



# 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 .....	2
Accessing Git Hub for getting Token and URL .....	2
Step1: Creating personal access token from GitHub. ....	2
Step2: Accessing the Git Repository URL.....	2
Testing the application UI .....	3
Step1: Enter the details in the screen and click validate. ....	3
Step2: After successful authentication you will be presented with a grid on right.....	3
Step3: Clicking on export to CSV will open up the directory. ....	4
Step 4: Open the CSV to validate the data.....	4
Authentication unsuccessful for invalid details entered .....	4
Solution structure .....	5
ClassFiles explained -> Constants .....	5
ClassFiles explained -> ExtensionFiles.....	5
ClassFiles explained -> HelperFiles.....	6
ClassFiles explained -> LogicFiles .....	6
ClassFiles explained -> Model .....	6
Refence Unit Test file.....	6
PreBuiltVersion1.0 .....	6
Windows Form.....	6
Unit Testing of the solution .....	7
For checking left string is in ascending ASCII order .....	7
For checking right string is in ascending ASCII order .....	7
For checking both the strings are equal.....	7
For checking binary string insertion.....	8

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**

The screenshot shows a web application titled "Github BI Demo". The main heading is "Welcome To Github BI Tool". Below this, a note states: "Please note all the below fields are mandatory." There are three input fields: "Please enter Username:", "Please enter Private Access Token:", and "Please enter Git enterprise repository Url:". A "Validate" button is located below these fields. To the right, under "Important Guidelines:", there are three numbered points: 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. Below this, it says "After successful authorization:" and lists two benefits: "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." and "You will also have option to export the details in CSV." The footer contains "©KeshavKundu", "Version 1.0", and "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

The screenshot shows the "Developer settings" page in GitHub. The left sidebar has "GitHub Apps", "OAuth Apps", and "Personal access tokens". The main content area is titled "Personal access tokens" and has buttons for "Generate new token" and "Revoke all". Below this, it says "Tokens you have generated that can be used to access the GitHub API." There is one token listed: "KeshavPersonalToken" with a "repo" scope, "Last used within the last week", and a "Delete" button. It also shows the expiration date: "Expires on Tue, Nov 2 2021." At the bottom, a note explains: "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.

The screenshot shows a GitHub repository page for "keshavkundu". The top bar shows "master" branch, "1 branch", and "0 tags". There are buttons for "Go to file", "Add file", and "Code". The "Code" button is highlighted, and a dropdown menu is open showing "Clone". Below "Clone", there are three options: "HTTPS", "SSH", and "GitHub CLI". The "HTTPS" option is selected, and the URL "https://github.com/keshavkundu/KK.GitHub.t" is displayed. A "Copied!" notification with a green checkmark is shown next to the URL. Below the URL, it says "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: \***

**Please enter Private Access Toker \***

**Please enter Git enterprise repository Uri \***

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

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: \***

**Please enter Private Access Toker \***

**Please enter Git enterprise repository Uri \***

Validate

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

Commented Words	Occurence
This	1
UnitTest	1
Update	1
a	1
and	7
application	2
authentication	1

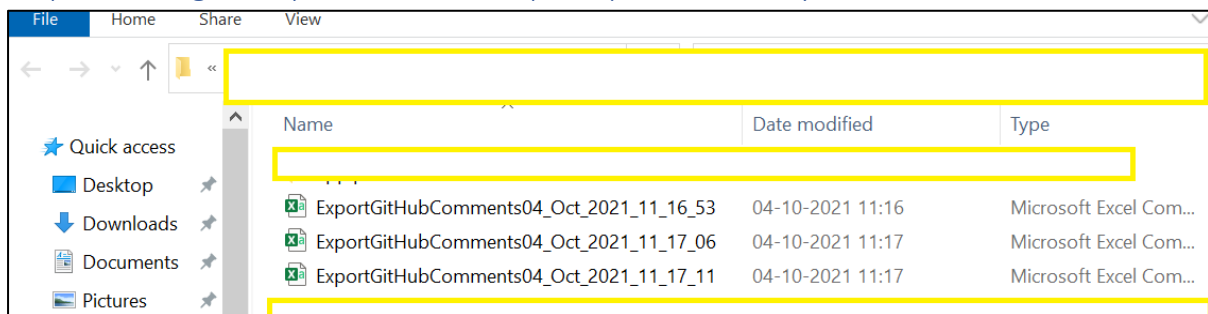
Export to CSV

©KeshavKundu

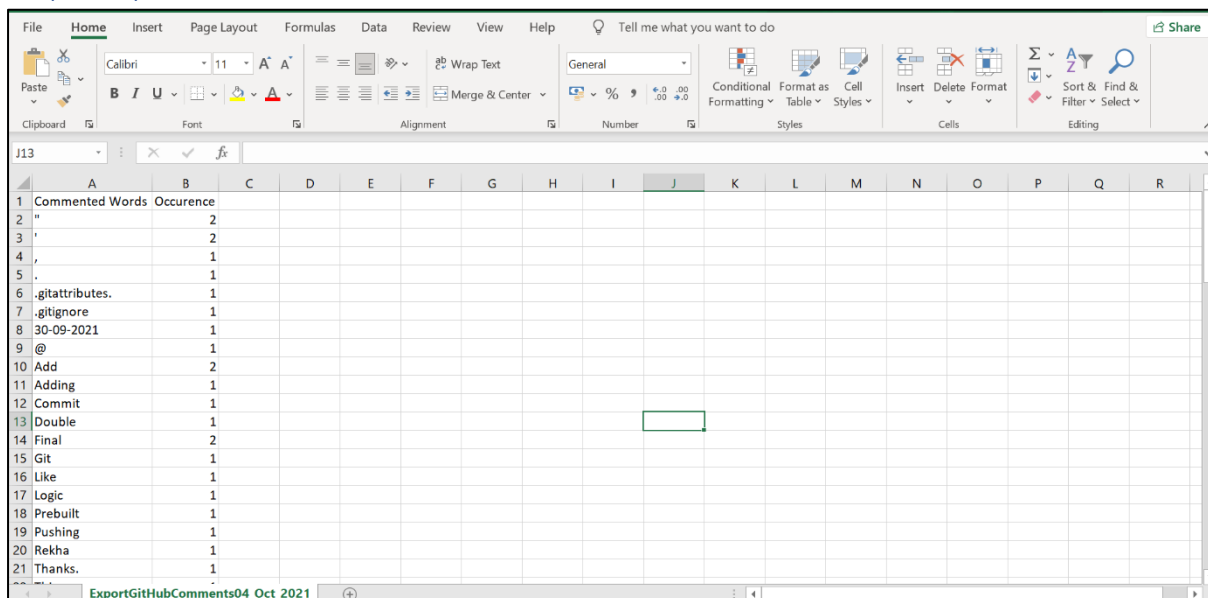
Version 1.0

Release 2021

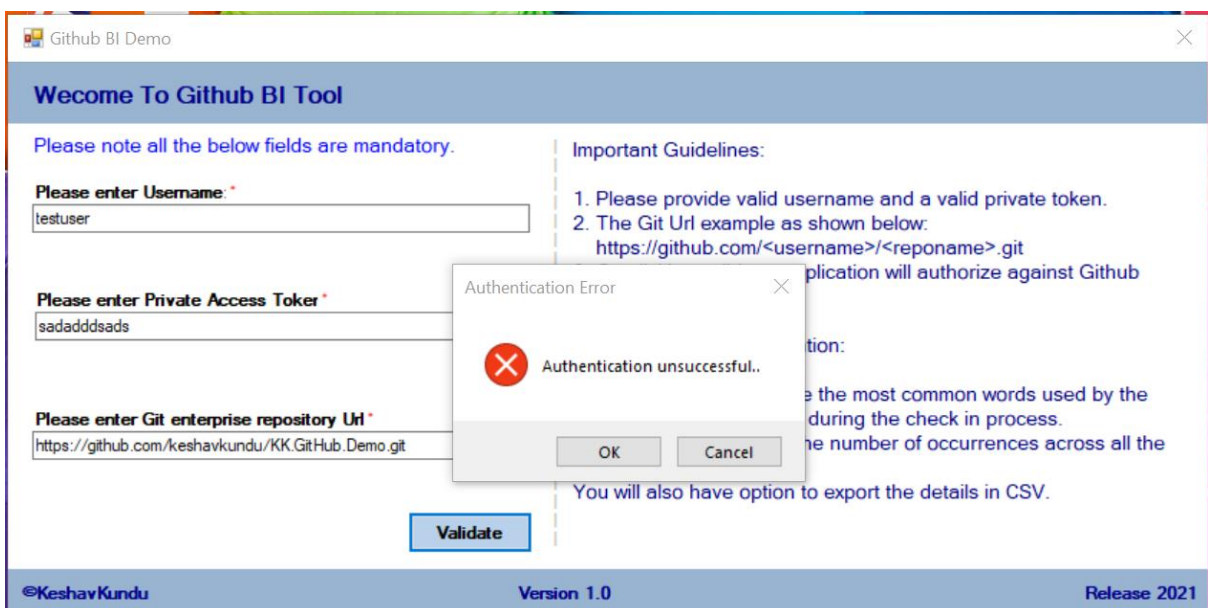
Step3: Clicking on export to CSV will open up the directory.



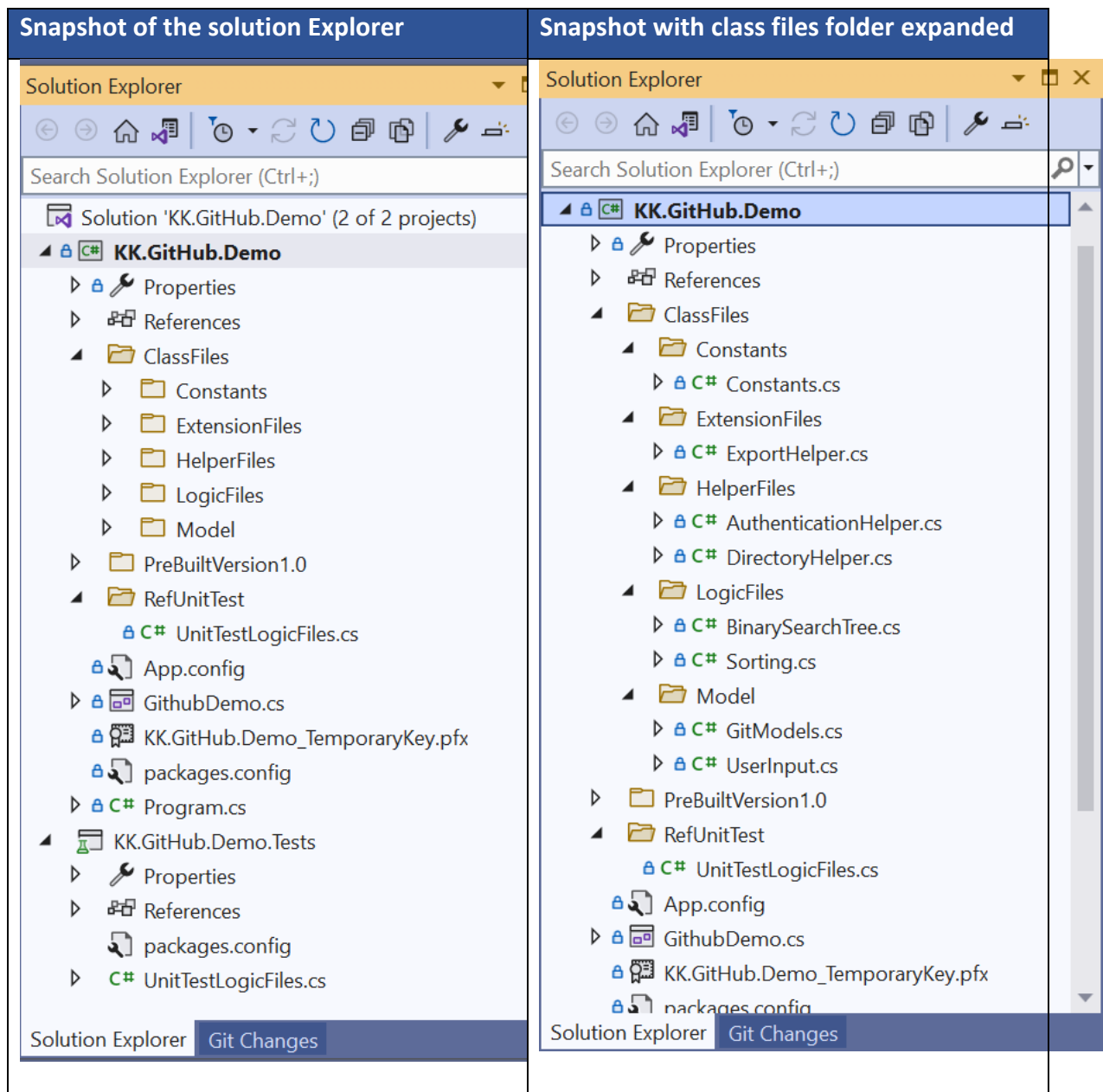
Step 4: Open the CSV to validate the data



Authentication unsuccessful for invalid details entered



## Solution structure



### 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.

## 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.

## Refence Unit Test file

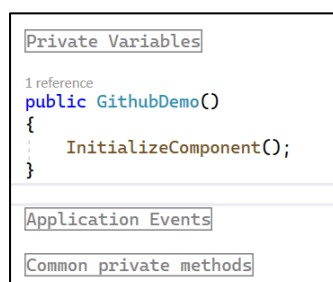
**UnitTestLogicFiles.cs:** The reference unit test file is kept in the solution in commented mode to understand the unit test cases written and run for the logic files -> BinarySearchTree and Sorting.

## 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.**

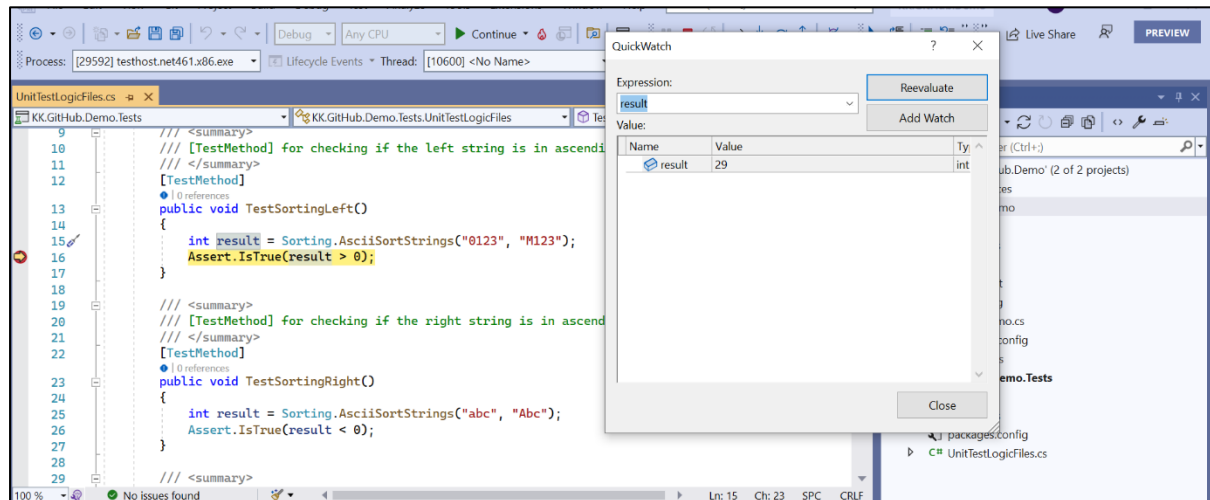
## Windows Form

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.

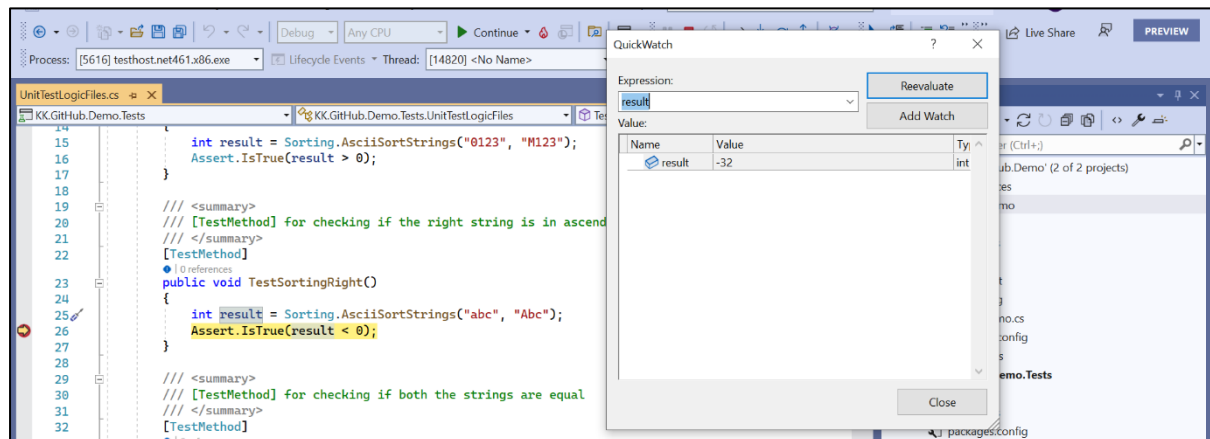


## Unit Testing of the solution

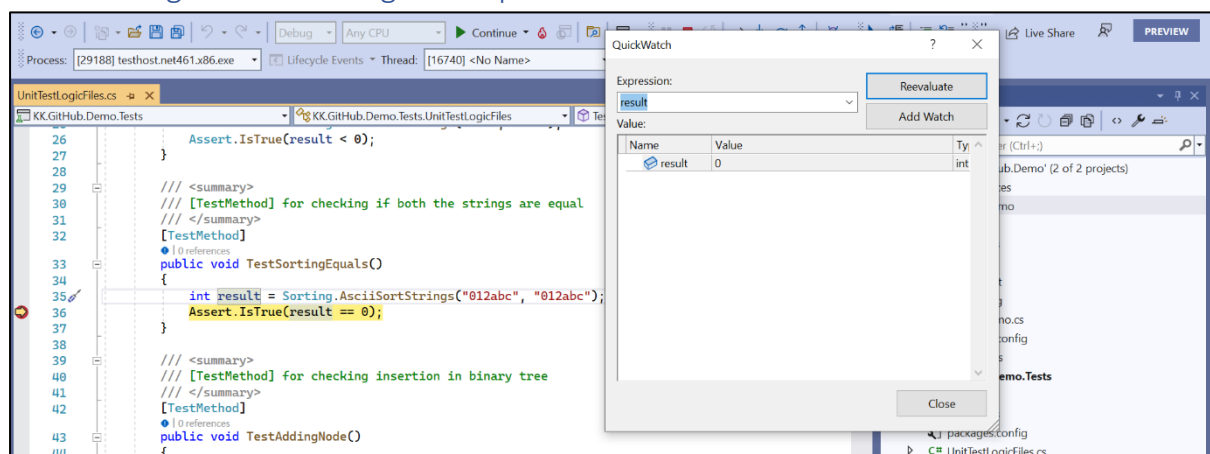
For checking left string is in ascending ASCII order



For checking right string is in ascending ASCII order

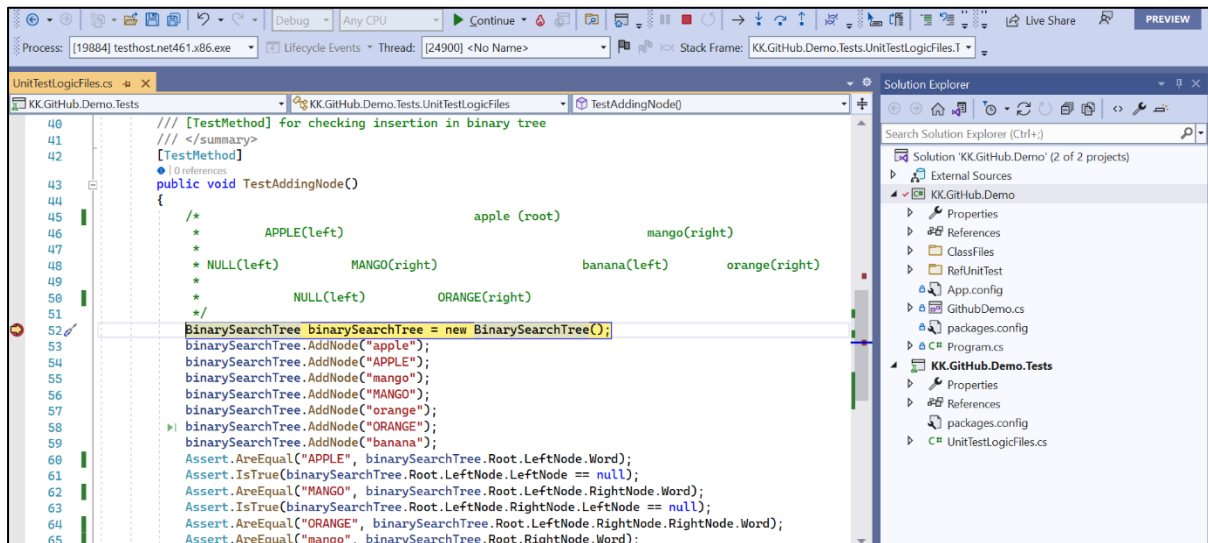


For checking both the strings are equal.

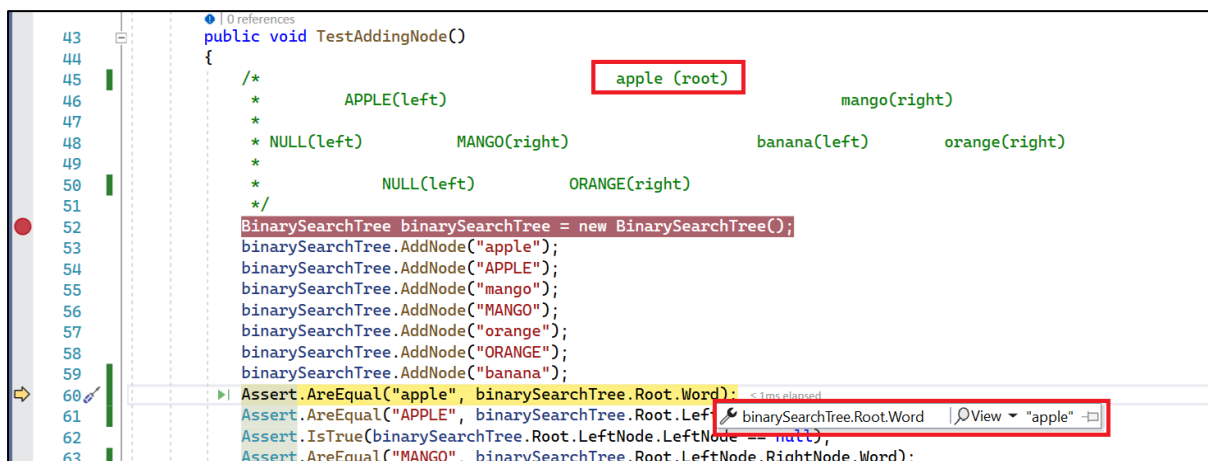




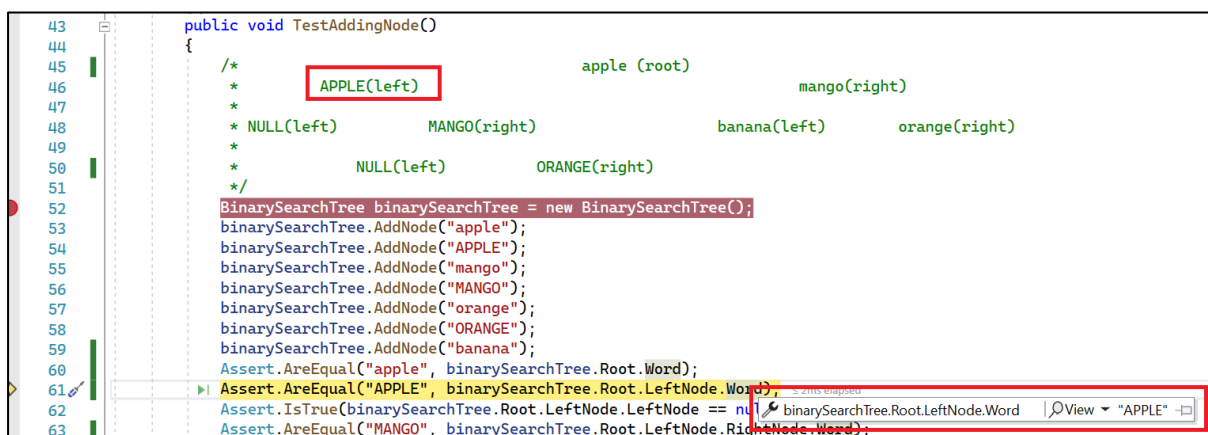
For checking binary string insertion



### Test Execution 1: To check the root node



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



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

```

43 public void TestAddingNode()
44 {
45     /*                                apple (root)
46     *      APPLE(left)                                mango(right)
47     *
48     *  NULL(left)      MANGO(right)      banana(left)      orange(right)
49     *
50     *      NULL(left)      ORANGE(right)
51     */
52     BinarySearchTree binarySearchTree = new BinarySearchTree();
53     binarySearchTree.AddNode("apple");
54     binarySearchTree.AddNode("APPLE");
55     binarySearchTree.AddNode("mango");
56     binarySearchTree.AddNode("MANGO");
57     binarySearchTree.AddNode("orange");
58     binarySearchTree.AddNode("ORANGE");
59     binarySearchTree.AddNode("banana");
60     Assert.AreEqual("apple", binarySearchTree.Root.Word);
61     Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
62     Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null); // true elapsed
63     Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.LeftNode.Word);
64     Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.RightNode == null);

```

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

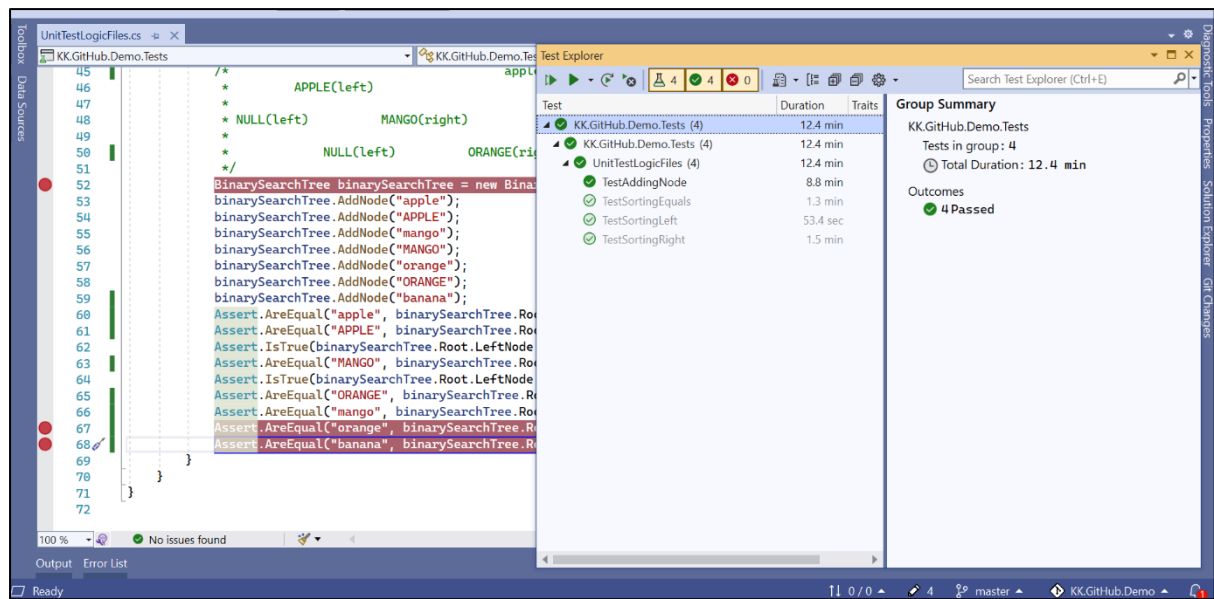
```
public void TestAddingNode()
{
    /*
        apple (root)
        *
        * APPLE(left) mango(right)
        *
        * NULL(left) MANGO(right) banana(left) orange(right)
        *
        * NULL(left) ORANGE(right)
        */
    BinarySearchTree binarySearchTree = new BinarySearchTree();
    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);
    Assert.AreEqual("banana", binarySearchTree.Root.RightNode.RightNode.RightNode.Word);
}
```

```

45  /*                                apple (root)                                mango(right)
46  *      APPLE(left)
47  *
48  * NULL(left)      MANGO(right)      banana(left)      orange(right)
49  *
50  *      NULL(left)      ORANGE(right)
51  */
52  BinarySearchTree binarySearchTree = new BinarySearchTree();
53  binarySearchTree.AddNode("apple");
54  binarySearchTree.AddNode("APPLE");
55  binarySearchTree.AddNode("mango");
56  binarySearchTree.AddNode("MANGO");
57  binarySearchTree.AddNode("orange");
58  binarySearchTree.AddNode("ORANGE");
59  binarySearchTree.AddNode("banana");
60  Assert.AreEqual("apple", binarySearchTree.Root.Word);
61  Assert.AreEqual("APPLE", binarySearchTree.Root.LeftNode.Word);
62  Assert.IsTrue(binarySearchTree.Root.LeftNode.LeftNode == null);
63  Assert.AreEqual("MANGO", binarySearchTree.Root.LeftNode.RightNode.Word);
64  Assert.IsTrue(binarySearchTree.Root.LeftNode.RightNode.LeftNode == null);
65  Assert.AreEqual("ORANGE", binarySearchTree.Root.LeftNode.RightNode.RightNode.Word);
66  Assert.AreEqual("mango", binarySearchTree.Root.RightNode.Word);
67  Assert.AreEqual("orange", binarySearchTree.Root.RightNode.RightNode.Word);
68  Assert.AreEqual("banana", binarySearchTree.Root.RightNode.LeftNode.Word);
69  }
70  }

```

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



----- END OF DOCUMENT -----