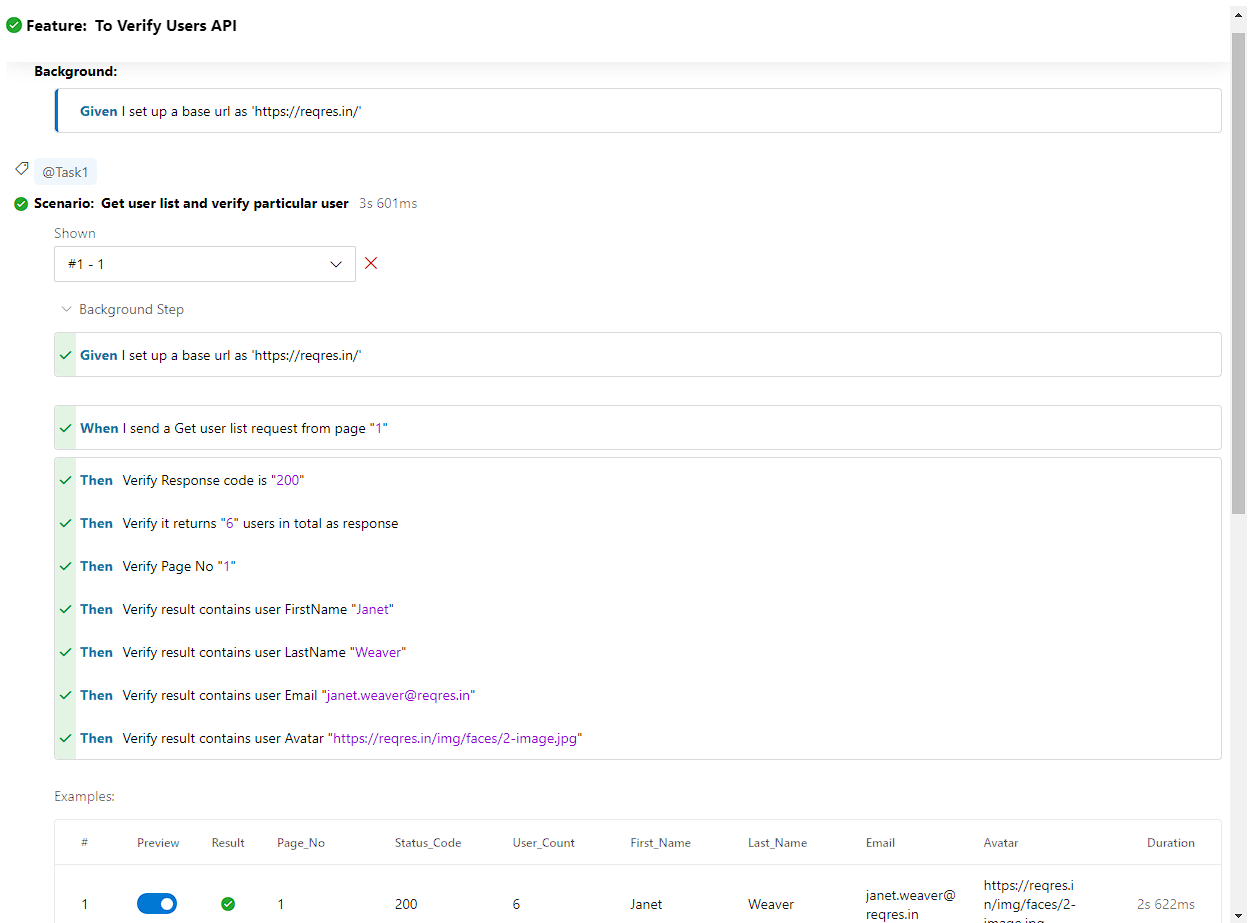
## Task 1

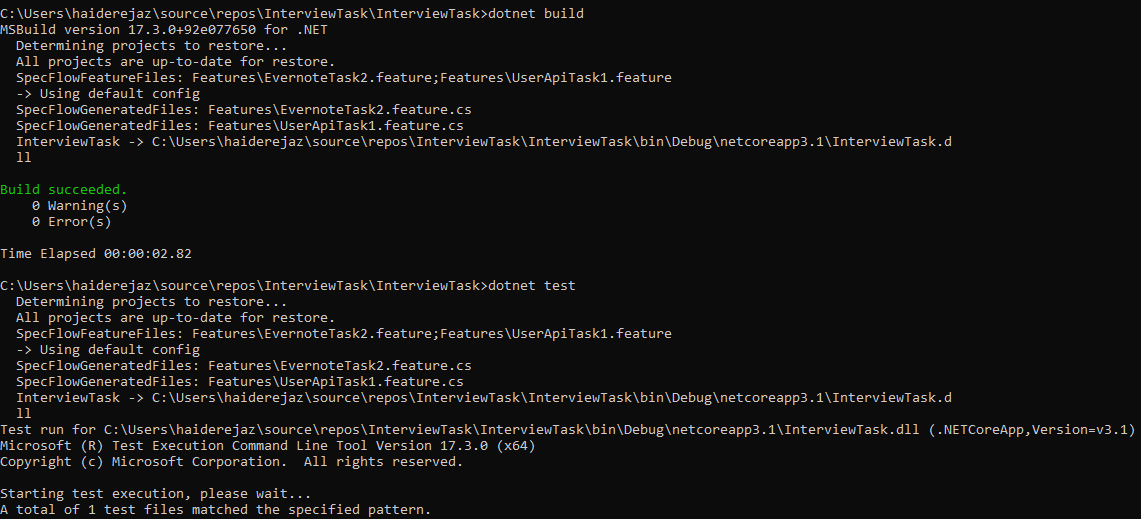
“*As a QA engineer you are requested to test a particular API method with the following URL https://reqres.in/api/users?page=1. During the code analysis, 3 main scenarios to be tested were identified. Develop a .Net Test Framework to implement the below tests (ideally using a BDD language such as Gherkin).*”

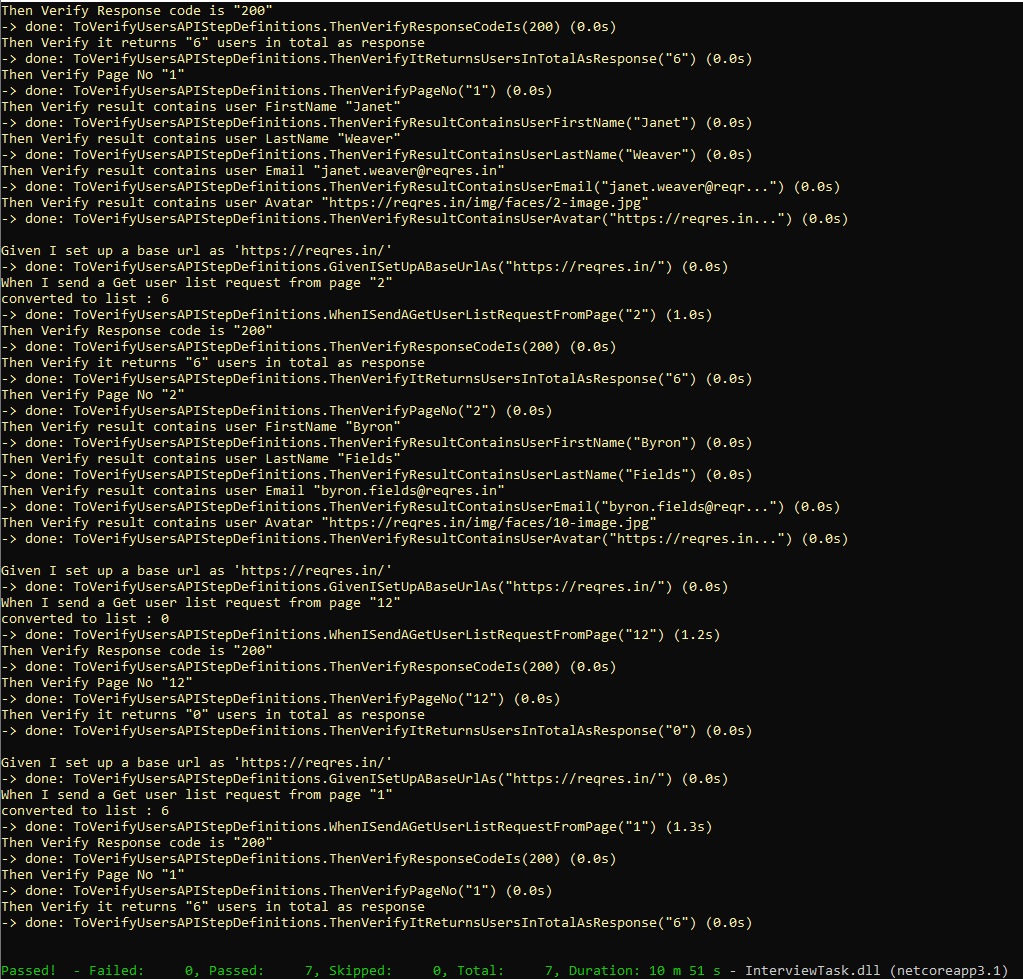


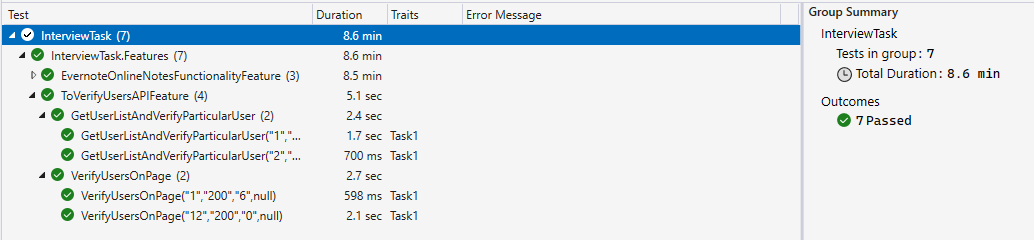












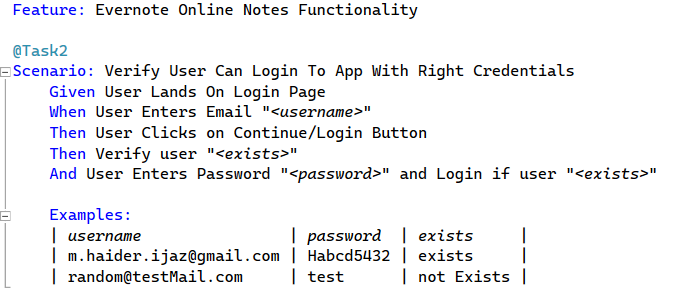
## Task 2

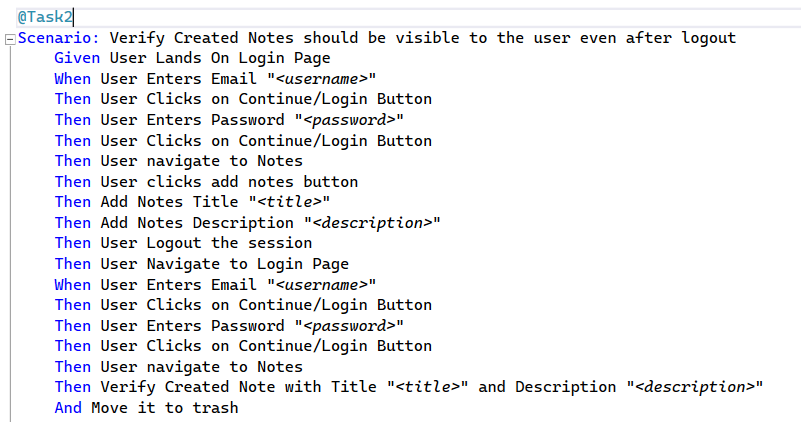
“*Consider that you are working with a well-known online note taking company, https://evernote.com/, as QA Test Automation Engineer. You are given the task to develop a test automation framework using Selenium, together with a BDD language such as gherkin to implement UI automated tests on different core functionalities of the Website. These tests should cover:*

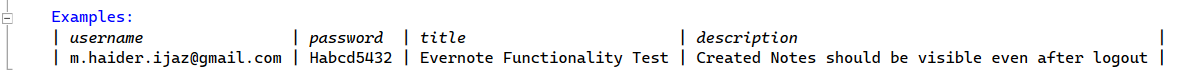
*1. Unsuccessful login using email and assert that the proper error messages are returned*

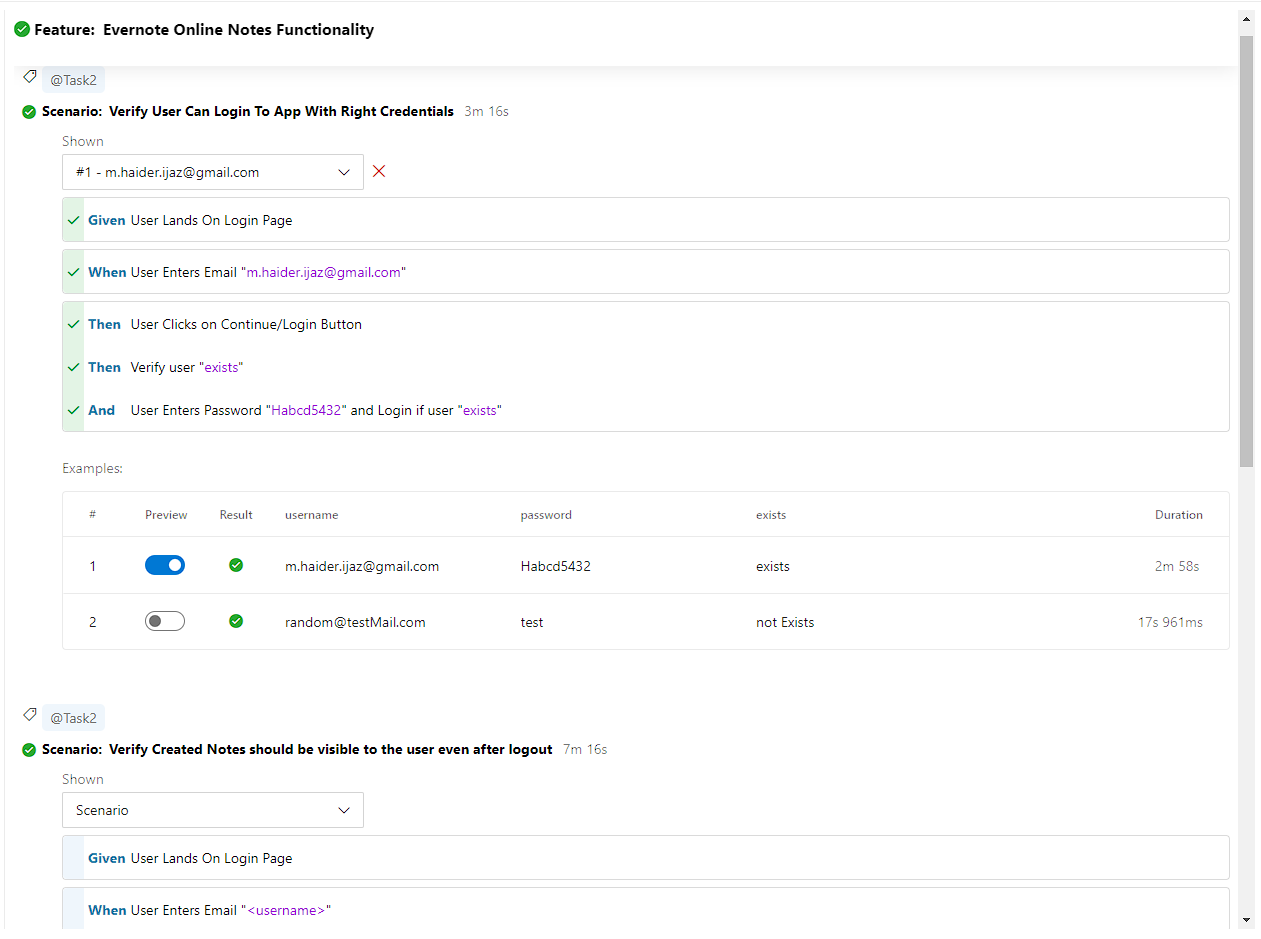
*2. Successful login using email and assert that the user is granted access to the site*

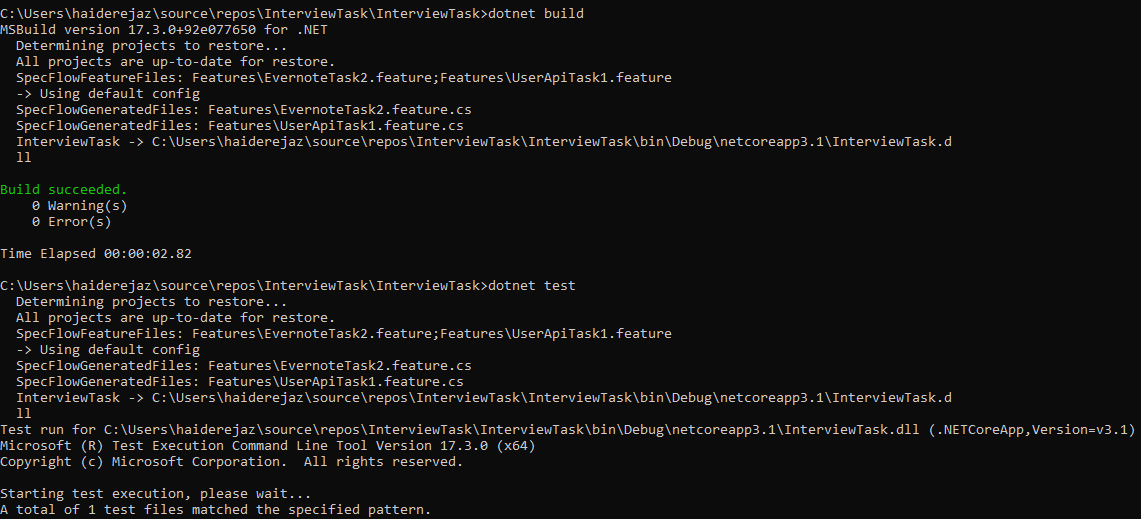
*3. Login and write a note followed by a logout. Finally, login again and make sure you can open and view the note created previously. ”*

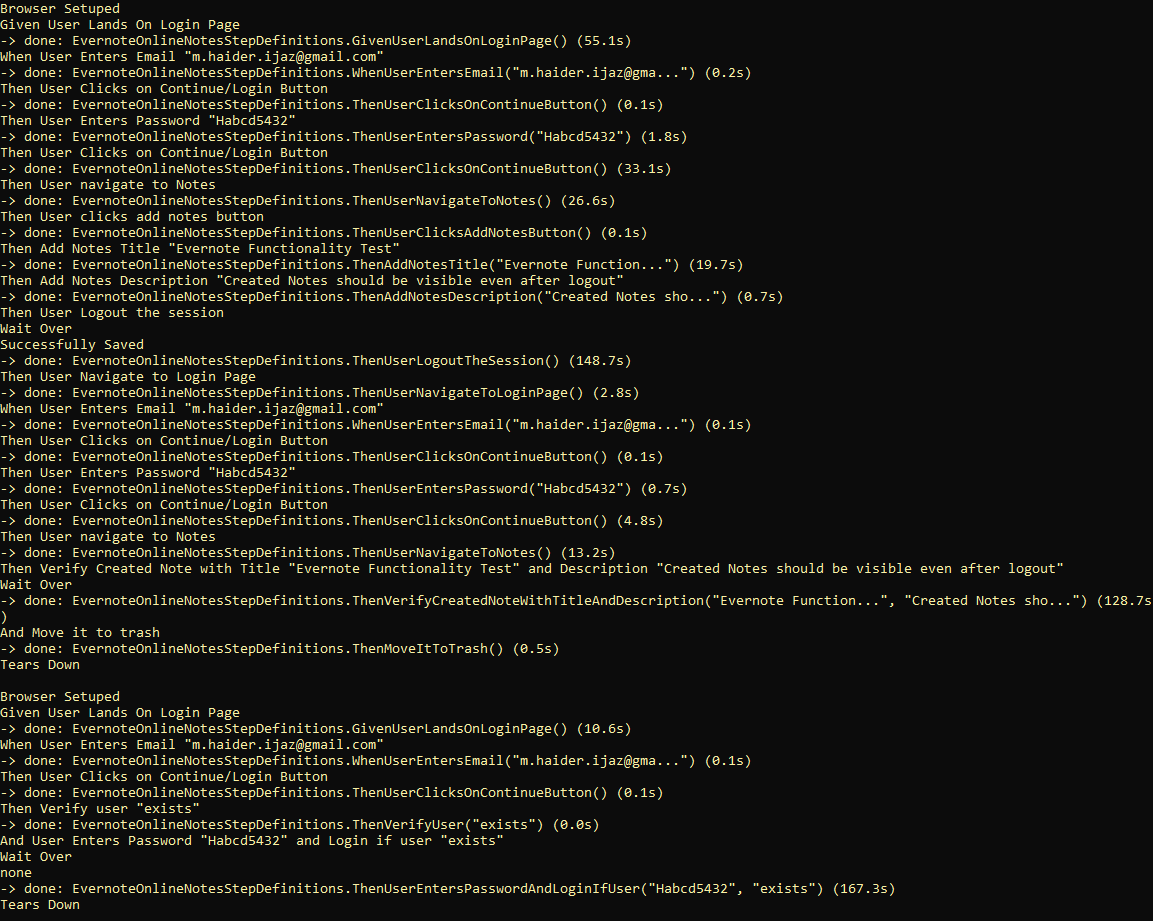


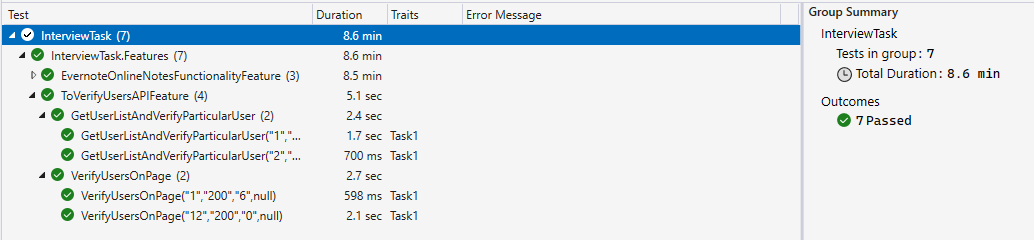












## Bonus Requirement

“*Following the foundations set out in the previous section you are invited to visit* <https://openweathermap.org/api> *and design several different test cases. Each test case you design must be documented and implemented in a testing framework of your choice, be sure to justify the reason for testing each of the chosen scenarios.”*

# Solution

I’ve covered few basic test cases which any QA Engineer has to have to test before moving to next, and those are very needful in this current era where cyber-attacks count growing evert minute. All these test cases I’ve have implemented/handled in the same framework so you can find and Feature File named “**BonusTask.feature**”.

Cases:

* Verify API Authorization with correct and incorrect App Id
* Verify Status Codes and Status Messages set for users (Success/Failed)
* Verify Required Parameter with correct and incorrect data
* Verify Correct Data as per Param Criteria in Response and should not null

