PRACTICAL FILE
ON
SOFTWARE TESTING AND QUALITY ASSURANCE
[IT 414]

SUBMITTED TO:
DR. Rajni Sehgal
Associate Professor

SUBMITTED BY:
Suchinton Chakravarty
A2345920100
B.Tech. CSE 3rd year
6CSE-2EVE

COMPUTER SCIENCE AND ENGINEERING
AMITY SCHOOL OF ENGINEERING & TECHNOLOGY
AMITY UNIVERSITY,
UTTAR PRADESH

INDEX

Serial No.	Name of the experiment	Date	Maximum Marks	Marks Obtained	Signature
1.	Design test cases using Boundary value analysis by taking quadratic equation problems.				
2.	Design the test cases for login page of Gmail				
3.	Design test cases using boundary value analysis by the triangle problem(1-100) for Equilateral triangle , isosceles triangle and scalene triangle and not a triangle.				
4.	Design test cases using Equivalence class testing analysis by the triangle problem(1-100) for Equilateral triangle , isosceles triangle and scalene triangle and not a triangle.				
5.	Design test cases using Decision tables taking date problems and form the independent paths by calculating cyclomatic complexity using date problems.				
6.	Generate test cases for ATM machines.				
7.	Automated testing of login page of flight application GUI using UFT.				
8.	Automated testing for the flight gui application for keyword view.				
9.	Automated testing for the flight gui application for Data driver.				
10.	Automated testing for the flight gui application for Bitmap check point.				
11.	Automated testing for the flight gui application for GMAIL Login functionality test.				
12.	Automated testing for the flight gui application for If Else case.				

EXPERIMENT NO. 1

OBJECTIVE:

Design test cases using Boundary value analysis by taking quadratic equation problems.

SOFTWARE REQUIRED:

Turbo C++ Environment

THEORY:

Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behavior at the edge of the equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects.

It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

PROGRAM:

Consider a program for the determination of the nature of roots of a quadratic equation. Its input is a triple of positive integers (say a,b,c) and values may be from interval [0,200]. The program output may have one of the following words.

Not a quadratic equation
Real roots
Imaginary roots
Equal roots

Our objective is to design the boundary value test cases. Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. A boundary value analysis has a total of 4*n+1 distinct test cases, where n is the number of variables in a problem. Here we have to consider all three variables and design all the distinct possible test cases. We will have a total of 13 test cases as n = 3.

- Roots are real if (b2 – 4ac) > 0
- Roots are imaginary if (b2 – 4ac) < 0
- Roots are equal if (b2 – 4ac) = 0
- Equation is not quadratic if a = 0

How do we design the test cases ? For each variable we consider below 5 cases:

- amin = 0
- amin+1 = 1
- anominal = 50
- amax-1 = 99
- amax = 100

TEST CASES:

Project Name: Boundary value analysis
Created By: SUCHINTON CHAKRAVARTY
Creation Date:
Reviewed By: Dr Rajni Sehgal
Reviewed Date

Test Scenario ID	Test Scenario Description	Test Case ID	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Carry out boundary value analysis on given values	1001	To check the nature of roots	Input the values in the program and verify the output.	a= 0, b=50, c=50	Not Quadratic	Not Quadratic	Success
2	Carry out boundary value analysis on given values	1002	To check the nature of roots	Input the values in the program and verify the output.	a= 1, 0=50, c=50	Real Roots	Real Roots	Success
3	Carry out boundary value analysis on given values	1003	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=50, c=50	Imaginary Roots	Imagnary Roots	Success
4	Carry out boundary value analysis on given values	1004	To check the nature of roots	Input the values in the program and verify the output.	a= 99, 0=50, c=50	Imaginary Roots	Imaginary Roots	Success
5	Carry out boundary value analysis on given values	1005	To check the nature of roots	Input the values in the program and verify the output.	a= 100, c=50, b=50	magi nary Roots	maginary Roots	Success
6	Carry out boundary value analysis on given values	1006	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=0, c=50	maginary Roots	Imaginary Roots	Success

7	Carry out boundary value analysis on given values	1007	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=1, c-50	Imaginary Roots	Imaginary Roots	Success
8	Carry out boundary value analysis on given values	1008	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=99, c=50	Imaginary Roots	Imaginary Roots	Success
9	Carry out boundary value analysis on given values	1009	To check the nature of roots	Input the values in the program and verify the output.	a= 50, 0=100 c=50	Equal Roots	Equal Roots	Success
10	Carry out boundary value analysis on given values	1010	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=50, c=0	Real Roots	Real Roots	Success
11	Carry out boundary value analysis on given values	1011	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=50, c=1	Real Roots	Real Roots	Success
12	Carry out boundary value analysis on given values	1012	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=50, c=99	Imaginary Roots	imaginary Roots	Success
13	Carry out boundary value analysis on given values	1013	To check the nature of roots	Input the values in the program and verify the output.	a= 50, b=50, c=100	magi nary Roots	Imaginary Roots	Success

RESULT:

The result interprets that the program used conforms to Boundary Value Analysis.

EXPERIMENT NO. 2

OBJECTIVE:

Design the test cases for login page of Gmail

SOFTWARE USED:

Manual testing

PROCEDURE:

HOME PAGE:
test URL : https://www.gmail.com
Preconditions: Open Web browser and enter the given url in the address bar. Home page must be displayed. All test cases must be executed from this page.

OBSERVATION/OUTPUT:

Project Name: Valid username and password detection
Created By: Suchinton Chakravarty
Creation Date:
Reviewed By: Dr Rajni Sehgal
Reviewed Date

Test Scenario ID	Test Scenario Description	Test Case ID	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result
1	Verify the login functionality of Gmail page	1001	Enter a valid username and password	Enter username and password and click login	Email: abc1@gmail.com, Password:	Successful	Successful
2	Verify the login functionality of Gmail page	1002	Enter a valid username and pas sword	Enter username and password and click login	Email:arham.jan@gmail.com, Password:	Successful	Successful
3	Verify the login functionality of Gmail page	1003	Enter a valid username and password	Enter username and password and click login	Email: .///!!!@gmal co m, Password:	Unsuccessful	Unsuccessful

4	Verify the login functionality of Gmail page	1004	Enter a valid username and password	Enter username and password and click login	Email:, Password:	Unsuccessful	Unsuccessful
5	Verify the login functionality of Gma/ page	1005	Enter a valid username and password	Enter username and password and click login	Email: gmail.com@arham jain, Password:	Unsuccessful	Unsuccessful
6	Verify the login functionality of Gmail page	1006	Enter a valid username and password	Enter username and password and click login	Email: @gmail.com Password:	Unsuccessful	Unsuccessful
7	Verlfy the login functionality of Gmail page	1007	Enter a valid username and password	Enter username and password and click login	Email: prime@gmail.com, Password:	Successful	Successful
8	Verfy the login functionality of Gmail page	1008	Enter a valid username and password	Enter username and password and click login	Email: 123@gmail.com Password:	Unsuccessful	Unsuccessful
9	Verify the login functionality of Gmail page	1009	Enter a valid username and password	Enter username and password and click login	Email: ?00@gmail.com Password:	Unsuccessful	Unsuccessful
10	Verify the login functionality of Gmail page	1010	Enter a valid username and password	Enter username and password and click login	Email: arham jain@gmail.com, Password:	Unsuccessful	Unsuccessful
11	Verify the login functionality of Gmail page	1011	Enter a valid username and password	Enter username and password and click login	Email: Abc*l@gmal.com Password:	Successful	Successful
12	Verify the login functionality of Gmail page	1012	Enter a valid username and password	Enter username and password and click login	Email: jonny@.net, Password:	Unsuccessful	Unsuccessful

13	Verify the login functionality of Gmail page	1013	Enter a valid username and password	Enter username and password and click login	Email: gmallgmall.com, Password:	Unsuccessful	Unsuccessful

CONCLUSION:

Test cases have been formulated.

EXPERIMENT NO. 3

OBJECTIVE:

Design test cases using boundary value analysis by the triangle problem(1-100) for Equilateral triangle , isosceles triangle and scalene triangle and not a triangle.

SOFTWARE USED:

Manual testing

PROCEDURE:

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. The idea comes from the boundary. Given that we have a set of test vectors to test the system, a topology can be defined on that set.

OBSERVATION/OUTPUT:

TEST ID	A	B	C	EXPECTED OUTPUT	PROGRAM OUTPUT	TESTED OUTCOME
1	1	5	5	Isosceles	Isosceles	Pass
2	2	5	5	Isosceles	Isosceles	Pass
3	9	5	5	Isosceles	Isosceles	Pass
4	10	5	5	Not a Triangle	Not a Triangle	Pass
5	5	1	5	Isosceles	Isosceles	Pass
6	5	2	5	Isosceles	Isosceles	Pass
7	5	9	5	Isosceles	Isosceles	Pass
8	5	10	5	Not a Triangle	Not a Triangle	Pass
9	5	5	1	Isosceles	Isosceles	Pass
10	5	5	2	Isosceles	Isosceles	Pass

CONCLUSION:

Test cases have been formulated.

EXPERIMENT NO. 4

OBJECTIVE:

Design test cases using Equivalence class testing analysis by the triangle problem(1-100) for Equilateral triangle , isosceles triangle and scalene triangle and not a triangle.

SOFTWARE USED:

Manual testing

PROCEDURE:

In this method, the input domain of a program is partitioned into a finite number of equivalence classes such that one can reasonably assume, but not be absolutely sure, that the test of a representative value of each class is equivalent to a test of any other value.

OBSERVATION/OUTPUT:

	Project Name: Class Equivalence Problem(Triangle Problem)
	Created By: Suchinton Chakravarty
	Creation Date:
	Reviewed By: Dr Rajni Sehgal
	Reviewed Date:

Test Scenario ID	Test Scenario Description	Test Case ID	Test Case Description	Test Steps	Test Data	Expected Result	Final Result	Status
			VALID CLASSES		O<R<200			
1	Determining the type of triangle	1001	Entering the values of sides	Enter the length of sides of triangle	o- 100- 100	Not a triangle	Not a triangle	Successful
2	Determining the type of triangle	1002	Entering the values of sides	Enter the length of sides of triangle	50- 100- 100	Isosceles	Isosceles	Successful
3	Determining the type of triangle	1003	Entering the values of sides	Enter the length of sides of triangle	100- 100- 100	Equilateral	Equilateral	Successful

4	Determining the type of triangle	1004	Entering the values of sides	Enter the length of sides of triangle	150- 100- 100	Isosceles	Isosceles	Successful
5	Determning the type of triangle	1005	Entering the values of sides	Enter the length of sides of triangle	200- 100- 100	Isosceles	Isosceles	Successful
6	Determining the type of triangle	1006	Entering the values of sides	Enter the length of sides of triangle	100-0- 100	Not a triangle	Not a triangle	Successful
7	Determining the type of triangle	1007	Entering the values of sides	Enter the length of sides of triangle	100- 50- 100	Isosceles	Isosceles	Successful
8	Determining the type of triangle	1008	Entering the values of sides	Enter the length of sides of triangle	100- 150- 100	Isosceles	Isosceles	Succes sful
9	Determining the type of triangle	1009	Entering the values of sides	Enter the length of sides of triangle	100- 200- 100	Isosceles	Isosceles	Successful
10	Determining the type of triangle	1010	Entering the values of sides	Enter the length of sides of triangle	100- 100-0	Not a triangle	Not a triangle	Successful
11	Determining the type of triangle	1011	Entering the values of sides	Enter the length of sides of triangle	100- 100- 50	Isosceles	Isosceles	Successful
12	Determining the type of triangle	1012	Entering the values of sides	Enter the length of sides of triangle	100- 100- 150	Isosceles	Isosceles	Successful
13	Determning the type of triangle	1013	Entering the values of sides	Enter the length of sides of triangle	100- 100- 200	Isosceles	Isosceles	Successful

			INVALID CLASSES		R<O OR R>200			
14	Determining the type of triangle	1014	Entering the values of sides	Enter the length of sides of triangle	(-2)- (-1)- (3)	Scalene	Invalid Input	Unsuccessful
15	Determining the type of triangle	1015	Entering the values of sides	Enter the length of sides of triangle	(_1)_ 100- 100	Isosceles	Invalid Input	Unsuccessful
16	Determining the type of triangle	1016	Entering the values of sides	Enter the length of sides of triangle	100- (-100)- 100	Equilateral	Invalid Input	Unsuccessful
17	Determining the type of triangle	1017	Entenng the values of sides	Enter the length of sides of triangle	(-1). 0- 0	Not a triangle	Invalid Input	Unsuccessful
18	Determning the type of triangle	1018	Entering the values of sides	Enter the length of sides of triangle	100- 150- (-250)	Scalene	Invalid Input	Unsuccessful

CONCLUSION:

Test cases have been formulated.

EXPERIMENT NO. 5

OBJECTIVE:

Design test cases using Decision tables taking date problems and form the independent paths by calculating cyclomatic complexity using date problems.

SOFTWARE REQUIRED:

Manual Testing

THEORY:

Decision tables are a precise yet compact way to model complicated logic. Decision tables, like flowcharts and if-then-else and switch-case statements, associate conditions with actions to perform, but in many cases do so in a more elegant way. Decision tables, especially when coupled with the use of a domain-specific language, allow developers and policy experts to work from the same information, the decision tables them. Tools to render nested if statements from traditional programming languages into decision tables can also be used as a debugging tool. Decision tables have proven to be easier to understand and review than code, and have been used extensively and successfully to produce specifications for complex systems.

Printer troubleshooter

		Rules							
Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

TEST CASES:

The input domain can be divided into following classes:

I1= {M1: month has 30 days}
I2= {M2: month has 31 days except March, August and January}
I3= {M3: month is March}
I4= {M4: month is August}
I5= {M5: month is January}
I6= {M6: month is February}
I7= {D1: day = 1}

I8= {D2: $2 \leq day \leq 28$}

I9= {D3: day = 29}
I10= {D4: day = 30}
I11= {D5: day = 31}
I12= {Y1: year is a leap year}
I13= {Y2: year is a common year}

The decision table is given below:

Sr. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C_1: Months in	M_1	M_1	M_1	M_1	M_1	M_1	M_1	M_1	M_1	M_1	M_2	M_2	M_2	M_2	M_2
C_2: Days in	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4	D_5	D_5	D_1	D_1	D_2	D_2	D_3
C_3: year in	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1
a_1: Impossible									X	X					
a_2: Decrement day			X	X	X	X	X	X					X	X	X
a_3: Reset day to 31	X	X													
a_4: Reset day to 30											X	X			
a_5: Reset day to 29															
a_6: Reset day to 28															
a_7: decrement month	X	X									X	X			
a_8: Reset month to December															
a_9: Decrement year															

Sr. No.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
C_1: Months in	M_2	M_2	M_2	M_2	M_2	M_3	M_3	M_3	M_3	M_3	M_3	M_3	M_3	M_3	M_3
C_2: Days in	D_3	D_4	D_4	D_5	D_5	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4	D_5	D_5
C_3: year in	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2
a_1: Impossible															
a_2: Decrement day	X	X	X	X	X			X	X	X	X	X	X	X	X
a_3: Reset day to 31	X	X													
a_4: Reset day to 30															

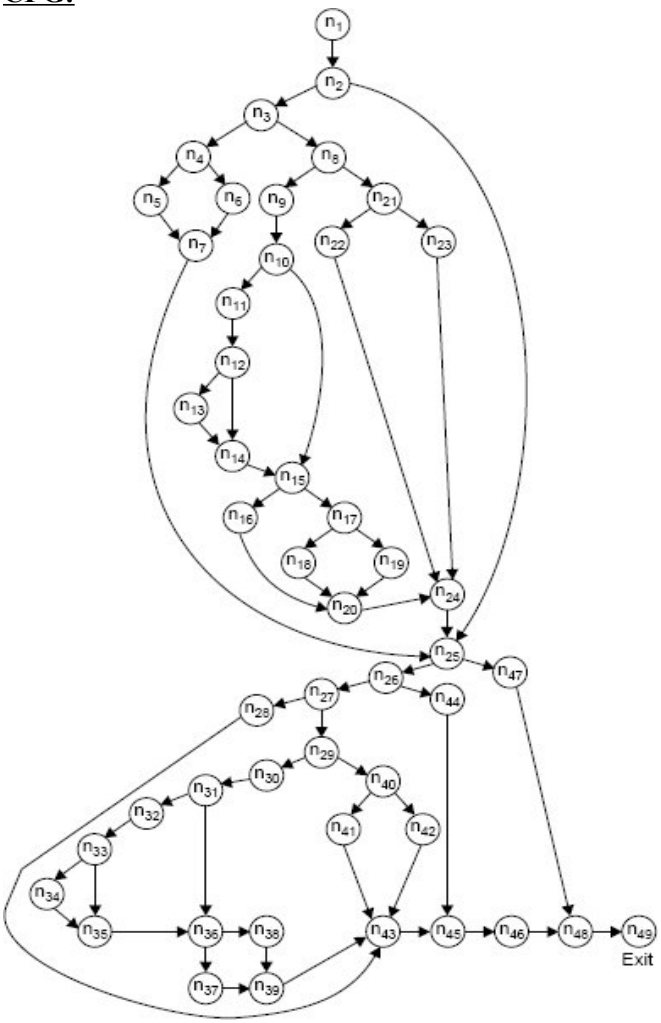
a_5: Reset day to 29						X									
a_6: Reset day to 28							X								
a_7: decrement month						X	X								
a_8: Reset month to December															
a_9: Decrement year															

Sr. No.	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
C_1: Months in	M_4	M_4	M_4	M_4	M_4	M_4	M_4	M_4	M_4	M_4	M_5	M_5	M_5	M_5	M_5
C_2: Days in	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4	D_5	D_5	D_1	D_1	D_2	D_2	D_3
C_3: year in	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1
a_1: Impossible															
a_2: Decrement day			X	X	X	X	X	X	X	X			X	X	X
a_3: Reset day to 31	X	X									X	X			
a_4: Reset day to 30															
a_5: Reset day to 29															
a_6: Reset day to 28															
a_7: decrement month	X	X													
a_8: Reset month to December											X	X			
a_9: Decrement year											X	X			

Sr. No.	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
C_1: Months in	M_5	M_5	M_5	M_5	M_5	M_6	M_6	M_6	M_6	M_6	M_6	M_6	M_6	M_6	M_6
C_2: Days in	D_3	D_4	D_4	D_5	D_5	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4	D_5	D_5
C_3: year in	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2
a_1: Impossible											X	X	X	X	X
a_2: Decrement day	X	X	X	X	X			X	X	X					

a_3: Reset day to 31						X	X								
a_4: Reset day to 30															
a_5: Reset day to 29															
a_6: Reset day to 28															
a_7: decrement month						X	X								
a_8: Reset month to December															
a_9: Decrement year															

CFG:



RESULT:

The result interprets that the program used conforms to the Decision Table.

EXPERIMENT NO. 6

OBJECTIVE:
Generate test case for ATM machine

SOFTWARE USED:
Manual testing

PROCEDURE:
An ATM card is swapped and checked for errors.

OBSERVATION/OUTPUT:

ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Output	Actual Output	Status	Comments
ATM_01	Verification of User Interface of ATM machine	N/A	N/A	N/A	User Interface is OK, asks to insert card	User Interface is OK, asks to insert card	Pass	
ATM_02	Verification of Card validation	N/A	a. Insert card to the machine b. Enter PIN	PIN entered: 0000	A welcome message pops up	As expected	Pass	
ATM_03	Verification of user controls	Card and its PIN are entered	a. Insert card to the machine b. Enter PIN c. Press button for balance check	PIN entered: 0000	Button is working properly and account balance is shown	As expected	Pass	
ATM_04	Checking the font size and text color	Card and its PIN are entered	a. Insert card to the machine b. Enter PIN c. Press all non transaction buttons	PIN entered: 0000	User should read everything with ease	As expected	Pass	N/A
ATM_05	Verification of Language Selector	N/A	Press on the Language Selector button and choose any language	N/A	Available language lists should be displayed	There is no language selector button	Fail	

CONCLUSION:
Test cases have been formulated and ATM cards have been tested.

EXPERIMENT NO. 7

OBJECTIVE:
Automated testing of login page of flight application GUI using UFT.

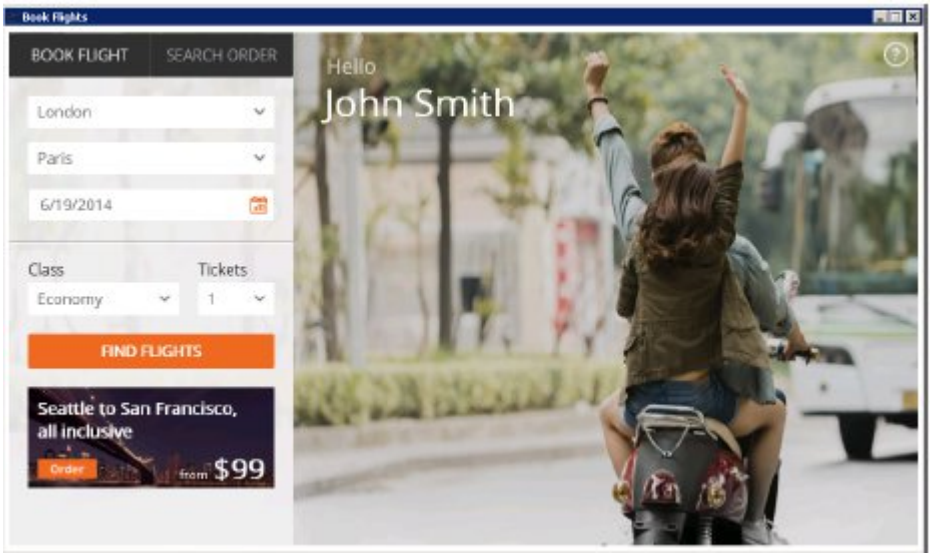
SOFTWARE USED:
UFT

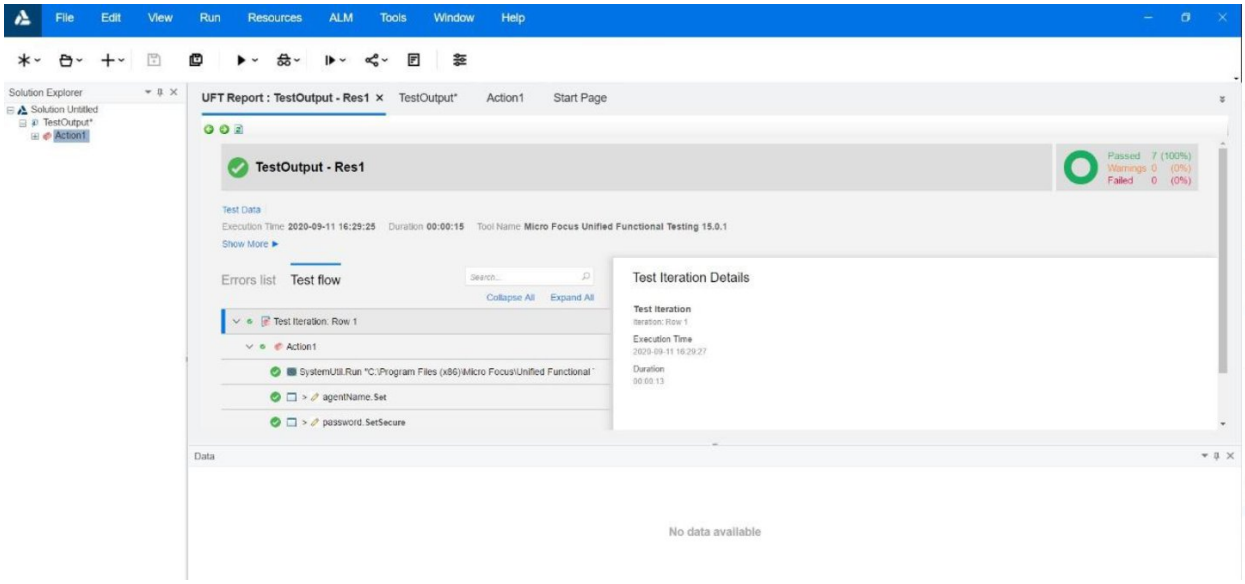
PROCEDURE:
1. Open the flight reservation application.
   Open both the Book Flights (GUI) layer and the Flights API (service) layer:
The Book Flights layer is available at Start > All Programs > HP Software > HP
   Unified Functional Testing > Sample Applications > Flight GUI.
The Flights API layer is available at Start > All Programs > HP Software > HP
   Unified Functional Testing > Sample Applications > Flight API.
2. Log in to the Book Flights application.
username: john
password: hp

3. Explore the application layers.
For Book Flights layer
Enter the requested information or selections on each page to follow the reservation process. As you navigate through the application, consider what user actions you might want to test, and which objects you would need to create to set up your test.

4. Exit your application browsing session.

OBSERVATION/OUTPUT:





CONCLUSION:

Automated testing of login page of flight application GUI using UFT run successfully.

EXPERIMENT NO. 8

OBJECTIVE:

Automated testing for the flight gui application for keyword view.

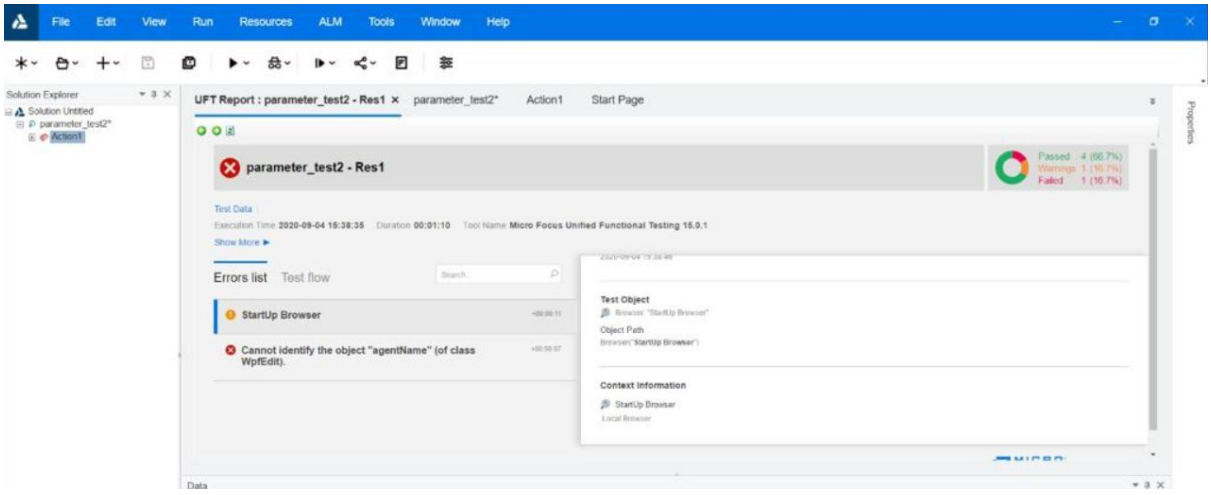
SOFTWARE USED:

UFT

PROCEDURE:

1. Start UFT and open the Book Flights test.
2. Open the Select Flight action.
3. In the opened Keyword View window, select the Item by directly clicking on the Login row. The Item list opens with object repository and other options. Here, the first test object is the Welcome: Mercury Tours browser object.
4. select “Object from repository” to open the Select Test Object dialog box. Select Test Object dialog box opens. In Select Test Object dialog box, expand the test object tree and select username and click “OK”, In Select Test Object dialog box, expand the test object tree and select username and click “OK”.
5. To add the next step, click below the username row of the Item column. The Item list opens, listing the previous step test object.
6. Select “password” from the Item list. This adds only one new row because the object shares the same parent objects as the previous step.
7. Fifth, use the HP Password Encoder application to create an encoded password. To do so, go to Tools > Password Encoder. The Password Encoder dialog box opens.
8. add the last step in the Login action .Click on the last step, in the Item column to add the next step. The Item list opens, select sign-in from the item list.
9. save the test by going to File > Save.

OBSERVATION/OUTPUT:



CONCLUSION:

Automated testing for the flight gui application for keyword view ran successfully.

EXPERIMENT NO. 9

OBJECTIVE:

Automated testing for the flight gui application for Data driver.

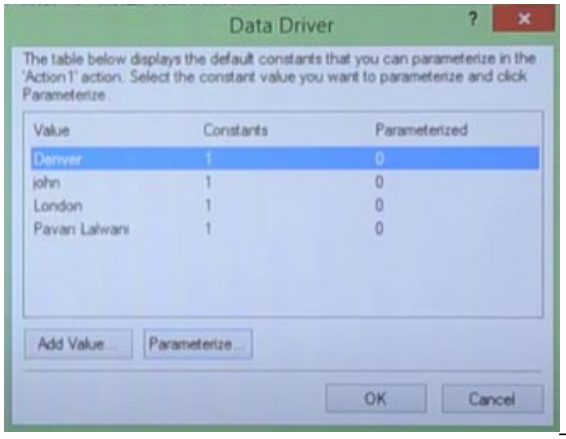
SOFTWARE USED:

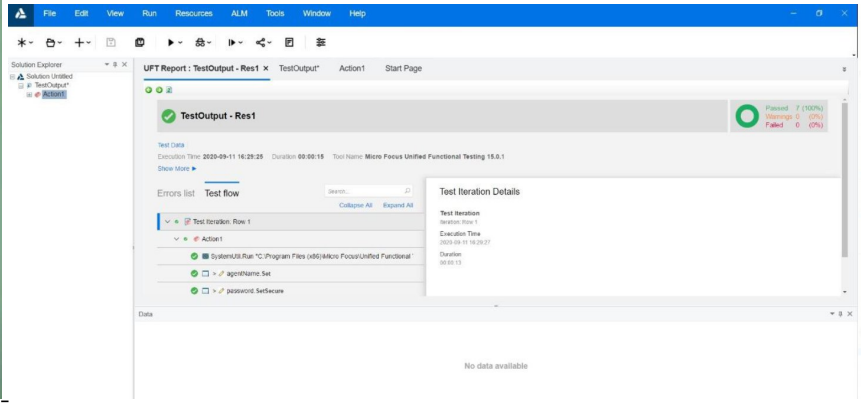
UFT

PROCEDURE:

1. Start UFT and open the Book Flights test.
2. Open the Select Flight action.
3. Choose Tools > Data Driver. Quick Test scans the test for constants before the Data Driver opens. The data driver displays the Constants list for the action. For each constant value, it displays the number of times the constant value appears in action.
4. If you want to parameterize a value that is not currently displayed in the list click Add Value. The Add Value dialog box opens. Enter a constant value in the dialog box and click OK. The constant is added to the list.
5. Select the value you want to parametrize from the Constants list and click Parametrize. The data driver wizard opens.
6. If you selected Step-by-step parametrization, click Next. The parametrize the selected step screen opens. If you selected Parametrize all, the parameter option is enabled in the Configure value area. Select your parametrization preferences the same way that you would for an individual step.
7. In the Step to parametrize area the first step with an object property or checkpoint value containing the selected value is displayed in the test tree on the left.
8. Click next to parametrize the selected step and view the next step.
9. Click Skip if you do not want to parametrize the selected step.
10. Click Finish to apply the parametrization settings of the current step to all remaining steps containing the selected value. The Data Driver Wizard closes and the Data Driver main screen shows how many occurrences you selected to parametrize and how many remain as constants.
11. When you are finished with the parametrization constants click OK. The parametrization options you selected are applied to your action.

OBSERVATION/OUTPUT:





CONCLUSION:

Automated testing for the flight gui application for Data driver run successfully.

EXPERIMENT NO. 10

OBJECTIVE:

Automated testing for the flight gui application for Bitmap checkpoint.

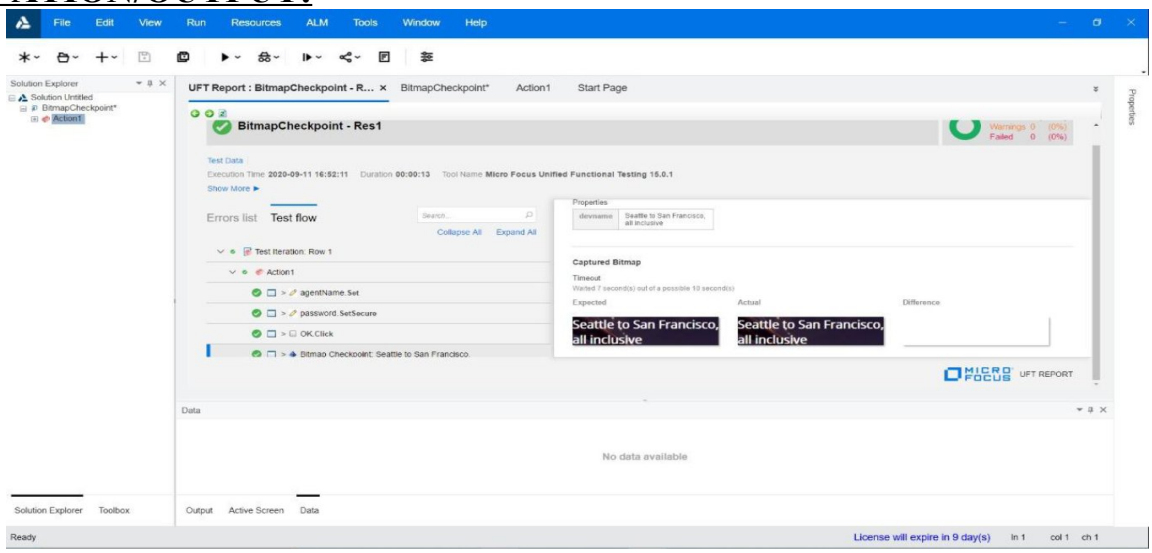
SOFTWARE USED:

UFT

PROCEDURE:

1. Start UFT and open the Book Flights test.
2. Display the action in which you want to add a checkpoint.
3. Open the Select Flight action.
4. Enter the login information:
   o Username: john
   o Password: hp
   a. Click OK. The Flight Finder page opens.
   b. Enter the flight search details:
   o Departure City: Los Angeles
   o Arrival City: Sydney
   o Date: tomorrow's date
   o Class: Business
   o Tickets: 2
   c. Click the Find Flights button. The Select Flight page opens.
   d. In the Select Flight page, select the first row and click Select Flight. The Flight Details page opens.
5. Create a bitmap checkpoint.
   a. If the Editor is displayed, click the Keyword View button to display the Keyword View.
   b. In the Keyword View, select the passengerName row by clicking in the right margin of the grid.
6. Select Design > Checkpoint > bitmap Checkpoint.
7. Save the test.

OBSERVATION/OUTPUT:



CONCLUSION:

Automated testing for the flight gui application for Bitmap checkpoint run successfully.

EXPERIMENT NO. 11

OBJECTIVE:

Automated testing for the flight gui application for GMAIL Login functionality test.

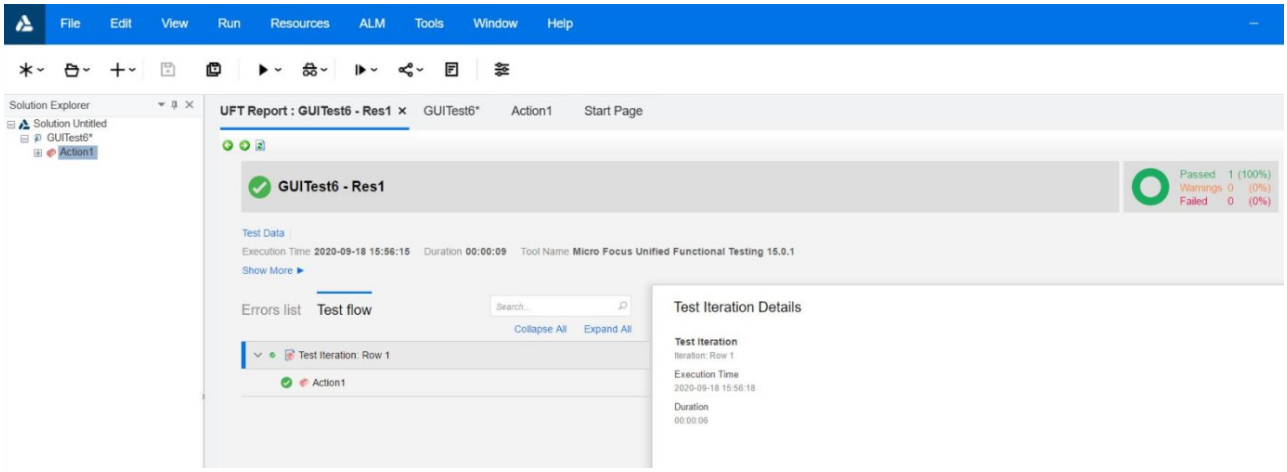
SOFTWARE USED:

UFT

PROCEDURE:

1. Start UFT
2. make sure the default webpage selected is **www.gmail.com**
3. start the recording before opening the webpage.
4. Once the recording is started you have to checkpoint each step.
5. create the standard checkpoint on the first webpage of **www.gmail.com** on the word GOOGLE.
6. Now enter your GMAIL login username and click on next. Now again create a standard checkpoint on your name and enter the password and then click next.
7. Now your Gmail inbox will open again you have to create a standard checkpoint on word Gmail and then logout.
8. after logging out now u need to create a Bitmap checkpoint so click on your Gmail icon image and make a bitmap checkpoint on it. After this, close the webpage.
9. Now you can stop the recording. Once the recording is stopped, run your program and it will show you all the steps which you have performed using different checkpoints.
10. Save the test.

OBSERVATION/OUTPUT:



CONCLUSION:

Automated testing for the flight gui application for Gmail login functionality test run successfully.

EXPERIMENT NO. 12

OBJECTIVE:

Automated testing for the flight gui application for If Else case.

SOFTWARE USED:

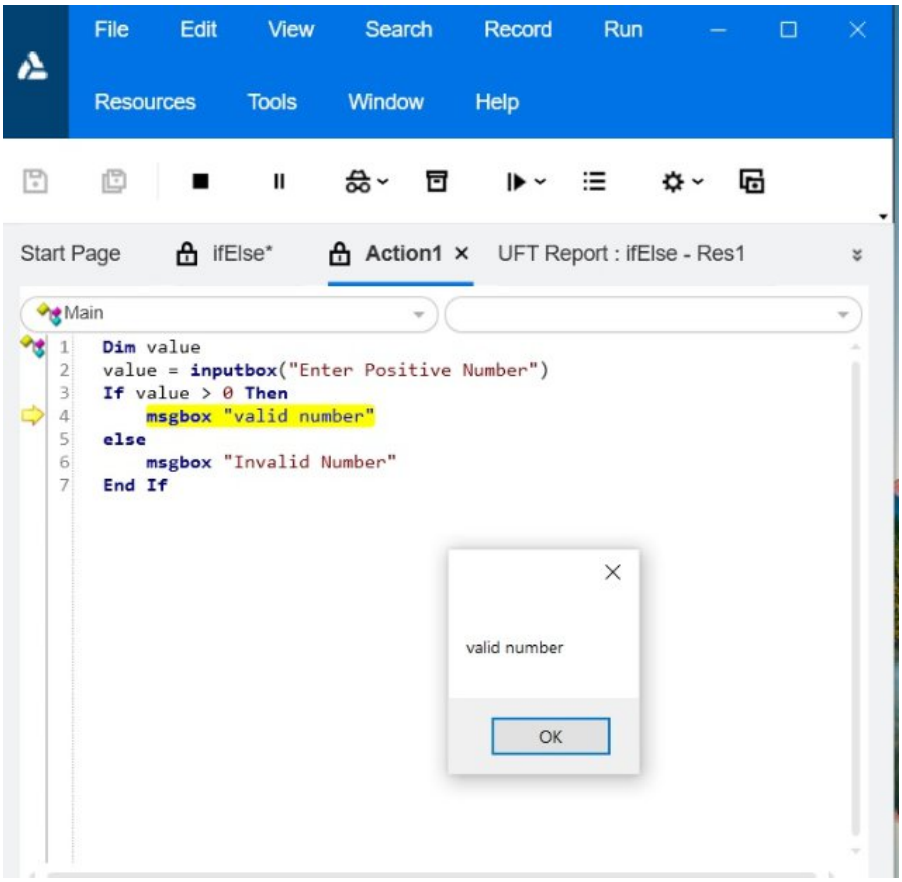
UFT

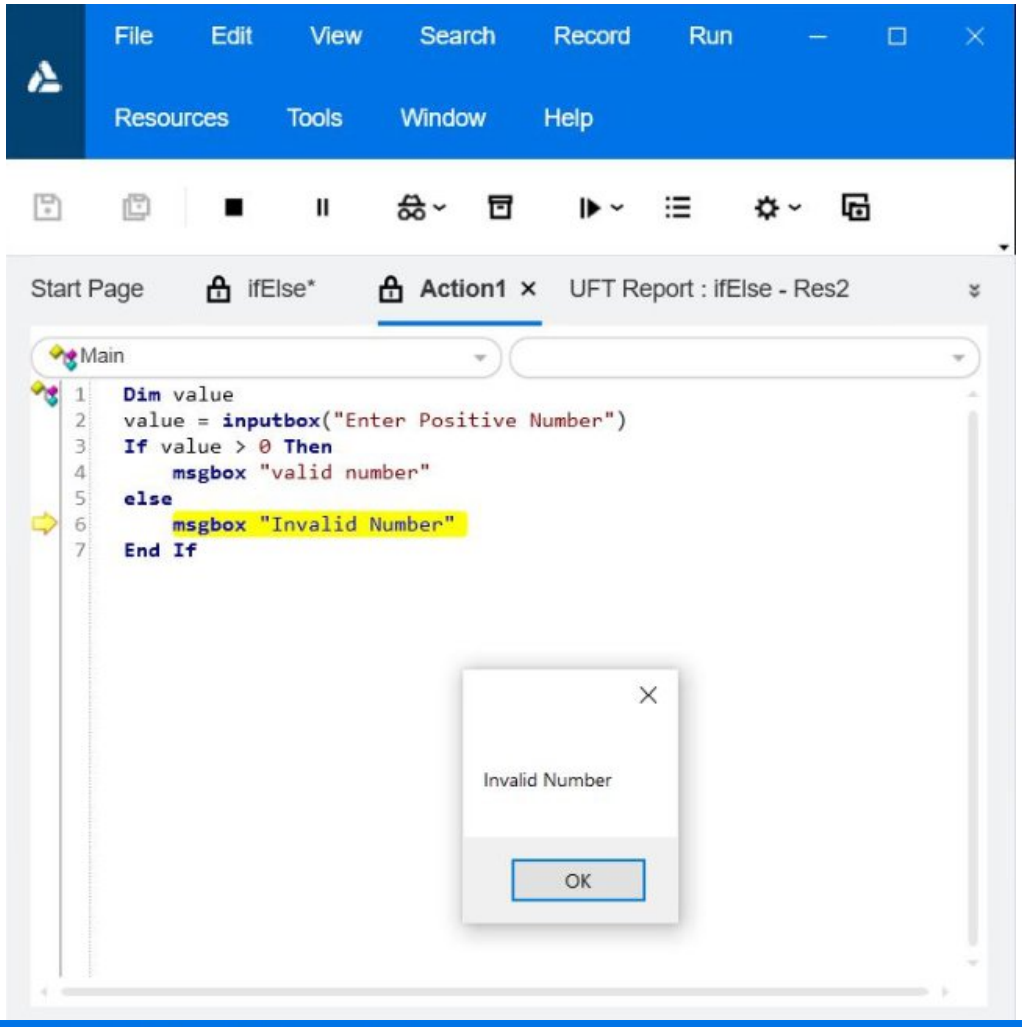
PROCEDURE:

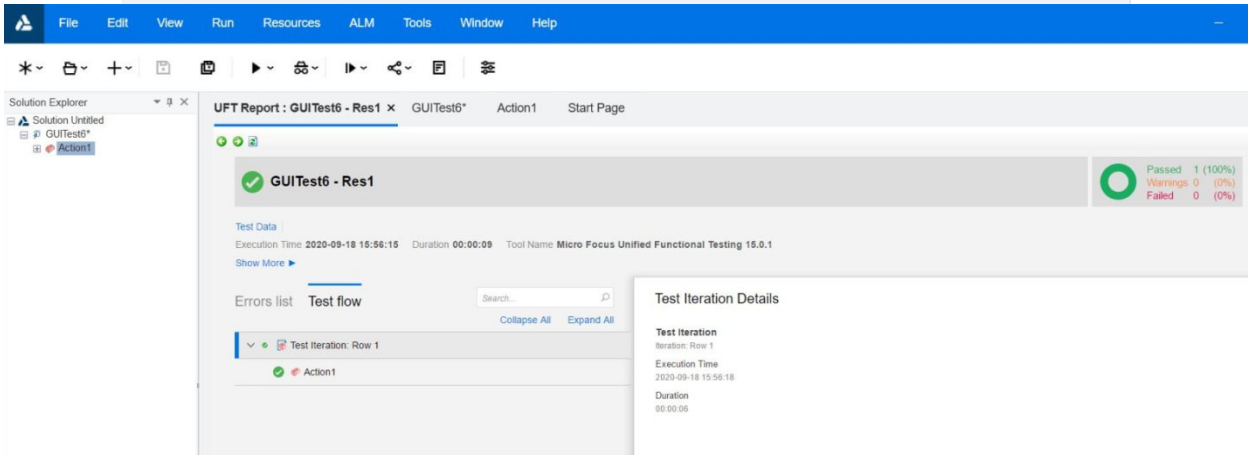
1. Start UFT
2. enter the given code
3. Dim value
   value = inputbox("Enter Positive Number")
   If value > 0 Then
   msgbox "valid Number"
   else
   msgbox "Invalid Number"

   End If
4. Now run the program and enter the positive number for example for 90 the output will be valid number, where as if you enter -90 then the output will be Invalid number.
5. Save the test.

OBSERVATION/OUTPUT:







CONCLUSION:

Automated testing for the flight gui application for If Else case run successfully.