GITHUB BI TOOL USER MANUAL AND DOCUMENTATION

GitHub assignment

Abstract

This document is to let users know on how to use the tool along with documentation on the code and unit testing part.

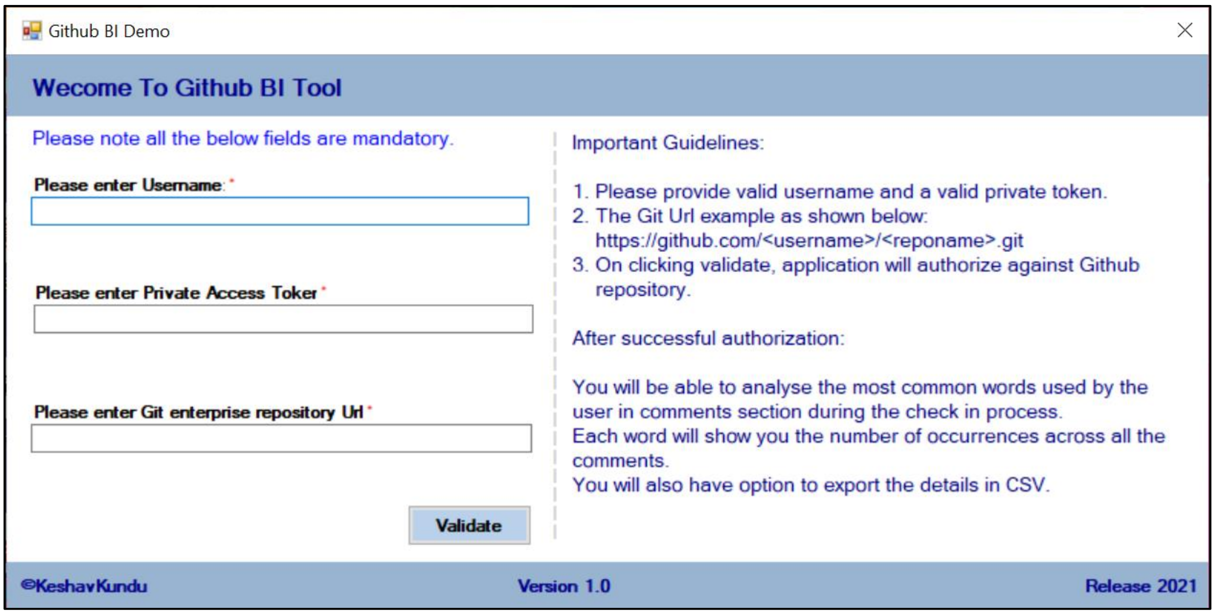
Keshav Kundu

Keshav.kundu88@gmail.com

Contents

The user interface is shown below along with guidelines on the right

Please note the application is built on target .NET Framework 4.6.1


Github BI Demo

Wecome To Github BI Tool

Please note all the below fields are mandatory.

Please enter Username *

Please enter Private Access Toker *

Please enter Git enterprise repository Url *

Validate

Important Guidelines:

1. Please provide valid username and a valid private token.
2. The Git Url example as shown below: https://github.com/<username>/<reponame>.git
3. On clicking validate, application will authorize against Github repository.

After successful authorization:

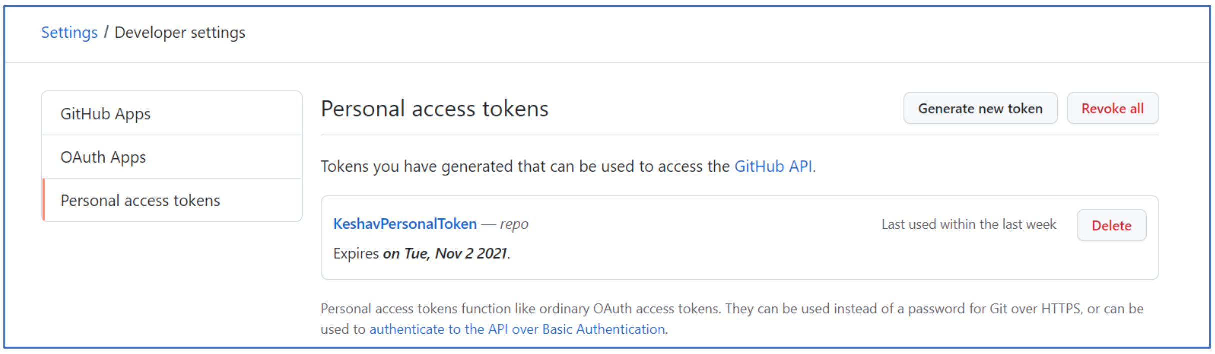
You will be able to analyse the most common words used by the user in comments section during the check in process.
Each word will show you the number of occurrences across all the comments.
You will also have option to export the details in CSV.

©KeshavKundu Version 1.0 Release 2021

Accessing Git Hub for getting Token and URL

Step1: Creating personal access token from GitHub.

User need to create a personal access token in GitHub -> Setting -> Developer Settings -> Personal Access token


Settings / Developer settings

GitHub Apps
OAuth Apps
Personal access tokens

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the GitHub API.

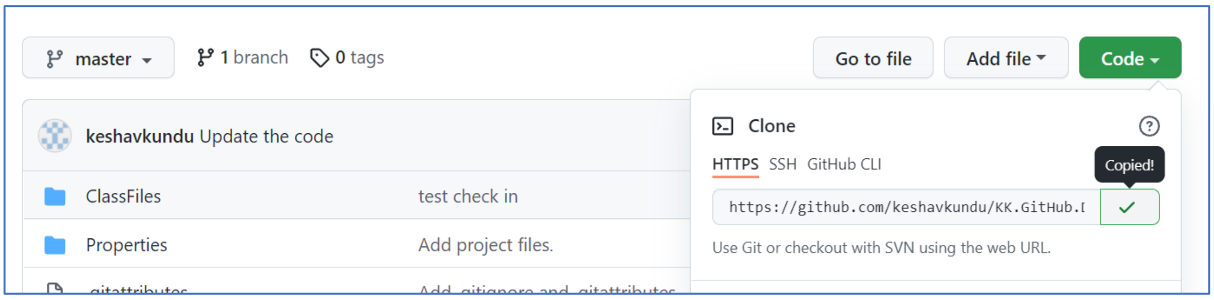
KeshavPersonalToken — *repo* Last used within the last week Delete

Expires *on Tue, Nov 2 2021.*

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Step2: Accessing the Git Repository URL

User needs to copy the Git hub as shown below.


master 1 branch 0 tags Go to file Add file Code

keshavkundu Update the code

ClassFiles	test check in
Properties	Add project files.

Clone

HTTPS SSH GitHub CLI

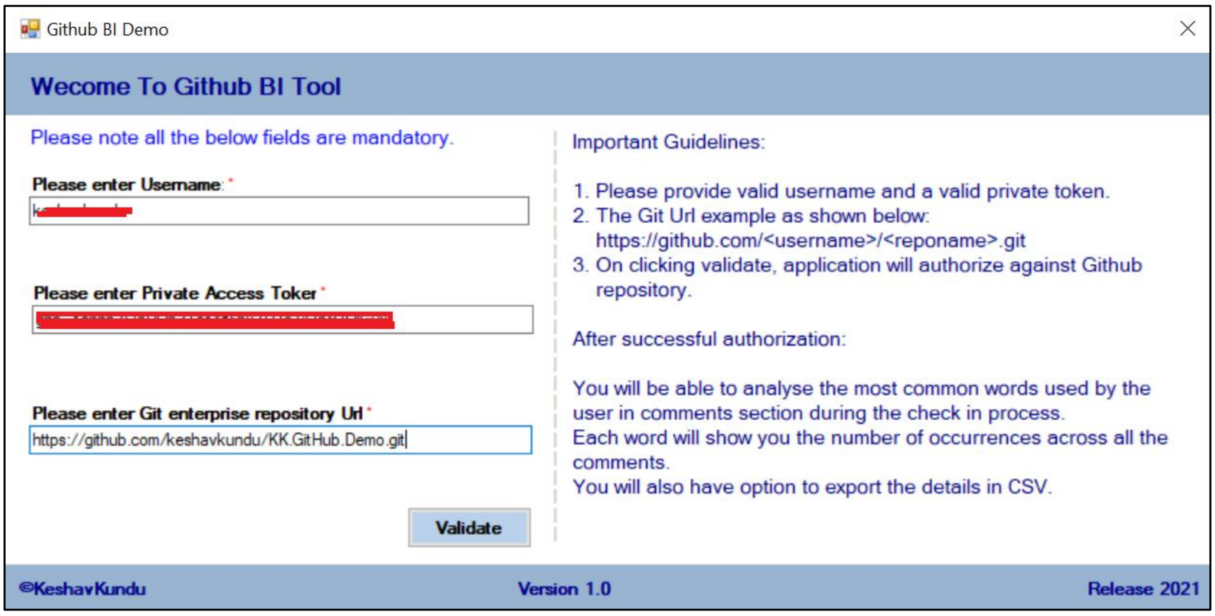
Copied!

https://github.com/keshavkundu/KK.GitHub.I

Use Git or checkout with SVN using the web URL.

Testing the application UI

Step1: Enter the details in the screen and click validate.



Github BI Demo

Wecome To Github BI Tool

Please note all the below fields are mandatory.

Please enter Username: *
[redacted]

Please enter Private Access Toker *
[redacted]

Please enter Git enterprise repository Url *
https://github.com/keshavkundu/KK.GitHub.Demo.git

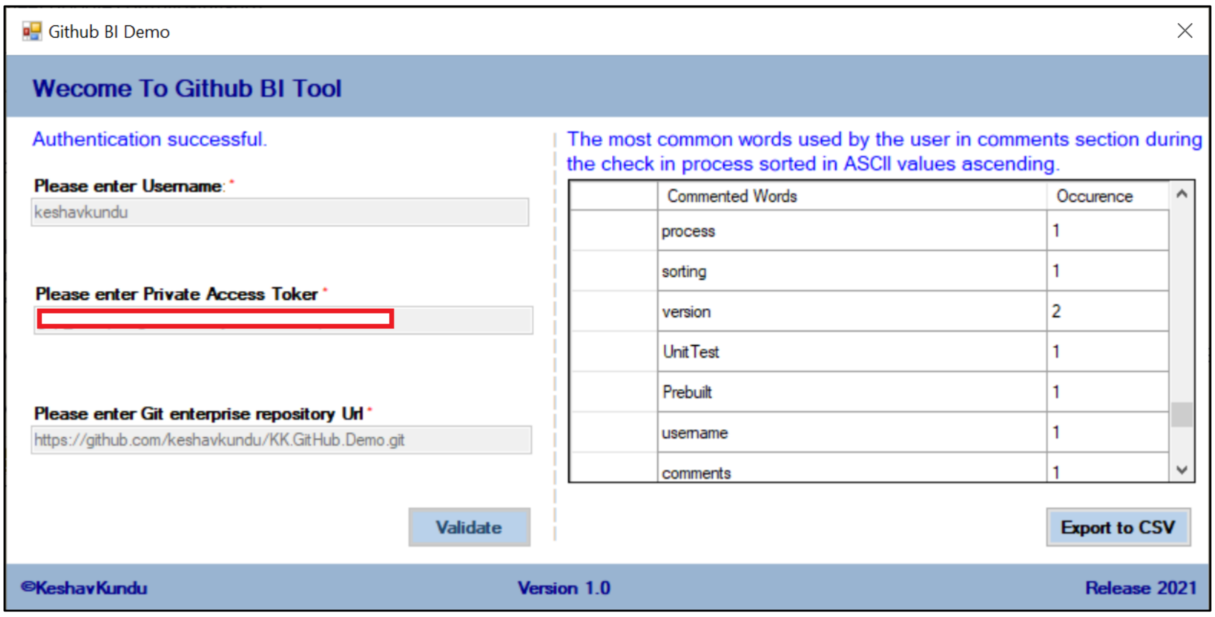
Important Guidelines:

1. Please provide valid username and a valid private token.
2. The Git Url example as shown below:
   https://github.com/<username>/<reponame>.git
3. On clicking validate, application will authorize against Github repository.

After successful authorization:

You will be able to analyse the most common words used by the user in comments section during the check in process.
Each word will show you the number of occurrences across all the comments.
You will also have option to export the details in CSV.

Step2: After successful authentication you will be presented with a grid on right



Github BI Demo

Wecome To Github BI Tool

Authentication successful.

Please enter Username: *
keshavkundu

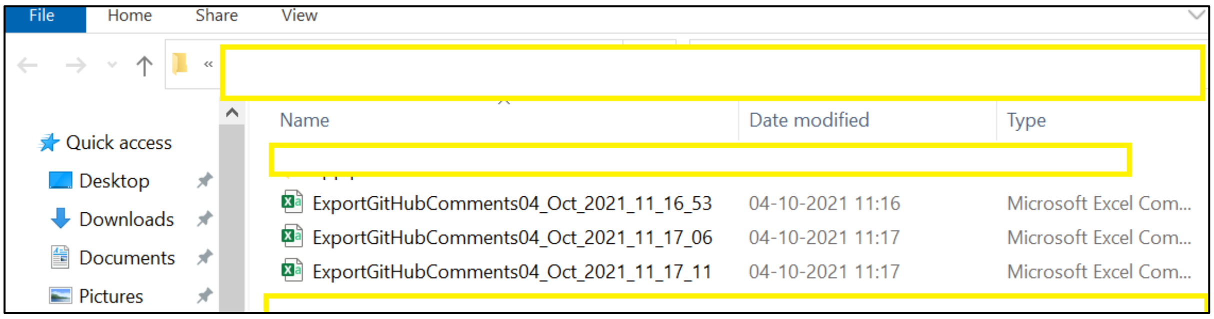
Please enter Private Access Toker *
[redacted]

Please enter Git enterprise repository Url *
https://github.com/keshavkundu/KK.GitHub.Demo.git

The most common words used by the user in comments section during the check in process sorted in ASCII values ascending.

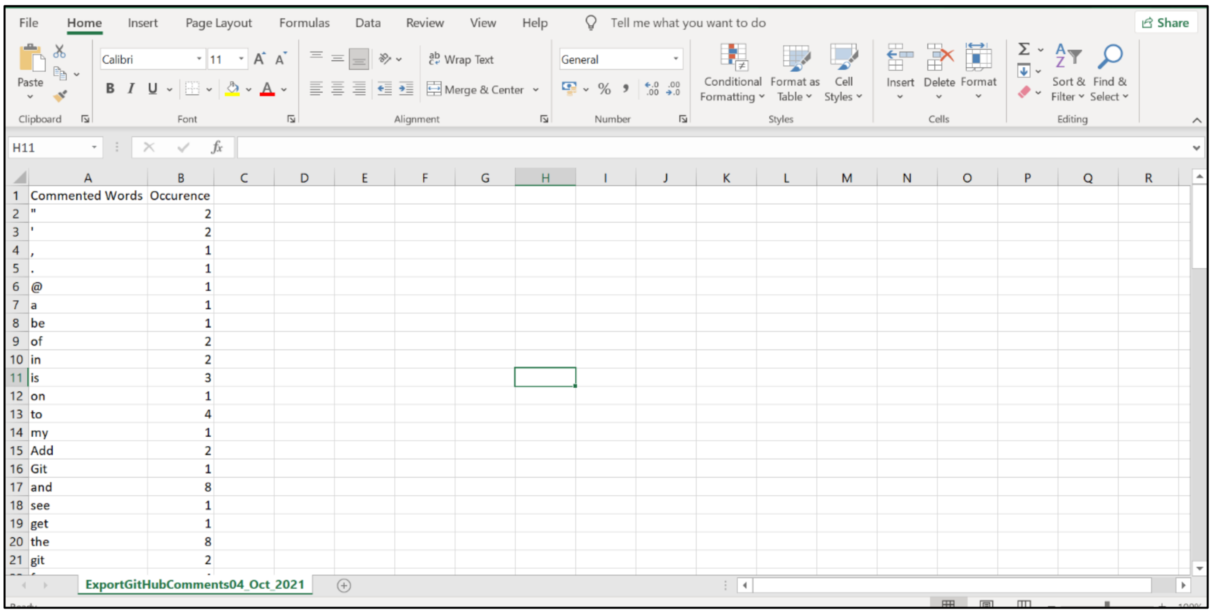
	Commented Words	Occurence
	process	1
	sorting	1
	version	2
	UnitTest	1
	Prebuilt	1
	username	1
	comments	1

Step3: Clicking on export to CSV will open up the directory.



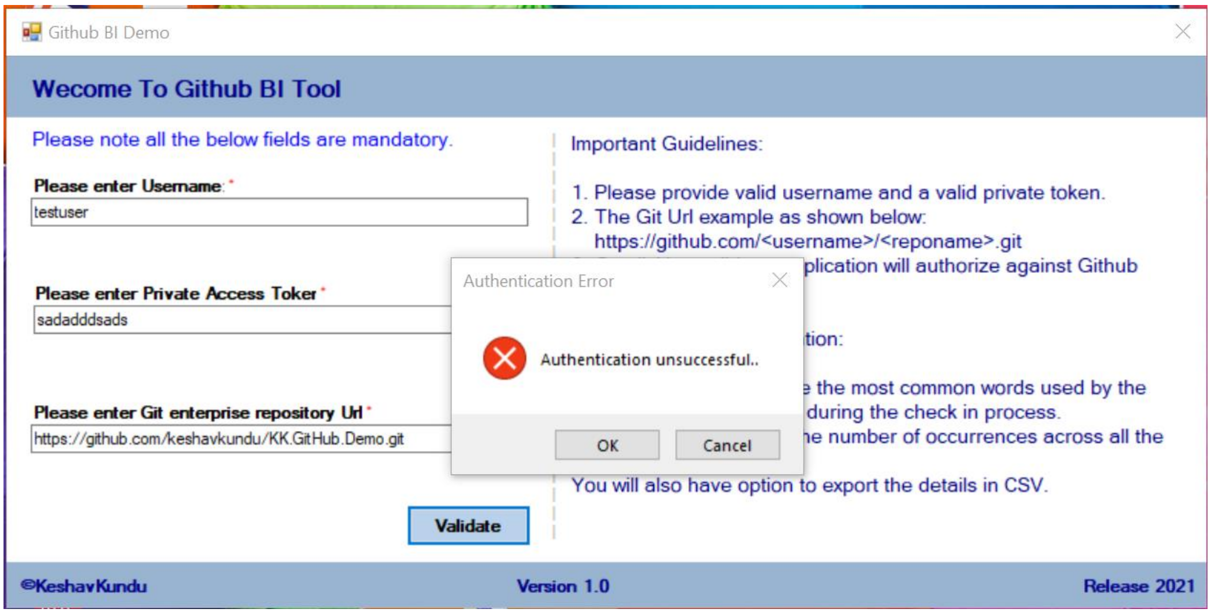
Name	Date modified	Type
ExportGitHubComments04_Oct_2021_11_16_53	04-10-2021 11:16	Microsoft Excel Com...
ExportGitHubComments04_Oct_2021_11_17_06	04-10-2021 11:17	Microsoft Excel Com...
ExportGitHubComments04_Oct_2021_11_17_11	04-10-2021 11:17	Microsoft Excel Com...

Step 4: Open the CSV to validate the data



	A	B
1	Commented Words	Occurence
2	"	2
3	'	2
4	,	1
5	.	1
6	@	1
7	a	1
8	be	1
9	of	2
10	in	2
11	is	3
12	on	1
13	to	4
14	my	1
15	Add	2
16	Git	1
17	and	8
18	see	1
19	get	1
20	the	8
21	git	2

Authentication unsuccessful for invalid details entered



Github BI Demo

Wecome To Github BI Tool

Please note all the below fields are mandatory.

Please enter Username *
testuser

Please enter Private Access Toker *
sadadddsads

Please enter Git enterprise repository Url *
https://github.com/keshavkundu/KK.GitHub.Demo.git

Validate

Important Guidelines:

1. Please provide valid username and a valid private token.
2. The Git Url example as shown below:
   https://github.com/<username>/<reponame>.git

plication will authorize against Github

tion:

e the most common words used by the
during the check in process.
he number of occurrences across all the

You will also have option to export the details in CSV.

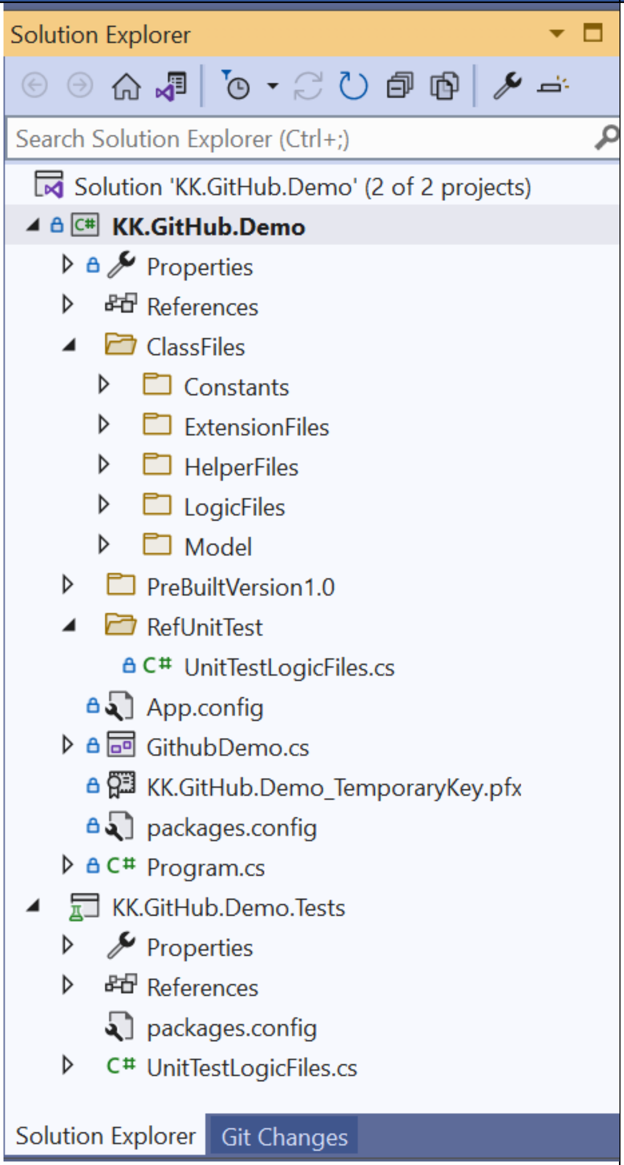
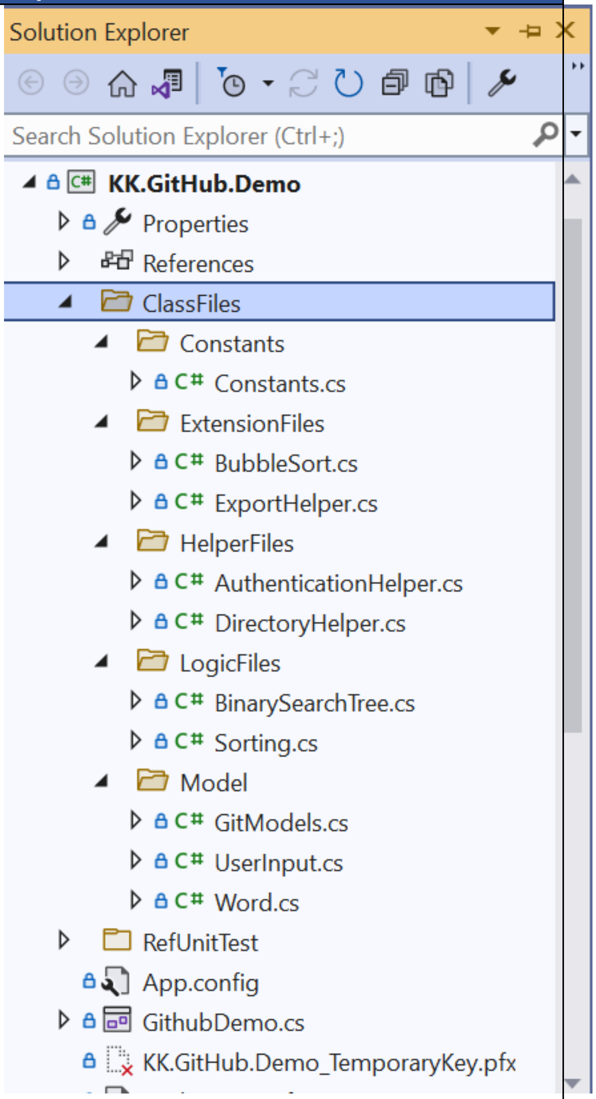
Authentication Error

Authentication unsuccessful..

OK Cancel

©KeshavKundu Version 1.0 Release 2021

Solution structure

Snapshot of the solution Explorer	Snapshot with class files folder expanded
	

```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'KK.GitHub.Demo' (2 of 2 projects)
  KK.GitHub.Demo
    Properties
    References
    ClassFiles
      Constants
      ExtensionFiles
      HelperFiles
      LogicFiles
      Model
    PreBuiltVersion1.0
    RefUnitTest
      UnitTestLogicFiles.cs
    App.config
    GithubDemo.cs
    KK.GitHub.Demo_TemporaryKey.pfx
    packages.config
    Program.cs
  KK.GitHub.Demo.Tests
    Properties
    References
    packages.config
    UnitTestLogicFiles.cs
Solution Explorer   Git Changes
```

```
Solution Explorer
Search Solution Explorer (Ctrl+;)
KK.GitHub.Demo
  Properties
  References
  ClassFiles
    Constants
      Constants.cs
    ExtensionFiles
      BubbleSort.cs
      ExportHelper.cs
    HelperFiles
      AuthenticationHelper.cs
      DirectoryHelper.cs
    LogicFiles
      BinarySearchTree.cs
      Sorting.cs
    Model
      GitModels.cs
      UserInput.cs
      Word.cs
  RefUnitTest
  App.config
  GithubDemo.cs
  KK.GitHub.Demo_TemporaryKey.pfx
```

ClassFiles explained -> Constants

Constants.cs: A constant class to store all the hardcoded values in application. Some of the constants can be included in App.config file.

ClassFiles explained -> ExtensionFiles

ExportHelper.cs: This class is used for exporting a data table into csv format and logging exception. This extension method is reusable for exporting a csv from any datatable format and logging in text file.

BubbleSort.cs: This class is used for exporting a list of word into debatable after bubble sorting to bind in data grid view.

ClassFiles explained -> HelperFiles

AuthenticationHelper.cs: This class is used for authenticating GitHub based on user inputs and uses no thread safe singleton design pattern.

DirectoryHelper.cs: This class is used to open the folder location where the CSV is downloaded or exception is captured (in case exception happens) once user clicks export and uses no thread safe singleton design pattern.

ClassFiles explained -> LogicFiles

BinarySearchTree.cs: BinarySearchTree class methods are being used to add nodes in the search tree. A static method is present to convert the tree/root to data table. This also has a class called node which has LeftNode of type Node, RightNode of type node, Word as string and Count as integer.

Sorting.cs: This class is being used to compare which string is in ascending order based on ASCII characters present in both the strings.

ClassFiles explained -> Model

GitModel.cs: This has two classes GitModelCommit and CommitComments and is used to deserialize JSON response from GIT using HTTPClient.

UserInput.cs: This class is used as a model for the user inputs.

Word.cs: This class is for storing the commented word and the occurrence.

Refence Unit Test file

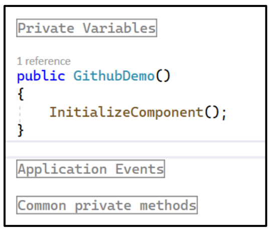
UnitTestLogicFiles.cs: The reference unit test file is kept in the solution in commented mode to understand the unit test cases written and run for the logic files -> BinarySearchTree and Sorting along with BubbleSort extension method.

PreBuiltVersion1.0

A pre published release files are kept inside the folder so that anyone can run the application without building it. **Exclude this folder if you want to run the solution or build it.**

Windows Form

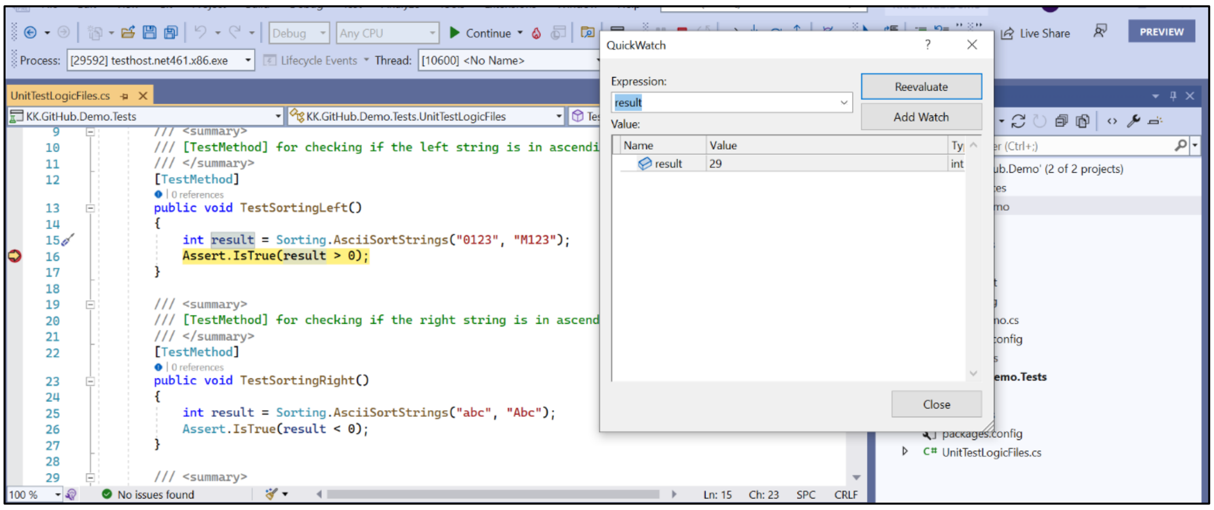
The form name is GithubDemo and basic validations like mandatory is provided for the form. The labels and controls show/hide and disabling feature is present in run time. The form also has a data grid view to bind the response and show to user. Pagination has not been implemented and load testing is not done for the grid. Region is created in the cs file for user readability.



```
Private Variables

1 reference
public GithubDemo()
{
    InitializeComponent();
}

Application Events

Common private methods
```

Unit Testing of the solution

For checking left string is in ascending ASCII order



Process: [29592] testhost.net461.x86.exe   Thread: [10600] <No Name>

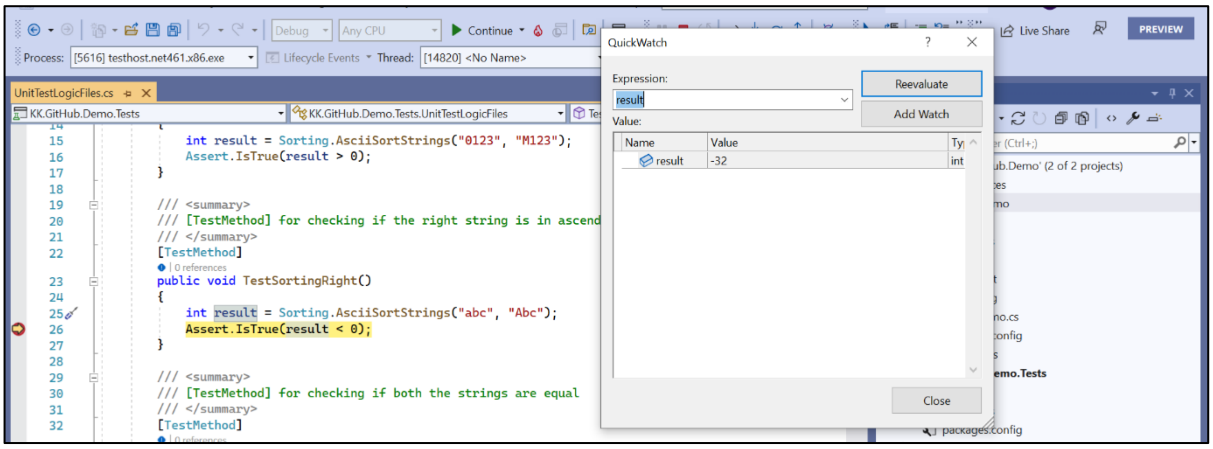
```
        /// <summary>
        /// [TestMethod] for checking if the left string is in ascendi
        /// </summary>
        [TestMethod]
        0 references
        public void TestSortingLeft()
        {
            int result = Sorting.AsciiSortStrings("0123", "M123");
            Assert.IsTrue(result > 0);
        }

        /// <summary>
        /// [TestMethod] for checking if the right string is in ascend
        /// </summary>
        [TestMethod]
        0 references
        public void TestSortingRight()
        {
            int result = Sorting.AsciiSortStrings("abc", "Abc");
            Assert.IsTrue(result < 0);
        }

        /// <summary>
```

QuickWatch

Expression: result

Name	Value	Ty
result	29	int

For checking right string is in ascending ASCII order



Process: [5616] testhost.net461.x86.exe   Thread: [14820] <No Name>

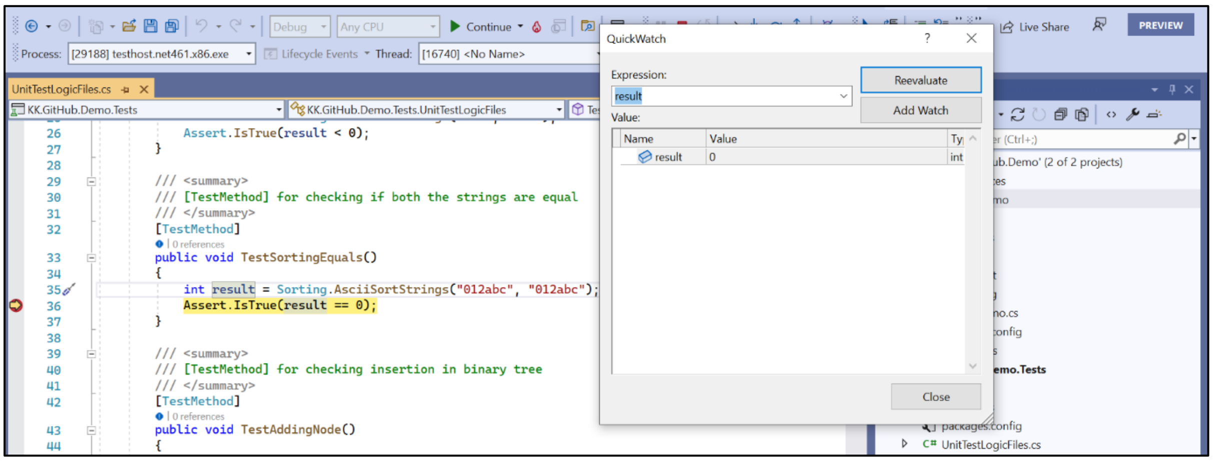
```
            int result = Sorting.AsciiSortStrings("0123", "M123");
            Assert.IsTrue(result > 0);
        }

        /// <summary>
        /// [TestMethod] for checking if the right string is in ascend
        /// </summary>
        [TestMethod]
        0 references
        public void TestSortingRight()
        {
            int result = Sorting.AsciiSortStrings("abc", "Abc");
            Assert.IsTrue(result < 0);
        }

        /// <summary>
        /// [TestMethod] for checking if both the strings are equal
        /// </summary>
        [TestMethod]
```

QuickWatch

Expression: result

Name	Value	Ty
result	-32	int

For checking both the strings are equal.



Process: [29188] testhost.net461.x86.exe   Thread: [16740] <No Name>

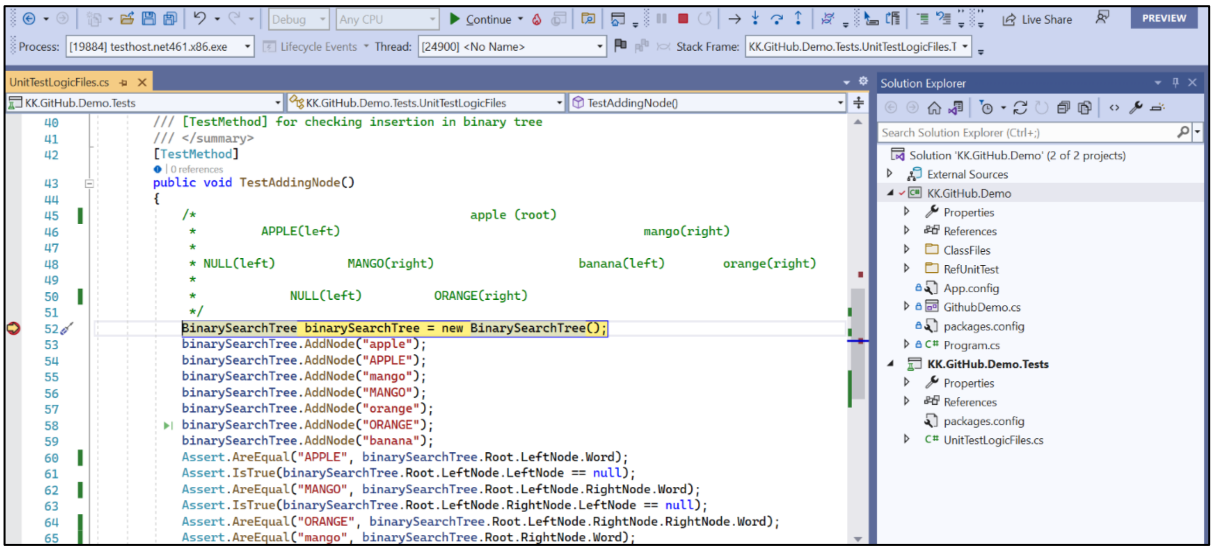
```
            Assert.IsTrue(result < 0);
        }

        /// <summary>
        /// [TestMethod] for checking if both the strings are equal
        /// </summary>
        [TestMethod]
        0 references
        public void TestSortingEquals()
        {
            int result = Sorting.AsciiSortStrings("012abc", "012abc");
            Assert.IsTrue(result == 0);
        }

        /// <summary>
        /// [TestMethod] for checking insertion in binary tree
        /// </summary>
        [TestMethod]
        0 references
        public void TestAddingNode()
        {
```

QuickWatch

Expression: result

Name	Value	Ty
result	0	int

For checking binary string insertion



Debug · Any CPU · Continue · Live Share · PREVIEW

Process: [19884] testhost.net461.x86.exe · Lifecycle Events · Thread: [24900] <No Name> · Stack Frame: KK.GitHub.Demo.Tests.UnitTestLogicFiles.T

UnitTestLogicFiles.cs · KK.GitHub.Demo.Tests · KK.GitHub.Demo.Tests.UnitTestLogicFiles · TestAddingNode()

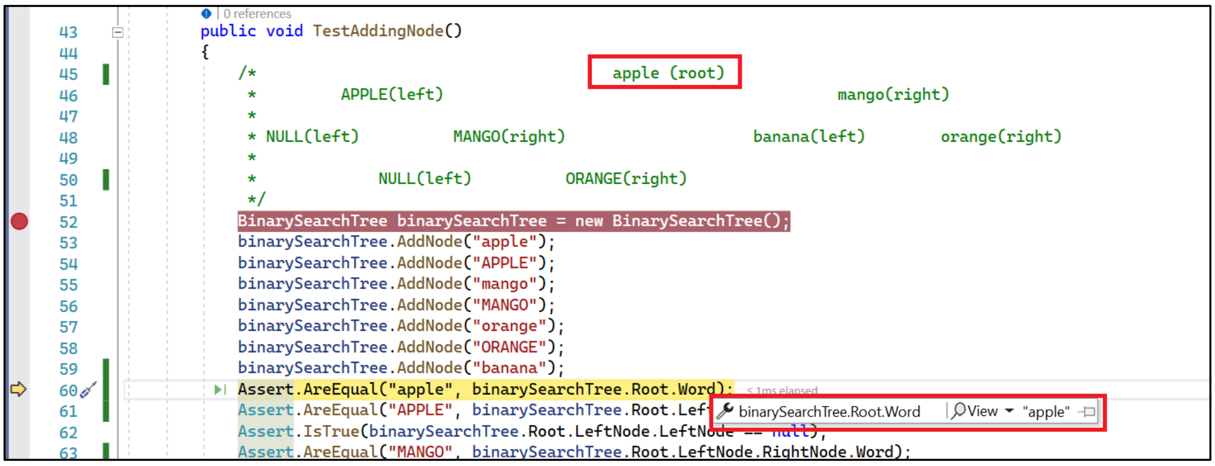
```
        /// [TestMethod] for checking insertion in binary tree
        /// </summary>
        [TestMethod]
        0 references
        public void TestAddingNode()
        {
            /*                                       apple (root)
             *          APPLE(left)                                           mango(right)
             *
             * NULL(left)          MANGO(right)                   banana(left)         orange(right)
             *
             *             NULL(left)          ORANGE(right)
             */
            BinarySearchTree binarySearchTree = new BinarySearchTree();
            binarySearchTree.AddNode("apple");
            binarySearchTree.AddNode("APPLE");
            binarySearchTree.AddNode("mango");
            binarySearchTree.AddNode("MANGO");
            binarySearchTree.AddNode("orange");
            binarySearchTree.AddNode("ORANGE");
            binarySearchTree.AddNode("banana");
            Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
            Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);
            Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
            Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == null);
            Assert.AreEqual("ORANGE", binarySearchTree.Root.LeftNode.RightNode.RightNode.Word);
            Assert.AreEqual("mango", binarySearchTree.Root.RightNode.Word);
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'KK.GitHub.Demo' (2 of 2 projects)
  External Sources
  KK.GitHub.Demo
    Properties
    References
    ClassFiles
    RefUnitTest
    App.config
    GithubDemo.cs
    packages.config
    Program.cs
  KK.GitHub.Demo.Tests
    Properties
    References
    packages.config
    UnitTestLogicFiles.cs
```

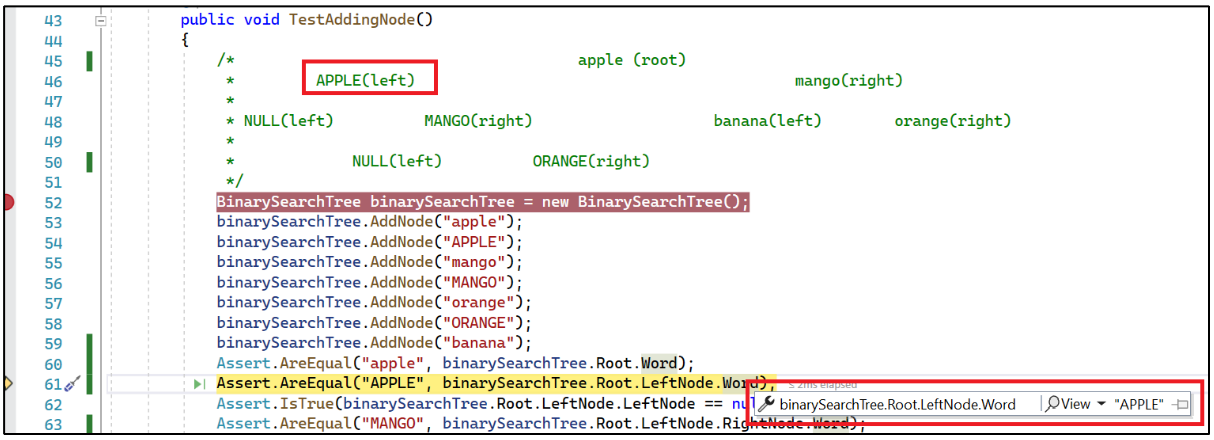
Test Execution 1: To check the root node



```
        0 references
        public void TestAddingNode()
        {
            /*                                      apple (root)
             *          APPLE(left)                                      mango(right)
             *
             * NULL(left)          MANGO(right)                banana(left)        orange(right)
             *
             *             NULL(left)          ORANGE(right)
             */
            BinarySearchTree binarySearchTree = new BinarySearchTree();
            binarySearchTree.AddNode("apple");
            binarySearchTree.AddNode("APPLE");
            binarySearchTree.AddNode("mango");
            binarySearchTree.AddNode("MANGO");
            binarySearchTree.AddNode("orange");
            binarySearchTree.AddNode("ORANGE");
            binarySearchTree.AddNode("banana");
            Assert.AreEqual("apple", binarySearchTree.Root.Word);  ≤ 1ms elapsed
            Assert.AreEqual("APPLE", binarySearchTree.Root.Lef
            Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);
            Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
```

binarySearchTree.Root.Word View "apple"

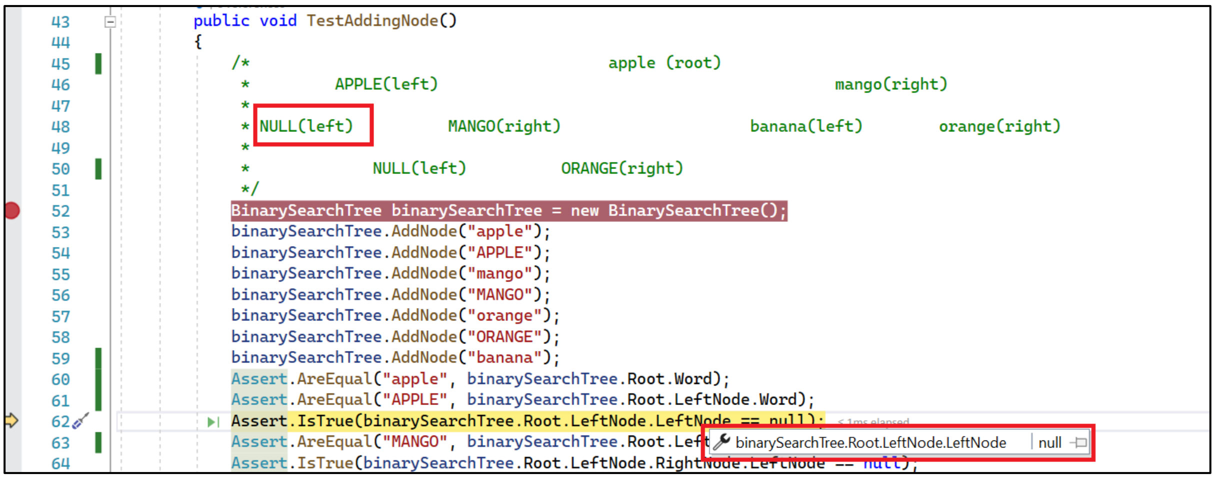
Test Execution 2: To check the first node on the left



```
        public void TestAddingNode()
        {
            /*                                      apple (root)
             *          APPLE(left)                                      mango(right)
             *
             * NULL(left)          MANGO(right)                banana(left)        orange(right)
             *
             *             NULL(left)          ORANGE(right)
             */
            BinarySearchTree binarySearchTree = new BinarySearchTree();
            binarySearchTree.AddNode("apple");
            binarySearchTree.AddNode("APPLE");
            binarySearchTree.AddNode("mango");
            binarySearchTree.AddNode("MANGO");
            binarySearchTree.AddNode("orange");
            binarySearchTree.AddNode("ORANGE");
            binarySearchTree.AddNode("banana");
            Assert.AreEqual("apple", binarySearchTree.Root.Word);
            Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);  ≤ 2ms elapsed
            Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == nu
            Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
```

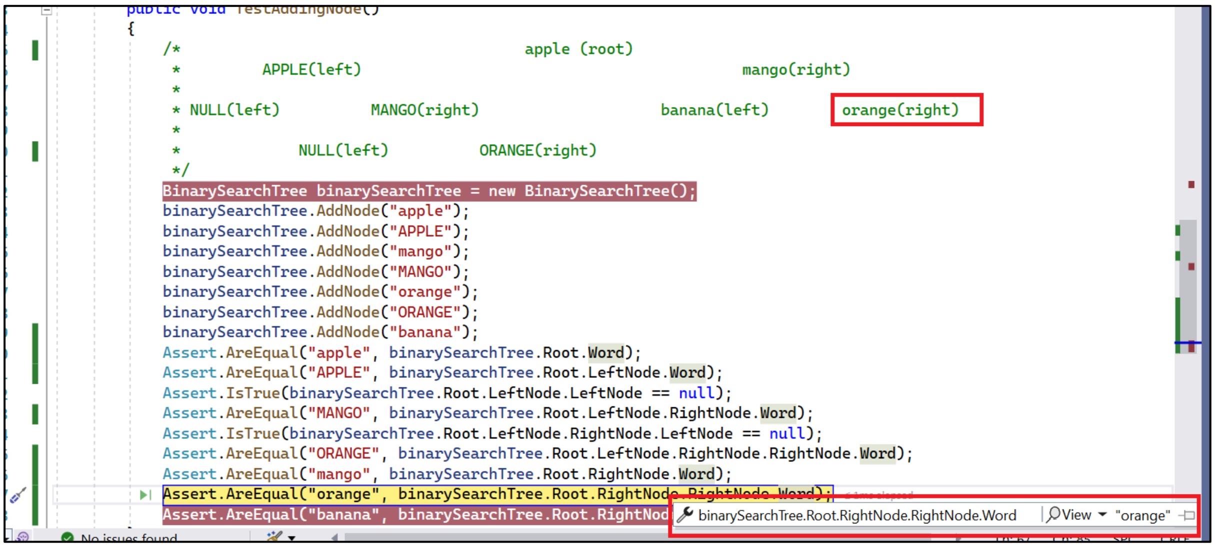
binarySearchTree.Root.LeftNode.Word View "APPLE"

Test Execution 3: To compare the left node for APPLE (left node for root)

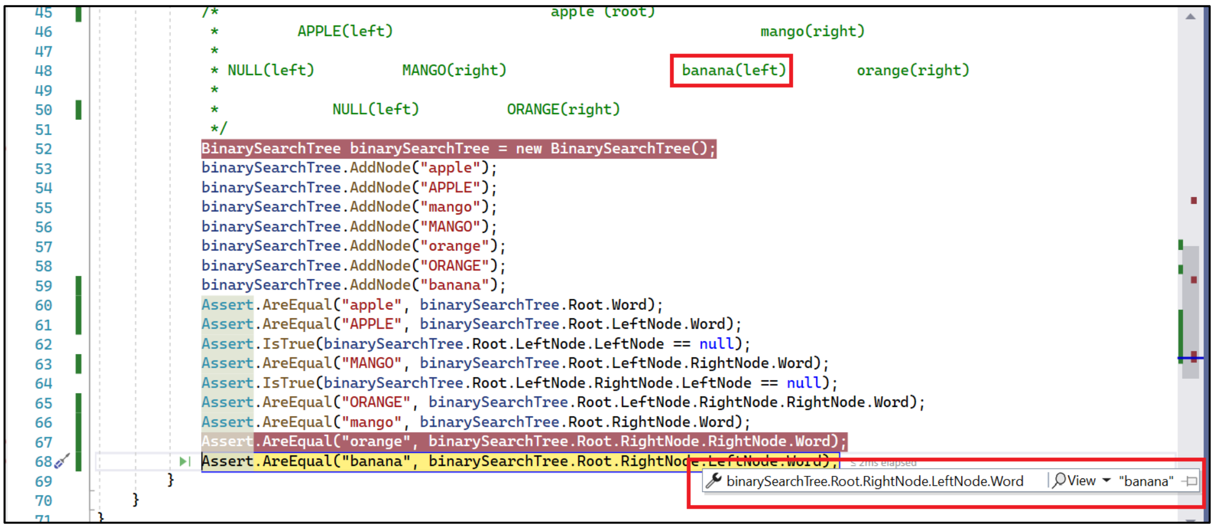


```
public void TestAddingNode()
{
    /*                                apple (root)
     *         APPLE(left)                                     mango(right)
     *
     * NULL(left)          MANGO(right)                banana(left)        orange(right)
     *
     *              NULL(left)          ORANGE(right)
     */
    BinarySearchTree binarySearchTree = new BinarySearchTree();
    binarySearchTree.AddNode("apple");
    binarySearchTree.AddNode("APPLE");
    binarySearchTree.AddNode("mango");
    binarySearchTree.AddNode("MANGO");
    binarySearchTree.AddNode("orange");
    binarySearchTree.AddNode("ORANGE");
    binarySearchTree.AddNode("banana");
    Assert.AreEqual("apple", binarySearchTree.Root.Word);
    Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
    Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);  ≤ 1ms elapsed
    Assert.AreEqual("MANGO", binarySearchTree.Root.Left   binarySearchTree.Root.LeftNode.LeftNode | null
    Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == null);
```

Test Execution 4: Similar way test execution is performed for all the nodes to check binary tree insertion.



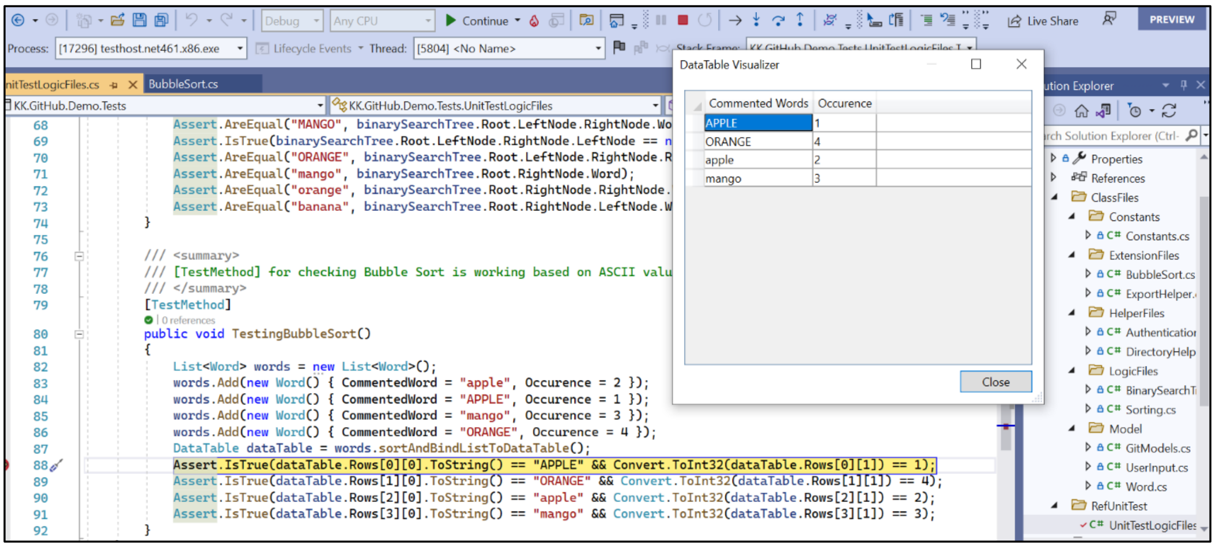
```
public void TestAddingNode()
{
    /*                                apple (root)
     *         APPLE(left)                                     mango(right)
     *
     * NULL(left)          MANGO(right)                banana(left)        orange(right)
     *
     *              NULL(left)          ORANGE(right)
     */
    BinarySearchTree binarySearchTree = new BinarySearchTree();
    binarySearchTree.AddNode("apple");
    binarySearchTree.AddNode("APPLE");
    binarySearchTree.AddNode("mango");
    binarySearchTree.AddNode("MANGO");
    binarySearchTree.AddNode("orange");
    binarySearchTree.AddNode("ORANGE");
    binarySearchTree.AddNode("banana");
    Assert.AreEqual("apple", binarySearchTree.Root.Word);
    Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
    Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);
    Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
    Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == null);
    Assert.AreEqual("ORANGE", binarySearchTree.Root.LeftNode.RightNode.RightNode.Word);
    Assert.AreEqual("mango", binarySearchTree.Root.RightNode.Word);
    Assert.AreEqual("orange", binarySearchTree.Root.RightNode.RightNode.Word);  ≤ 1ms elapsed
    Assert.AreEqual("banana", binarySearchTree.Root.RightNod   binarySearchTree.Root.RightNode.RightNode.Word | View ▾ "orange"
No issues found
```



```
    /*                                apple (root)
     *         APPLE(left)                                     mango(right)
     *
     * NULL(left)          MANGO(right)                banana(left)        orange(right)
     *
     *              NULL(left)          ORANGE(right)
     */
    BinarySearchTree binarySearchTree = new BinarySearchTree();
    binarySearchTree.AddNode("apple");
    binarySearchTree.AddNode("APPLE");
    binarySearchTree.AddNode("mango");
    binarySearchTree.AddNode("MANGO");
    binarySearchTree.AddNode("orange");
    binarySearchTree.AddNode("ORANGE");
    binarySearchTree.AddNode("banana");
    Assert.AreEqual("apple", binarySearchTree.Root.Word);
    Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
    Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);
    Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
    Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == null);
    Assert.AreEqual("ORANGE", binarySearchTree.Root.LeftNode.RightNode.RightNode.Word);
    Assert.AreEqual("mango", binarySearchTree.Root.RightNode.Word);
    Assert.AreEqual("orange", binarySearchTree.Root.RightNode.RightNode.Word);
    Assert.AreEqual("banana", binarySearchTree.Root.RightNode.LeftNode.Word);  ≤ 2ms elapsed
                                                        binarySearchTree.Root.RightNode.LeftNode.Word | View ▾ "banana"
        }
    }
}
```

For checking Bubble sorting for the commented words

Bubble sort mechanism check for the list of words based on their ASCII values.

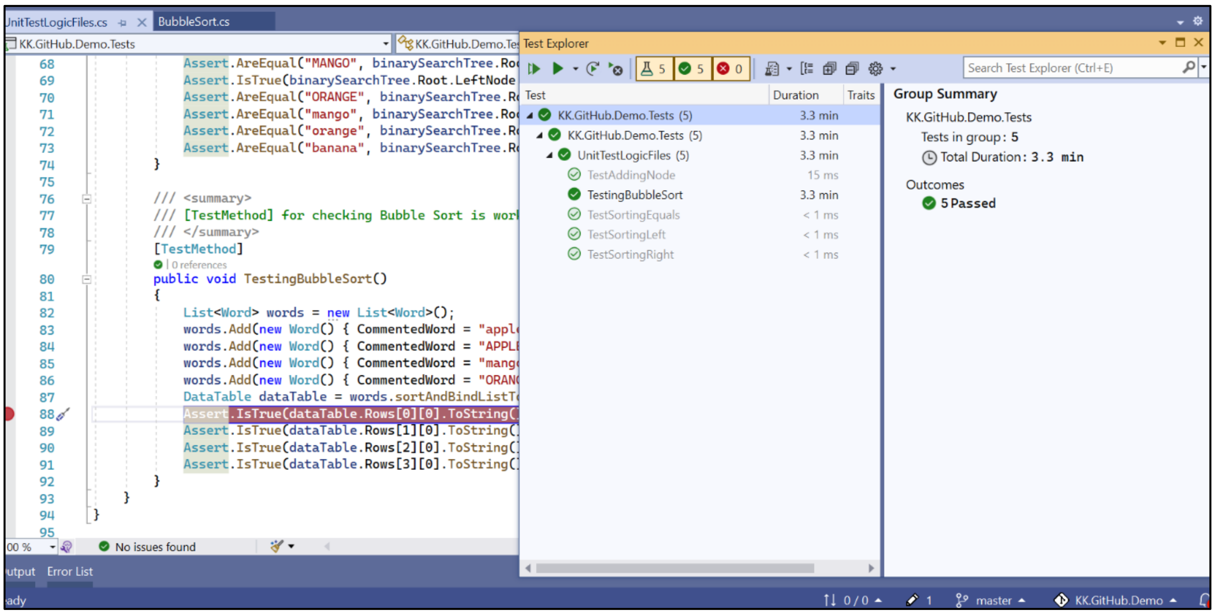


```
            Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Wo
            Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == n
            Assert.AreEqual("ORANGE", binarySearchTree.Root.LeftNode.RightNode.R
            Assert.AreEqual("mango", binarySearchTree.Root.RightNode.Word);
            Assert.AreEqual("orange", binarySearchTree.Root.RightNode.RightNode.
            Assert.AreEqual("banana", binarySearchTree.Root.RightNode.LeftNode.W
        }

        /// <summary>
        /// [TestMethod] for checking Bubble Sort is working based on ASCII valu
        /// </summary>
        [TestMethod]
        public void TestingBubbleSort()
        {
            List<Word> words = new List<Word>();
            words.Add(new Word() { CommentedWord = "apple", Occurence = 2 });
            words.Add(new Word() { CommentedWord = "APPLE", Occurence = 1 });
            words.Add(new Word() { CommentedWord = "mango", Occurence = 3 });
            words.Add(new Word() { CommentedWord = "ORANGE", Occurence = 4 });
            DataTable dataTable = words.sortAndBindListToDataTable();
            Assert.IsTrue(dataTable.Rows[0][0].ToString() == "APPLE" && Convert.ToInt32(dataTable.Rows[0][1]) == 1);
            Assert.IsTrue(dataTable.Rows[1][0].ToString() == "ORANGE" && Convert.ToInt32(dataTable.Rows[1][1]) == 4);
            Assert.IsTrue(dataTable.Rows[2][0].ToString() == "apple" && Convert.ToInt32(dataTable.Rows[2][1]) == 2);
            Assert.IsTrue(dataTable.Rows[3][0].ToString() == "mango" && Convert.ToInt32(dataTable.Rows[3][1]) == 3);
        }
```

DataTable Visualizer

Commented Words	Occurence
APPLE	1
ORANGE	4
apple	2
mango	3

All the test cases are passed to ensure that the logical layers are working fine.



```
            Assert.AreEqual("MANGO", binarySearchTree.Ro
            Assert.IsTrue(binarySearchTree.Root.LeftNode
            Assert.AreEqual("ORANGE", binarySearchTree.R
            Assert.AreEqual("mango", binarySearchTree.Ro
            Assert.AreEqual("orange", binarySearchTree.R
            Assert.AreEqual("banana", binarySearchTree.R
        }

        /// <summary>
        /// [TestMethod] for checking Bubble Sort is worl
        /// </summary>
        [TestMethod]
        public void TestingBubbleSort()
        {
            List<Word> words = new List<Word>();
            words.Add(new Word() { CommentedWord = "appl
            words.Add(new Word() { CommentedWord = "APPL
            words.Add(new Word() { CommentedWord = "mang
            words.Add(new Word() { CommentedWord = "ORAN
            DataTable dataTable = words.sortAndBindListT
            Assert.IsTrue(dataTable.Rows[0][0].ToString(
            Assert.IsTrue(dataTable.Rows[1][0].ToString(
            Assert.IsTrue(dataTable.Rows[2][0].ToString(
            Assert.IsTrue(dataTable.Rows[3][0].ToString(
        }
    }
}
```

Test Explorer

Test	Duration	Traits
KK.GitHub.Demo.Tests (5)	3.3 min	
KK.GitHub.Demo.Tests (5)	3.3 min	
UnitTestLogicFiles (5)	3.3 min	
TestAddingNode	15 ms	
TestingBubbleSort	3.3 min	
TestSortingEquals	< 1 ms	
TestSortingLeft	< 1 ms	
TestSortingRight	< 1 ms	

Group Summary
KK.GitHub.Demo.Tests
Tests in group: 5
Total Duration: 3.3 min
Outcomes
5 Passed

--- END OF DOCUMENT ---