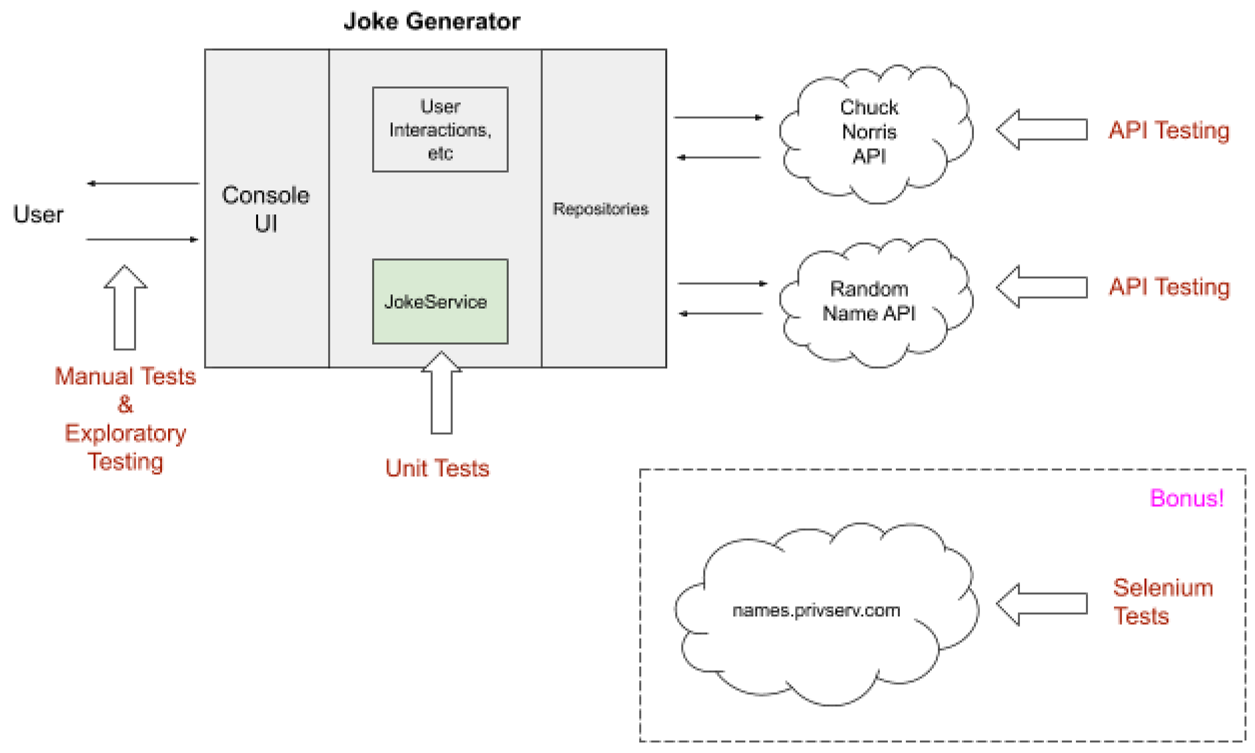
Table of Contents

Testing Strategy

I have approached the project as following:



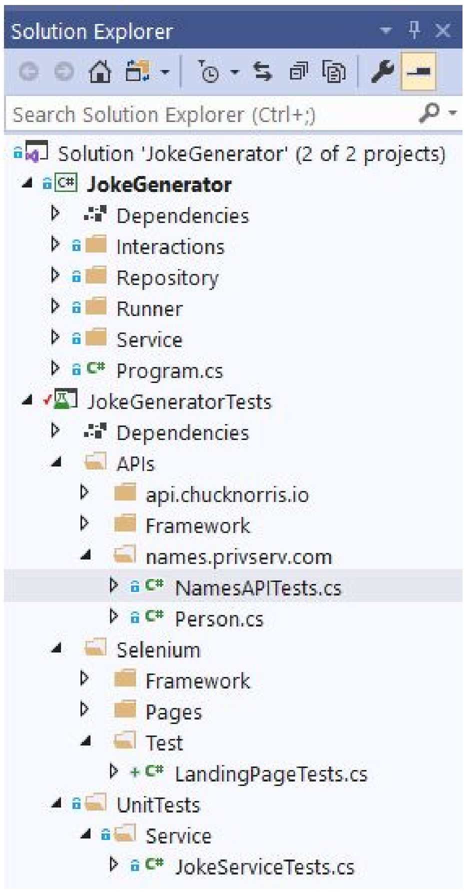
1. Unit tests for the business logic (eg: JokeService.cs)
2. API tests for interaction with the API endpoints.

3. Manual and Exploratory testing for testing user interaction through console and edge conditions.
4. Additionally, I wrote a small test project to demonstrate Selenium based testing using the `names.privserv.com` website.

Project Structure and Dependencies

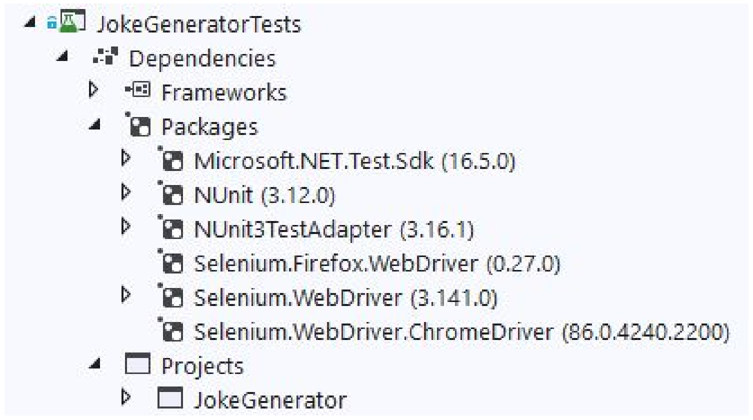
`JokeGeneratorTests` is the test project which contains all automated tests, and is part of the `JokeGenerator` solution. The tests are organized by categories at project root:

1. `APIs`: contains Framework and tests related to testing APIs.
2. `UnitTests`: contains unit tests. Directory structure mirrors the Project Directory Structure and test names follow the convention: “`$(FileName)Tests`”. E.g. Tests for `JokeService.cs` live in `UnitTests/Service/JokeServiceTests.cs`
3. `Selenium`: contains Framework + tests for web app (www.names.priserv.com)



```
Solution 'JokeGenerator' (2 of 2 projects)
  JokeGenerator
    Dependencies
    Interactions
    Repository
    Runner
    Service
    Program.cs
  JokeGeneratorTests
    Dependencies
    APIs
      api.chucknorris.io
      Framework
      names.privserv.com
        NamesAPITests.cs
        Person.cs
    Selenium
      Framework
      Pages
      Test
        LandingPageTests.cs
    UnitTests
      Service
        JokeServiceTests.cs
```

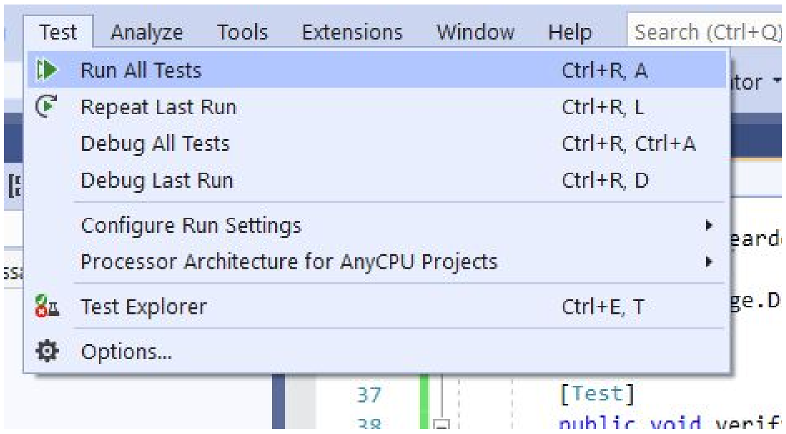
The test project has following Dependencies:


```
JokeGeneratorTests
  Dependencies
    Frameworks
    Packages
      Microsoft.NET.Test.Sdk (16.5.0)
      NUnit (3.12.0)
      NUnit3TestAdapter (3.16.1)
      Selenium.Firefox.WebDriver (0.27.0)
      Selenium.WebDriver (3.141.0)
      Selenium.WebDriver.ChromeDriver (86.0.4240.2200)
    Projects
      JokeGenerator
```

Additionally, the user needs to have chrome and firefox browsers installed on their machine (latest versions should do).

How to run tests

I used Visual Studio 2019 to develop the tests. Tests can be run from within the IDE by opening the solution and

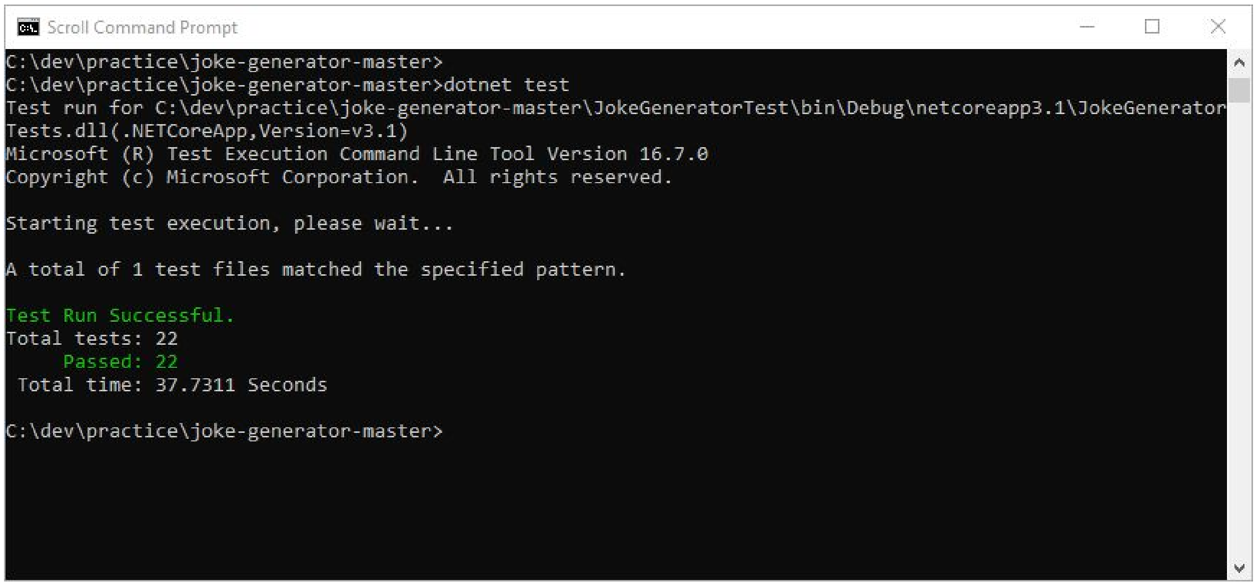


This should automatically fetch all the dependencies, build the solution and run the tests and display a report in the Test Explorer.

Alternatively, tests can be run from the command prompt. Open command prompt, navigate to project root and run:

```
dotnet test
```

This should install nuget packages, build all projects, run the tests and display a report:



```
C:\dev\practice\joke-generator-master>
C:\dev\practice\joke-generator-master>dotnet test
Test run for C:\dev\practice\joke-generator-master\JokeGeneratorTest\bin\Debug\netcoreapp3.1\JokeGenerator
Tests.dll(.NETCoreApp,Version=v3.1)
Microsoft (R) Test Execution Command Line Tool Version 16.7.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 22
     Passed: 22
 Total time: 37.7311 Seconds

C:\dev\practice\joke-generator-master>
```

List of bugs

1. **JokeService.cs**
   a. UseCase: Handling Invalid name inputs -

Case	**Old Behavior**	**Behavior after bugfix**
White space	Name in joke was replaced with white spaces	Original jokes are returned as is
Empty	Name in joke was replaced with empty string	Original jokes are returned as is

   b. UseCase: Handling invalid joke inputs -

Case	**Old Behaviour**	**Behaviour after bugfix**
Reference to Jokes passed null	NullReferenceException	ArgumentNullException
Jokes containing a null entry	ArgumentNullException was thrown from within regex.Replace	Entry is omitted from the updatedJokes collection
Jokes containing an empty string as joke	updatedJokes contained empty string	Entry is omitted from the updatedJokes collection

c. UseCase: Edge conditions

Case	**Old Behaviour**	**Behaviour after bugfix**
Handling scenario when joke contains text similar to the name Chuck Norris. (e.g. De**Chuck Norris**issm)	The text would get replaced with the new name e.g. De**Anna**issm	Name replacement doesn't happen unless the pattern match is perfect.

2. Exploratory testing- interaction with UI

a. UseCase: Ability to render non-english names (e.g. Chinese names)

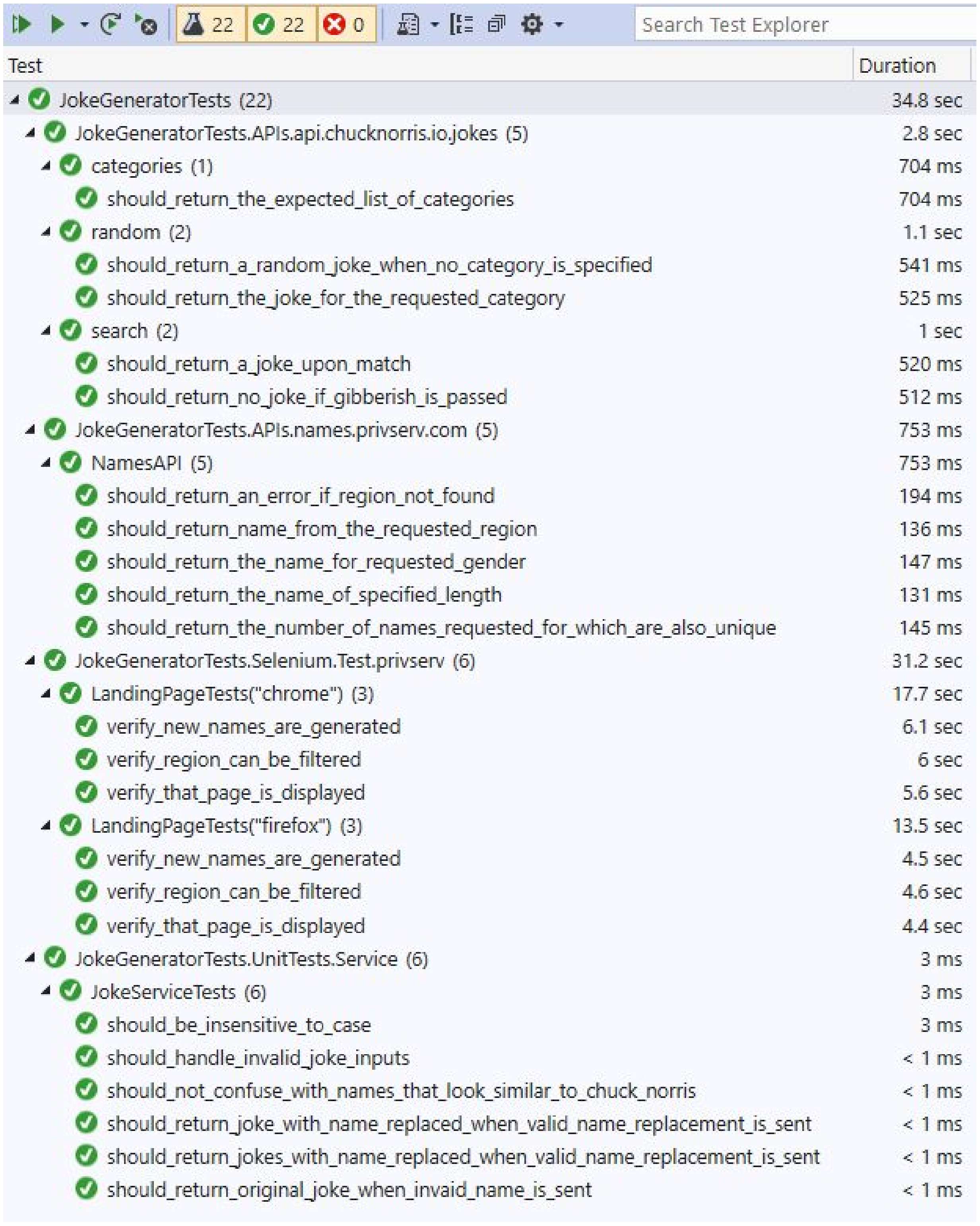
Case	**Old Behaviour**	**Behaviour after bugfix**
Rendering Chinese names	Special characters were displayed on the console (e.g. Name Chosen: ????ß?? ??????µ??)	The correct name in Chinese is displayed (e.g. Name Chosen: 崔豪) Note: The terminal font should be configured to NSimSun.

b. UseCase: Exception Handling

Case	**Old Behaviour**	**Behaviour after bugfix**
User is not connected to the internet	An unhandled error crashed the application	User understandable error is printed and application terminates gracefully.

List of Tests

Automated Tests



```
Test                                                                                  Duration
JokeGeneratorTests (22)                                                               34.8 sec
  JokeGeneratorTests.APIs.api.chucknorris.io.jokes (5)                                2.8 sec
    categories (1)                                                                    704 ms
      should_return_the_expected_list_of_categories                                   704 ms
    random (2)                                                                        1.1 sec
      should_return_a_random_joke_when_no_category_is_specified                       541 ms
      should_return_the_joke_for_the_requested_category                               525 ms
    search (2)                                                                        1 sec
      should_return_a_joke_upon_match                                                 520 ms
      should_return_no_joke_if_gibberish_is_passed                                    512 ms
  JokeGeneratorTests.APIs.names.privserv.com (5)                                      753 ms
    NamesAPI (5)                                                                      753 ms
      should_return_an_error_if_region_not_found                                      194 ms
      should_return_name_from_the_requested_region                                    136 ms
      should_return_the_name_for_requested_gender                                     147 ms
      should_return_the_name_of_specified_length                                      131 ms
      should_return_the_number_of_names_requested_for_which_are_also_unique           145 ms
  JokeGeneratorTests.Selenium.Test.privserv (6)                                       31.2 sec
    LandingPageTests("chrome") (3)                                                    17.7 sec
      verify_new_names_are_generated                                                  6.1 sec
      verify_region_can_be_filtered                                                   6 sec
      verify_that_page_is_displayed                                                   5.6 sec
    LandingPageTests("firefox") (3)                                                   13.5 sec
      verify_new_names_are_generated                                                  4.5 sec
      verify_region_can_be_filtered                                                   4.6 sec
      verify_that_page_is_displayed                                                   4.4 sec
  JokeGeneratorTests.UnitTests.Service (6)                                            3 ms
    JokeServiceTests (6)                                                              3 ms
      should_be_insensitive_to_case                                                   3 ms
      should_handle_invalid_joke_inputs                                               < 1 ms
      should_not_confuse_with_names_that_look_similar_to_chuck_norris                 < 1 ms
      should_return_joke_with_name_replaced_when_valid_name_replacement_is_sent       < 1 ms
      should_return_jokes_with_name_replaced_when_valid_name_replacement_is_sent      < 1 ms
      should_return_original_joke_when_invaid_name_is_sent                            < 1 ms
```

Manual Tests

1. To verify the software behaviour against all valid inputs.
2. To verify the software for invalid inputs, different from what is expected of the user to enter (beyond boundaries, special characters, etc)
3. To verify a random name is displayed when a user asks for one.
4. To verify that the name Chuck Norris should be replaced if a user chooses a random name.
5. To verify that all the categories are displayed correctly.
6. To verify that the chosen category is shown after the user selects one.
7. To verify that the correct category joke is displayed for a chosen category.
8. To verify the number of jokes displayed are the same as selected by the user.
9. To verify name encoding is done properly.
10. To verify that a user can exit from the application

Observations

This section highlights some observations related to user experience:

1. Inputs are accepted in lower case only. For e.g. "Press x to exit the program", does not consider X as a valid input.
2. The user doesn't have an option to exit in the middle (maybe `ctrl+c` is fine?)
3. A few joke categories e.g. fashion, history, music, religion, science, travel etc. are unable to retrieve all the jokes requested for.