Task to do

=====================================================================================

Create a service that searches a configured directory and subdirectories for files containing a list of words that are provided to the service.

The service should:

• Present a simple rest API to allow perform a search

• Allow searching with multiple words

• Multiple word searches should return files that match all words

• Should perform whole word matches

• No fuzzy search required

• Return results as quickly as possible

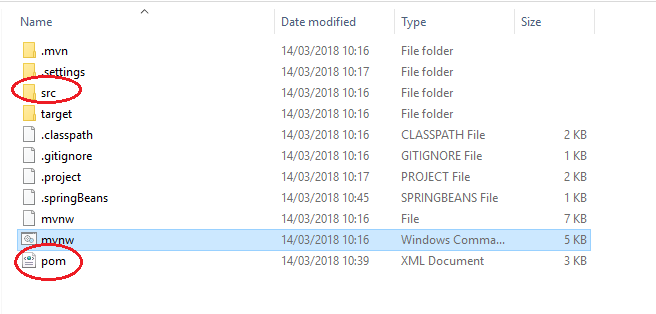
• Code should be production ready code to a professional standard

=====================================================================================

Technology used:

* Spring BOOT
* Java version: 1.8
* Build tool: Maven

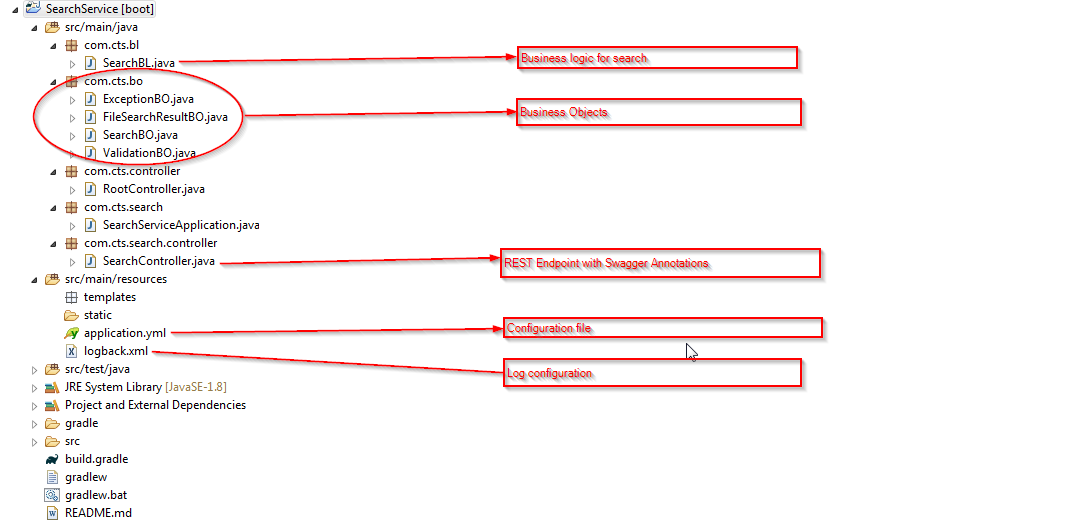
Code Directory Structure



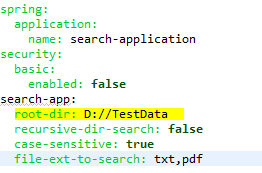
Src code folder contains all the source code

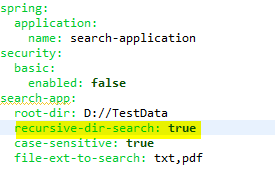
Pom.xml is the build file

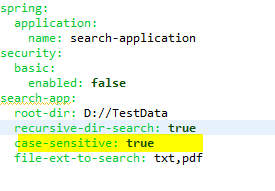
Main components



Some Important Parameters

* The default root path need to be set into application.yml file. If no input given for root path as part of request body, application will take the default root path from application.yml file.
* Setting the default base directory to search: Open the application.yml file and change the following.
* Setting the recursive directory search: By default the application only search for all files under the root directory and it’s subdirectory. To make it only limited to root directory change the following parameter to false



* Setting case sensitive or insensitive search: Change the following parameter to false to make the search case insensitive. 
* By default txt files and pdf files will be picked but search logic is implemented only for txt file.

How to run:

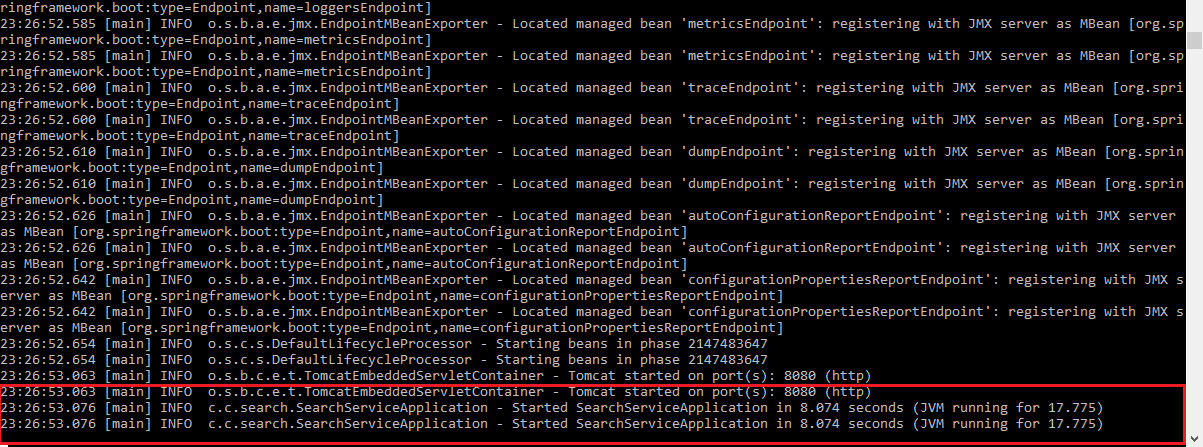
Pre-Requisite: JDK 1.8 and Maven needs to be installed. JAVA\_HOME & MAVEN\_HOME need to be set up.

Go to the directory containing the pom.xml from command prompt and issue the command.

mvn spring-boot:run

This will download all required files and build the application and fire the application after successful build.

You will get similar message as below when it is running.



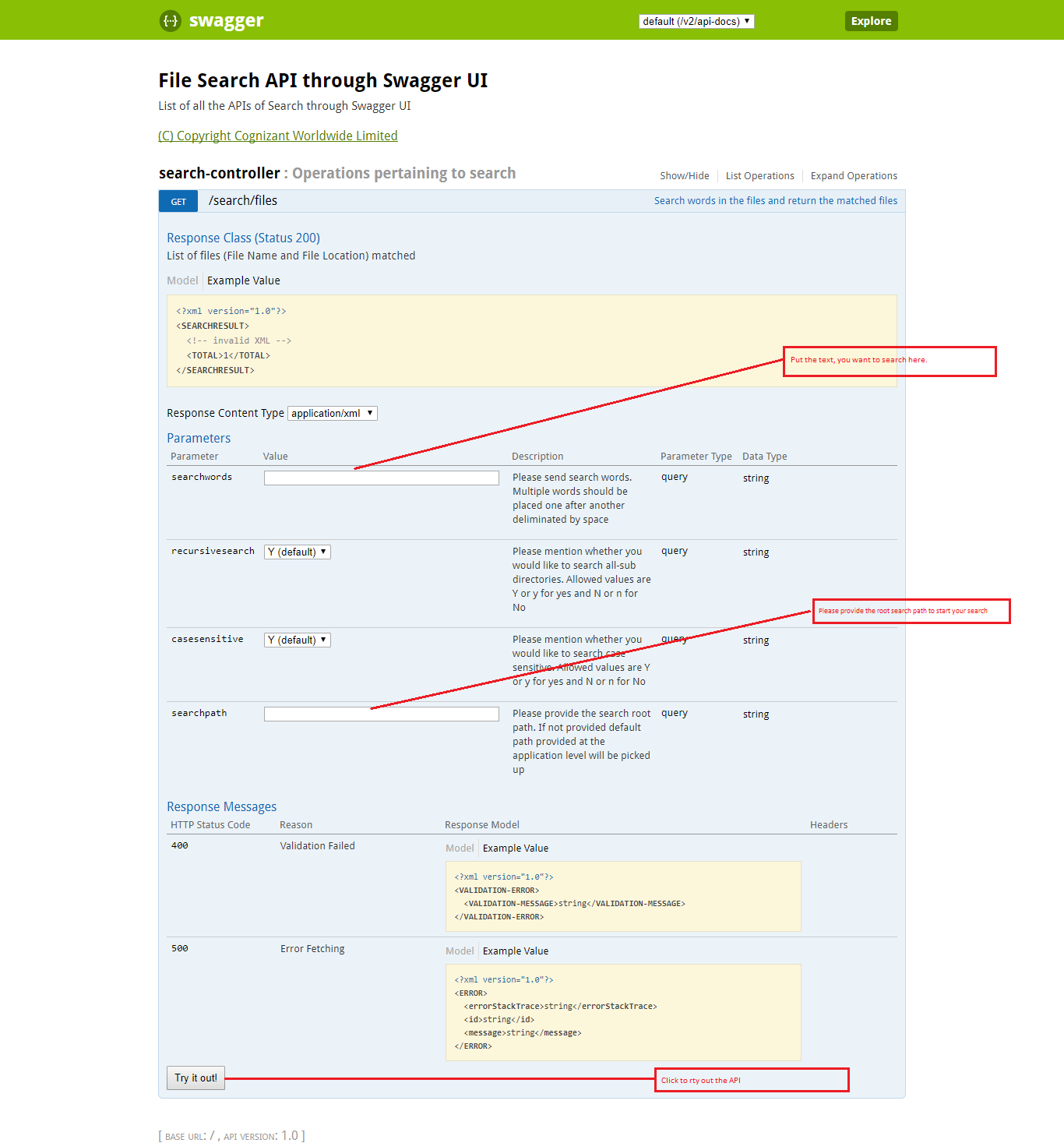
Now open the browser and give the following URL.

<http://localhost:8080/swagger-ui.html>

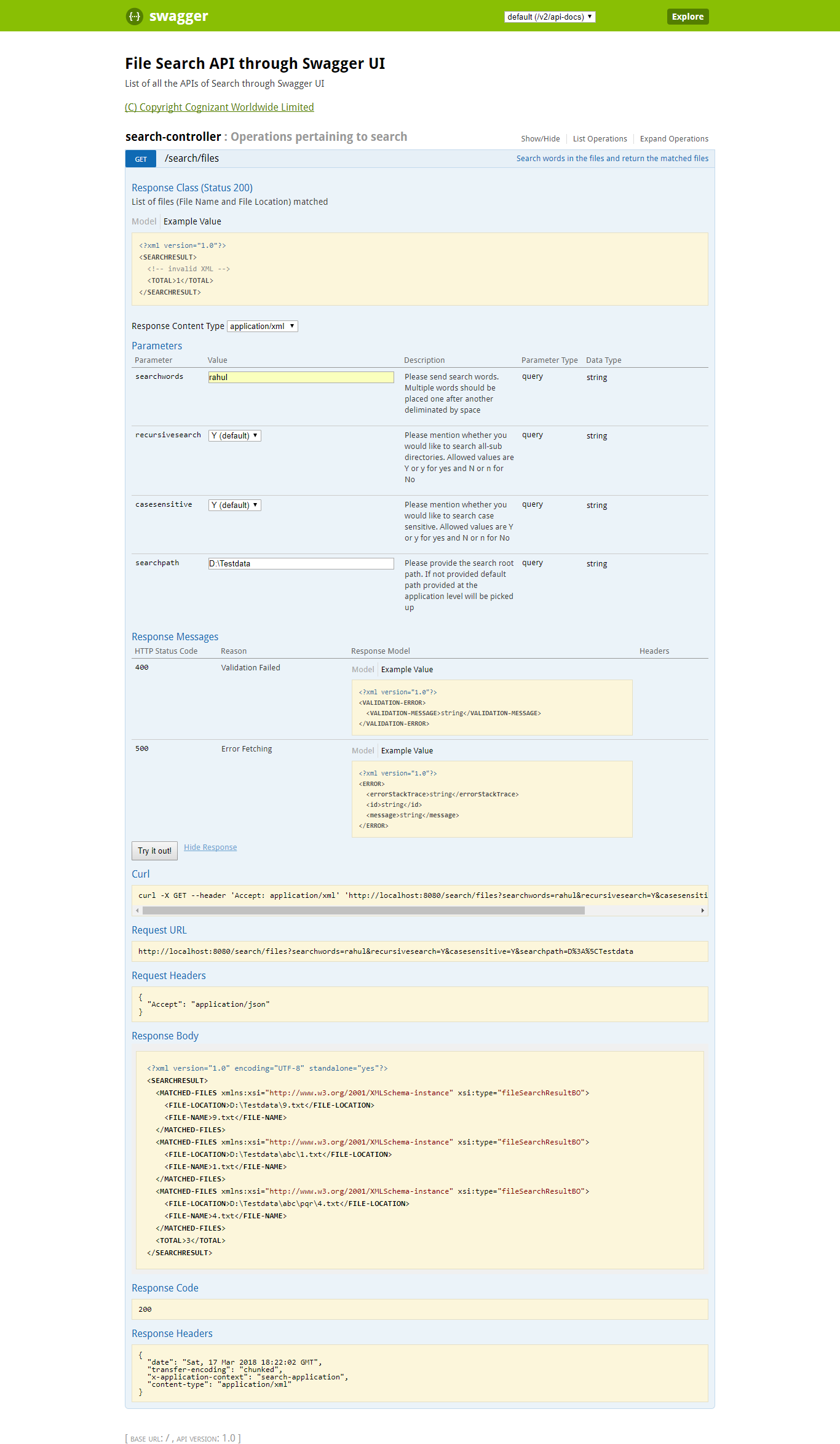
You will get a page of Swagger which gives a detailed specification of API with provision to test it out.

Sample URL to test out the service (change the highlighted word with the one you would like to search.

<http://localhost:8080/search/files?searchwords=rahul&recursivesearch=Y&casesensitive=Y>

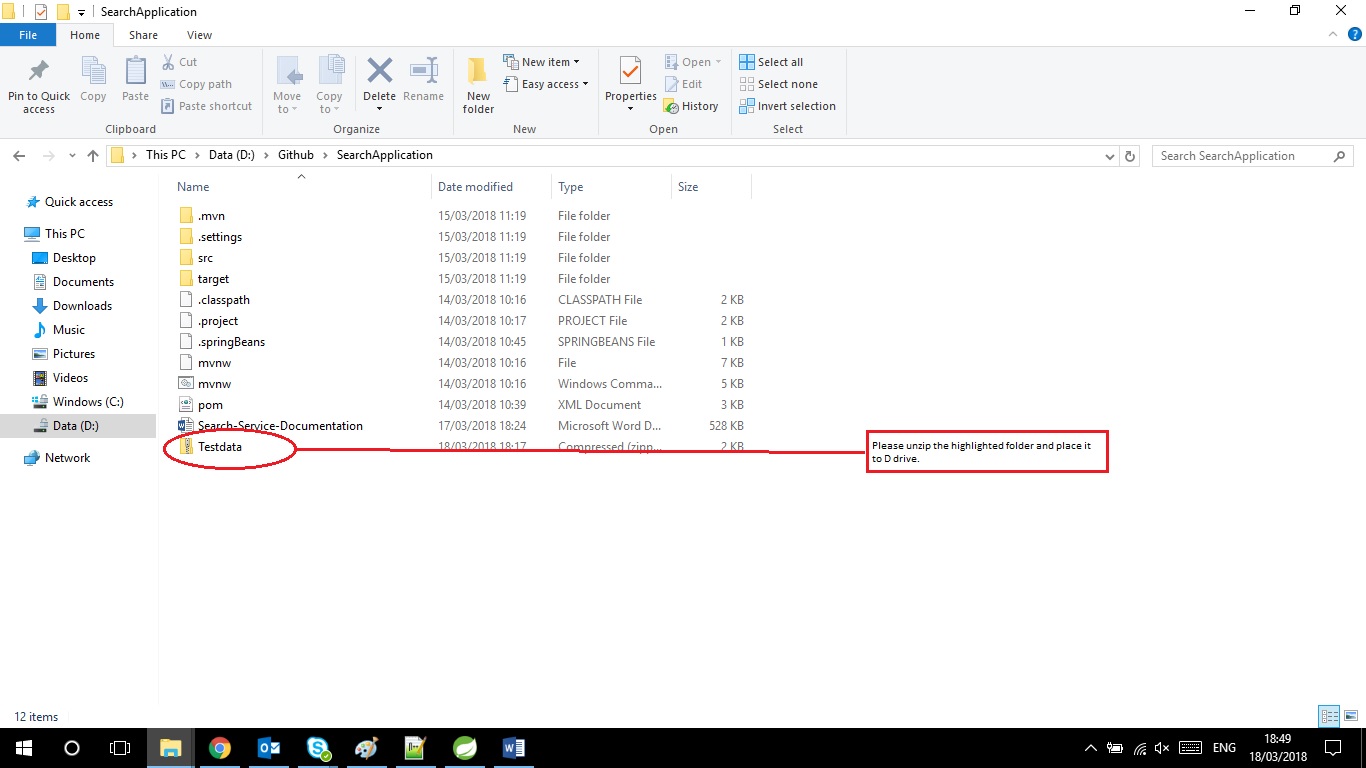


A sample run can be as follows.



Unit Test Case Execution: -

There are six different unit test cases has been written with almost 100% code coverage. To execute the unit test case, the data need to setup properly. Please unzip the below highlighted folder and place it into your D drive. Then run the unit test cases. Otherwise all the test case will fail.



Test Cases

1. **singleWordSearchCaseSensitive() :-**

This application is having a very unique functionality to make your word search case sensitive. This unit test case will check, whether that functionality is working properly or not when you are searching with single keyword.

1. **singleWordSearchCaseInSensitive() :-**

This application has ability to search case insensitive word also. This particular unit test case will help you to check that functionality for a single word search.

1. **singleWordSearchNonRecursive() :-**

Through this application you can perform a recursive or non-recursive search. If you want to restrict your search into your base folder location only, then you need to perform a non-recursive search. This particular test case will help you to understand whether this functionality of the application is working properly or not for a single word search.

1. **multiWordSearchCaseSensitive() :-**

This particular test case will help us to check the case sensitive functionality for multi word search. It will return a positive result only if all those case sensitive words are present in files together.

1. **multiWordSearchCaseInSensitive() :-**

This particular test case will help us to check the case insensitive functionality for multi word search. It will return a positive result only if all the keywords placed for search is case insensitive. Those words should be present in files together.

1. **multiWordSearchNonRecursive() :-**

To check the functionality of non-recursive search for multiple words we need to execute this test case. It will return a positive result only if all the keywords placed for search are present in base folder location. Those words should be present in files together.