Test Cases

Black-Box Test Cases

These test cases are defined based on the available documentation and execution of the program. The code was not inspected.

#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation (“pass” or failure description)
1	**notAcceptEmptyUsernameAndPassword**	Tracker application initialized and visible.	• Set username field to an empty string. • Set password field to an empty string. • Click the "Create Account & Sign In" button.	The application should not proceed with empty username and password fields and should display an error message.	**Fail** **The program should not allow to enter without username and password**
2	**notAcceptEmptyUsername**	Tracker application initialized and visible.	• Set username field to an empty string. • Set password field to a valid string (e.g., "password"). • Click the "Create Account & Sign In" button.	The application should not proceed with an empty username field and should display an error message.	**Fail** **The program should not allow to enter without username**
3	**notAcceptEmptyPassword**	Tracker application initialized and visible.	• Set username field to a valid string (e.g., "username"). • Set password field to an empty string.	The application should not proceed with an empty password field and should display an error message.	**Fail** **The program should not allow to enter without password**

			• Click the "Create Account & Sign In" button.		
4	**validUsernameAndPassword**	Tracker application initialized and visible.	• Set username field to a valid string (e.g., "validUser"). • Set password field to a valid string (e.g., "validPassword"). • Click the "Create Account & Sign In" button.	The application should proceed and log in successfully.	**Pass**
5	**CreateAccountAndSignIn**	Tracker application initialized and visible.	• Set username field to a new valid string (e.g., "newUser"). • Set password field to a new valid string (e.g., "newPassword"). • Click the "Create Account & Sign In" button.	The application should create an account and log in successfully.	**Pass**
6	**notNegativeNumbers**	Tracker application initialized and visible.	• Set all input fields to negative numbers (e.g., -1). • Click the "Display Variables" button.	The application should not accept negative numbers and should display an error message.	**Fail** **The program should not run with negative numbers**
7	**zeroConsumptionTest**	Tracker application initialized and visible.	• Set all input fields to zero.	The application should calculate zero emission for all zero inputs.	**Pass**

			• Click the "Display Variables" button.		
8	**checkCalculationsResult**	Tracker application initialized and visible.	• Set all input fields to specific test values. • Click the "Display Variables" button.	The application should correctly calculate the expected result based on the given inputs.	**Fail** **The calculations are not correct because of human errors**
9	**checkAllFieldsAcceptOnlyNumbers**	Tracker application initialized and visible.	• Set all input fields to valid numeric values (e.g., "123", "456"). • Click the "Display Variables" button.	The application should accept all valid numeric values and proceed without errors.	**Pass**
10	**notAcceptStringInData**	Tracker application initialized and visible.	• Set all input fields to non-numeric strings (e.g., "Hola"). • Click the "Display Variables" button.	The application should not accept non-numeric values and should display an error message.	**Pass**
11	**checkPlantedTreesAcceptIntegers**	Tracker application initialized and visible.	• Set the "Number of Trees Planted" field to an integer value (e.g., "5"). • Click the "Display Variables" button.	The application should accept integer values for the "Number of Trees Planted" field.	**Pass**
12	**checkPlantedTreesAcceptNotAcceptDecimals**	Tracker application initialized and visible.	• Set the "Number of Trees Planted" field to a decimal value (e.g., "5.5"). • Click the "Display Variables" button.	The application should not accept decimal values for the "Number of Trees Planted" field and should display an error message.	**Fail** **The program should not allow to put tree numbers in decimals**

13	**regularConsumptionUS**	Tracker application initialized and visible.	• Set input fields to average US consumption values. • Click the "Display Variables" button.	The application should calculate a high sustainability index for high consumption values.	Pass
14	**ecologicalConsumption**	Tracker application initialized and visible.	• Set input fields to low, ecological consumption values. • Click the "Display Variables" button.	The application should calculate a low sustainability index for low consumption values.	Pass

White-Box Test Cases

These additional test cases were defined during inspection of the code.

#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation ("pass" or failure description)
1	**Check Electricity Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf"	Put the Consumptions per day of Electricity = 30 Kw x hr.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 10 642.5 kg -> CO2.	Fail The program is not following the calculations of the pdf file.
2	**Check House Fuel Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf"	Put the Consumptions per day of Natural Gas = 8 kg.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 18 400.0 -> CO2.	Fail The program is not following the calculations of the pdf file. (They forget change kg to gr)

3	**Check Water Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf"	Put the Consumptions per day of Water = 340 m3.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 134 300.0 -> CO2.	Pass
4	**Check Transport Private Fuel Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf"	Put the Consumptions per day of Petrol = 5 lt.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 10 785.0 -> CO2.	Fail The program is not following the calculations of the pdf file. (They forget change kg to gr)
5	**Check Waste Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf	Put the Consumptions per day of Waste = 2 kg.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 485.7 -> CO2.	Pass
6	**Check Public Transport Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf	Put the Consumptions per day of Travel dist. = 10 km.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected 37.7 -> CO2.	Pass
7	**Check Feet Travel Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf	Put the Reduction of Consumptions per day. Feet = 2 km.	• Create a variable with expected result. • Make the calculations in the program. • Compare Results	Expected -142.3 -> CO2.	Pass

8	**Check Planted Trees Calculation** Check individual calculation is related with the assumptions in "Initial_Parameter_Implementation.pdf	Put the Reduction of Consumptions per year. Trees= 80 trees	- Create a variable with expected result. - Make the calculations in the program. - Compare Results	Expected -2 428.6 -> CO2.	Fail The program is not following the calculations of the pdf file.
9	**Generate Recommendations** Recommendation messages are given based on a numerical value with the name index.	Check the recommendations index	Test the conditions based on the range of values that we input in the test	A Recommendation message based on the value of the input	**4 Passes** and **one failure** because the function accepts negative values and to our understanding I don’t think it’s supposed to accept negative values
10	**Tips about improving your activities about each case**	Put values that are greater than the recommended one’s for an ‘ecological citizen’	1 .Create each case and test if the program returns the corresponding tip. 2. Write manually the expected outcome. 3. Compare both Results	Tips about each (No tips, electricity, House Fuel, Private Transport, Public Transport, Waste, Water, Trees, Feet Distance).	1 Pass, 8 Fails. This is because their values of consumptions in most of the cases have no logic. It seems that they were chosen arbitrarily