SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
(An Autonomous Institution approved by UGC and affiliated to JNTUH))
(Accredited by NAAC with 'A" Grade, Accredited by NBA of AICTE and
Recipient of World Bank under TEQIP-I and II)
Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301

LAB MANUALS

For

SOFTWARE TESTING & INFORMATION SECURITY

(From Page No. 2 to159)

&

DATA WAREHOUSING & DATA MINING

(From Page No. 160 to End)

FOR
B. Tech. IV year - I Semester
COMPUTER SCIENCE ENGINEERING

SNIST
Sreenidhi Institute of Science and Technology
'Because, Life is all about taking the right decisions
A U T O N O M O U S

DEPARTMENT OF
COMPUTER SCIENCE ENGINEERING
JUNE 2018

SREENIDHI INSTITUTE OF SCIENCE & TECHNOLOGY
(AN AUTONOMOUS INSTITUTION UNDER JNTUH)
(Approved by AICTE & Aided by World Bank under TEQIP)
Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301.

LAB MANUAL
For

SOFTWARE TESTING AND INFORMATION SECURITY

For
B. Tech. IV year - I Semester
CSE BRANCH

SNIST
Sreenidhi Institute of Science and Technology
Because, Life is all about taking the right decisions
A U T O N O M O U S

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Department of Computer Science & Engineering is established in the year 1997 to meet the requirements of the emerging IT industry/discipline. The Vision and the Mission of the department are

Vision

To emerge as a leading department in Technical Education and Research in Computer Science and Engineering with focus to produce professionally competent and socially sensitive engineers capable of working in global environment.

Mission

I. To prepare Computer Science and Engineering graduates to be a life long learner with competence in basic science & engineering and professional core, multidisciplinary areas , with continuous update of the syllabus, so that they can succeed in industry as an individual and as a team or to pursue higher studies or to become an entrepreneur.

II. To enable the graduates to use modern tools, design and create novelty based products required for the society and communicate effectively with professional ethics.

III. To continuously engage in research and projects development with financial management to promote scientific temper in the graduates and attain sustainability.

B.Tech (CSE) Programme Educational Objectives (PEOs)

A. Graduates will have a strong foundation in fundamentals of mathematics, Physics, Chemistry, Computer Science and basic engineering knowledge with abilities for analysis of the problem and to design, development of solutions and to arrive at an optimal solution using modern tools which help them to be employable.

B. Ability to work in a team/ lead a team which needs effective communication skills and knowledge of project management, finance and entrepreneurial abilities.

C. Graduates should have abilities to conduct investigation of complex problems and attitude for lifelong learning skills which will enable them to pursue advanced studies, Research and Development.

D. The graduates must be aware of the engineering professional ethics, the impact of engineering profession on the society and the need for environmental protection and sustainable development.

The Programme Outcomes (POs) of the B.Tech (CSE) programme, which every graduate must attain, are listed below:

a. An ability to apply knowledge of basic sciences, mathematics and engineering in the area of Computer Science.

b. An ability to design, implement and evaluate a software or software / hardware system to meet the desired needs within realistic constraints such as space and time.

c. An ability to use the techniques, skills, and modern engineering tools such as software testing tools, data warehousing and mining tools, necessary for practice as a CSE professional.

d. An ability to analyze and solve open-ended problems using mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices and to arrive at an optimal solution.

e. To understand principles of engineering, entrepreneurship with emphasis on women, and financial management through relevant management courses to demonstrate knowledge in the conceptualization and realizing group projects, mini & main projects.

f. An ability to function effectively as individual and as a member or leader in diverse team in achieving multidisciplinary tasks.

g. Learn to communicate effectively on complex engineering activities through report writing, experimental work, assignments, seminars, group projects, mini & main projects.

h. To recognize the need for and have the preparation and ability to be a life-long learner through the courses such as seminars & projects.

i. An ability to identify, formulate and analyze engineering problems.

j. An ability to conduct investigation of complex problems in multidisciplinary areas.

k. An understanding of professional ethics and responsibilities.

l. An engineer should be aware of social, safety, cultural and information security issues and also responsibilities relevant to professional practice and skills.

m. An ability to understand the impact of environmental protection and sustainable development.

a	b	c	d	e	f	g	h	i	j	k	l
	x	x		x							

Syllabus for B. Tech. IV Year I semester
Computer Science and Engineering
SOFTWARE TESTING AND INFORMATION SECURITY LAB

Code: 5EC75

L	T	P/D	C
-	-	4	2

Course Objectives:

- In software testing lab the various manual and automation testing processes are carried out to efficiently learn the testing activities.
- Both commercial and open source testing tools are being taught to better the software testingin detail.
- According the software industry requirements the testing tools are taught so that the students can directly make use of testing tools in industry.
- Implement various testing techniques and to make a thorough study on various testing tools.

Course Outcomes: After completing this course, student shall be able to

1. Students prepare Test Plan document and write Test Cases for Small scale Project (Like for their B.Tech IV Year Project or Post-Graduate Projects), they are learn how to Analyze SRS document in order to prepare Test Plan Document.
2. Students demonstrate skills to use modern software testing tools (EX: QTP, Bugzilla, Selenium, Test Director and Quality Center) and test application (web, Window application) by using the tools.
3. Students demonstrate the ability to differentiate between different Testing tools present in the market (like functional testing tools, Test Management Tools, Bug Tracking Tools and Performance Testing Tools) and prepare Test Plan document and write Test Cases for Small scale Project (Like for their B.Tech IV Year Project or Post-Graduate Projects).

SOFTWARE TESTING LAB:

Week 1

1. Write programs in 'C' Language to demonstrate the working of the following constructs:
   i) do…while ii) while…do iii) if …else iv) switch v) for
2. A program written in 'C' language for matrix multiplication fails" Introspect the causes for its failure and write down the possible reasons for its failure.

Week 2

3. Take any system (e.g. ATM system) and study its system specifications and report the various bugs.
4. write the test cases for any known application (e.g. Banking application)

Week 3

5. Create a test plan document for any application (e.g. Library management system)
6. Overview of any Test Management Tools (e.g. Test Director)

Week 4 & 5

7. Study of any Functional Testing Tools (UFT,QTP)

Week 6

8. Study of any bug tracking tool (e. g. Bugzilla, Bug bit)
9. Overview of Performance Testing Tools (Ex: Load runner)
10. Study of Selenium IDE (open source testing tool)

Information Security Lab:

1. Implement RSA algorithm

   (a) Generate Public key and Private key pair

   (b) Generate Ciphertext for the Plaintext

   (c) Obtain the Plaintext from the Ciphertext

2. Implement DES

   (a) Generate Cipher text for the given Plaintext

   (b) Retrieve the Plaintext from the given Ciphertext

3. Implement Diffie Hell man Algorithm and generate Secret Key
4. Implement Hash Algorithm
5. Generate Digital Signature
6. Implement Digital Envelope

CODING STANDARDS

1 Introduction

The purpose of these coding standards is to facilitate the maintenance, portability, and reuse of custom C, C++, and Java source code developed by students in the JD Edwards Honor Program, University of Nebraska-Lincoln.

You must receive explicit permission whenever the standards are not followed, and add a comment with the reason for non-conformance.

Language specific information will be denoted by parentheses containing the name of the language. Within tables, "n/a" denotes "not applicable."

1.1 Overview

Section 2 describes file organization, including file content, source file layout, header file layout, and header file guard. Section 3 describes the naming conventions for files, attributes, variables, methods, namespaces, etc. Section 4 describes style guidelines for the source files and header files. Every assignment should observe the standards described from section 2 to 4.

1.2 References and Applicable Documents

GNU standard: **http://www.gnu.org/prep/standards_toc.html**
IBM Standard: http://oss.software.ibm.com/icu/userguide/conventions.html
NASA Standard: **http://ccs.hst.nasa.gov/ccspages/policies/standards/coding_standards.html**
Javasoft Standard: http://Java.sun.com/products/jdk/Javadoc
Possibility Standard: **http://www.possibility.com/Cpp/CppCodingStandard.html**

2 File Organization

This chapter explains what belongs in the source and header files (section 2.1), and how the source and header files should be organized (sections 2.2 and 2.3).

2.1 File Contents

Files should be used to organize related code modules, either at the class (for C++ and Java) or function (for C/C++) level. Always use one file (or header/source file pair) per class or set of logical functions. Of course, nested classes are allowed in Java. Table 2.1 gives the details about what goes in each type of file.

	C	C++	Java
Class declaration	n/a	header file (.h)	n/a
Class implementation	n/a	source file (.cpp)	source file (.java)
Struct declaration	header file (.h)	header file (.h)	n/a
Struct implementation	source file (.c)	source file (.cpp)	n/a
Function prototypes	header file (.h)	header file (.h)	n/a

Function definitions	source file (.c)	source file (.cpp)	n/a

Table 2.1 File Contents

2.2 Source File Layout

Source files should contain the following components **in the order** shown in Table 2.2 (if they are used). Some are sections in themselves (like classes) and may have separate ordering within that scope. Some components may show up in a code block for purposes of scope.

File contents	**C**	**C++**	**Java**
Beginning Comments	X	X	X
Package imports	n/a	n/a	X
Class Document Comments/Prologue	X	X	X
C, C++ Prologue	X	X	n/a
System #includes *****	X	X	n/a
Application #includes *****	X	X	n/a
External functions	X	X	n/a
External variables	X	X	n/a
Constants	X	X	X
Static variable initializations	X	X	X
Class declaration	n/a	n/a	X
Public methods	n/a	X	X
Protected methods	n/a	X	X
Package methods	n/a	n/a	X
Private methods	n/a	X	X
Functions	X	X	n/a

+ ***(Java)*** also called Java Prologue

****(C, C++)*** When it's possible to put a needed #include line in the source file instead of in the header file, do so. This will reduce unnecessary file dependencies and save a little compile time.

Table 2.2. Source File Layout

2.3 Header File Layout (C, C++ only)

Header files should contain the following components **in the order** shown in the Table 2.3 (note that Java does not use header files). Some are sections in themselves (like classes) and may have separate ordering within that scope.

All header files should contain a **file guard** mechanism to prevent multiple inclusion. Below is an example:

```
#ifnde                    // first line of the header
f      MeaningfulNameH    file
#defin                    // second line of the
e      MeaningfulNameH    header file
.
.                         // body of the header file
```

```
.
       //                  // last line of the header
#endif MeaningfulNameH     file
```

File contents	C	C++
File guard	X	X
Prolog	X	X
System #includes	X	X
Application #includes	X	X
#defines	X	X
Macros	X	X
external functions	X	X
external variables	X	X
Constants	X	X
Structs	X	X
Class declaration	n/a	X
Public methods	n/a	X
Protected methods	n/a	X
Private methods	n/a	X
Inline method definitions *	n/a	X
Function declarations	X	X

**(C++)* Small inline methods may be implemented in the class definition.

Table 2.3 Header File Layout

3 Naming Conventions

We use a naming convention to make sure that all names in a project are defined and used in a consistent way, resulting in more readability and understandability. The most important principle for naming everything (files, classes, variables, etc.) is " short but descriptive" . In other words, avoid names that are extremely long, but provide enough detail so it is clear what the name represents. Abbreviations and contractions are discouraged.

For example:

```
public int setNumberReceiver(int aInt); // Recommended

public int setNumberRcv(int aInt); // Not recommended--uses
abbreviations
public int theNumberOfStudentsInTheJDEdwardsHonorsProgramClass;
                                        // Way too long to be
                                        useful
```

3.1 General Conventions

The general naming conventions you should follow are summarized on Table 3.1.

3.2 Valid Characters

All names should begin with a letter. Individual words in compound names should be differentiated by capitalizing the first letter of each word. The use of special characters (anything other than letters, digits and underscores) is discouraged.

Identifier	C	C++	Java
package	n/a		ShortName *
class, union, struct	EachWordCapitalized		
interface	n/a		EachWordCapitalized
typedef	EachWordCapitalized		n/a
enum	EachWordCapitalized		n/a
pointers	namePtr		n/a
function, method	internalWordsCapitalized		
class attribute	n/a	internalWordsCapitalized	
convert method	n/a	toX	
accessor method	n/a	getX, setX	
object, variable	internalWordsCapitalized		
#define, macro	ALL_CAPS_AND_UNDERSCORES		n/a
const, static final	ALL_CAPS_AND_UNDERSCORES		
source file	.c	.cpp	.Java
header file	.h		n/a

*If you must combine several words for package name, use format EachWordCapitalized.

Table 3.1 Naming Conventions

3.3 File Names

File names must conform to the following guidelines.

1. Names should be descriptive and relate directly to the purpose of the file.
2. Do not use spaces between words.
3. Do not use more than one period per filename (e.g. my.file.cpp is bad).
4. Capitalize the first letter of each word.
5. No more than four words in one name.
6. Underscores can be used (e.g. Document_Style.doc).

3.4 Function/Method Names

Function names should be an action verb. Functions with Boolean return types should be named with the "is" prefix, as in "isEmpty()" or "isRed()."

3.5 Namespaces/Packages

(C++) There are no requirements for namespaces in C or C++.

(Java) Java packages group classes of related functionality. Package source and class files then reside in a convenient hierarchical directory structure that maps directly to the package name. For example, package edu.unl.jdehp.schmoe.joe corresponds to the directory " … \edu\unl\jdehp\schmoe\joe\" .

4 Style Guidelines

The primary purpose of style guidelines is to facilitate long-term maintenance. During maintenance, programmers who are usually not the original authors are responsible for understanding source code from a variety of applications. A common presentation format reduces confusion and speeds comprehension of the code.

4.1 Lines

At most one statement is allowed per line. Keep the code as understandable as possible.

All lines should be displayable without wrapping on an 80-character display. When an expression will not fit on a single line, break it according to these general principles:

- Break after a comma.
- Break before an operator.
- Prefer higher-level breaks to lower-level breaks (see example).
- Align the new line with the start of the expression at the same level on previous

line. Here are some examples.

```
someMethod( firstInteger, secondInteger, thirdInteger,
          FourthInteger);            //Break at comma, and align

someMethod1( firstInteger,
           SomeMethod( secondInteger, thirdInteger));
                                     //Break at higher-level.

firstInteger = secondInteger * ( thirdInteger + fourthInteger)
             + fifthInteger ;        //Break before operator.
```

4.2 Comments

There are three types of comment delimiters used in C, C++, and/or Java. Table 4.1 shows each, including which can be used in each language.

(C++, Java) Single line comments should use the single line comment delimiter // whenever possible.

When commenting file prologues, classes, methods, and interfaces, you should follow the

Type	Begins with	Ends with	C	C++	Java
Single line	//	End of line		**X**	**X**
Multiple line	/*	*/	**X**	**X**	**X**
Javadoc[#]	/**	*/	**X***	**X***	**X**

[#]Javadoc is more fully discussed in chapter 5.

The Javadoc tool does not work with C/C++ code, but since the beginning delimiter has prefix /, this type is still valid in these languages.

Table 4.1 Commenting types

The subsequent sections describe in detail the commenting you must include in your code. Sections 4.2.1 and 4.2.2 describe the comments that every file must begin with, and sections 4.2.3 and 4.2.4 describe the comments used for classes, methods, etc.

4.2.1 Beginning Comments

All source files should begin with a comment that lists the filename, course number, author(s), version, and date. If the code is maintained by a different person/group from the original author(s), consider it a new version and state maintainers name(s). You should **not** use the *Javadoc* format for the beginning comments. Some of the information will be duplicated in the prologue, but do not worry about it. If you are using a software configuration management tool, you do not need to include this. For example:

```
/*
 * SomeClass.java
*
 * Written by : John Doe, Jane Smith
 * Written for: JDE155
 * Date       : July 30, 2002
 * Version    : 1.0
 *
 * Modified by: Pete Jones, Mary Nelson
 * Written for: JDE155
 * Date       : August 15, 2002
 * Version    : 2.0
 */
```

4.2.2 Prologue

The second type of comment every file should have is the prologue. The purpose of prologue is to provide both users and maintainers of the code with a better understanding of what the code does. In some cases, developers can write a small tool to extract information from the prologues, so observance to the prologue standard is very important. Prologues should always be written with *Javadoc*– type comments, and should **only** contain the following information:

- **Description**. A "big-picture" description (or responsibility) of code, including collaborations with other files.

- **Author**.

- **Version.** The current version number of the code.

- **See**. A list of related class(es), each on a separate

line. Here is an example:

```
/**
 * An class for doing something. A description should be given
   that
 * will give the user enough information to understand the
   basics.
 * It will be initialized by some UI class.
 *
 * Version 1.2: Fix bug #3
 * Version 1.3: Fix bug #5
*
             Joe Schmoe
 * @Author   jschmoe@cse.unl.edu
 *
 @Version    1.3, 08/12/00
 * @See      SomethingClass
 * @See      AnotherClass
 */
```

If you are using a software configuration management tool, you only need to include the description, since the tool will take care of the rest for you. The following subsections provide more specific instructions about each component of the prologue.

4.2.2.1 Description

The description must include:

- intent: why the code was developed and how it fits into the process, subsystem; and

- the modification history.

Optionally, you may include

- dependencies; and

- explanation of collaboration with other classes.

Don't include details that are better left as method or function descriptions, or as block comments within the code itself. The description should be as short or as long as is necessary, although one to three paragraphs should cover the majority of files.

For header files and include files, the description should focus on how the class or functions should be used. For source files, the description should focus on how the class or functions are implemented - algorithms, design patterns, etc. For Java files, the description should be a balance of both.

4.2.2.3 Author

List the authors in chronological order, original author first. You may use multiple author tags for multiple authors.

4.2.2.2 Version

The ideal situation is that the configuration management system automatically updates the version and date in the prologue. Otherwise pay attention to the synchronization between the version number in prologue and actual version number in configuration management system. If you are not using a configuration management system, use version numbers as specified by the instructor. If there are multiple versions, list only the current version using the *Javadoc* tag. You may list the other versions in the description.

4.2.2.3 See

List related books, links, classes, files, etc.

4.2.3 Code Comments

There is a blurry line between cluttering up your code, and putting meaningful comments in. Try to comment in such a way that keeps the code as clean as possible. Not adding comments is definitely not a good answer. This makes it much harder to figure out what you intended. When in doubt, comment the code.

In general, brief comments regarding individual statements may appear at the end of the same line, and should be vertically aligned with other comments in the vicinity for readability.

4.2.4 Classes, Methods/Functions, Interfaces, Class Attributes

Use the *Javadoc* format to comment classes, methods/functions, interfaces, and class attributes in C, C++, and Java.

Block comments should be put in the header files in C/C++ and (obviously) the Java source file. A block comment should be used to describe each method/function, and must be placed before the definition (or implementation) of the method/function. The block comment should include the first three of the below fields, and may contain the last two, particularly if they were written and/or modified by someone other than the original author.

- description: a brief description about the method or function.
- param: the passing parameters and their brief description
- return value
- author
- version

Here is a simple example:

```
/**
 * Compute the sum of two integers
*
 * @param param1, The first number
 * @param param2, The second number
 * @return  the sum of param1 and param2
 * @author  Some guy
 */
```

This sections details the requirements for spacing around operators (section 4.3.1), indentation and braces (section 4.3.2), and blank lines (section 4.3.3).

4.3.1 Spacing Around Operators

One space should be used around all operators with the following exceptions: the `::` and `->` operators in C++; and the `.`, `++`, `--` and `!` operators in C++ and Java. For example,

```
if  (value == 0)            // correct
{
    doSomething();
}
if  ( value==0 )            // wrong, no spaces around ==
{
    doSomething();
}
                            // wrong, spaces around
if  ( value == 0 )          parentheses
{
    doSomething();
}
```

4.3.2 Indentation and Braces

The contents of all code blocks should be indented to improve readability. Four spaces are recommended as the standard indentation. Place the beginning brace on the line below the method, loop, etc., and line the ending brace vertically with the method, loop, class, etc. to which it belongs. For nested if-then-else statements, place the else below the ending brace on the previous if. Use similar format for try-catch-finally blocks. Here is an example.

```
int main()
{
    doSomething();
    switch ( value )
    {
        case 1:
            while ( value == 0 )
            {
                doSomething();
            }
            break;
        case 2:
        case 3:
            doSomething();
            break;
        default:
            break;
    }
}

if ( value == 0 )
```

```
{
    doSomething();
}
else


{
    doSomething3();
}

try
{
    statement;
}
catch ( ExceptionClass e )
{
    statement;
}
finally
{
    statement;
}
```

The following three examples show the most common alternatives to the style we have chosen. Although there is nothing wrong with them, use the one described above for consistency.

```
if (value == 1) {
    doSomething();
}

if (value == 1)
    {
    doSomething();
    }

if (value == 1) {
    doSomething();
    }
```

4.3.3 Blank Lines

If it makes the code more readable, use a single blank line to separate logical groups of code. Two blank lines to separate each function or method definition may also make it easy to tell where each new function begins. You may also use dashed lines between functions and methods. For example:

```
//----------------------------------------
--------- /**
 * Javadoc style comments
 */
void doNothing()
{
```

```
//-----------------------------------------
--------- /**
 * Javadoc style comments
 */
void returnOne()
{
    return 1;
}


//-------------------------------------------------
```

4.4 Statements

This section describes some standards you must follow related to control structures (section 4.4.1), conditional statements (section 4.4.2), and include and package import statements (section 4.4.3).

4.4.1 Control Statements

In general, all control statements must be followed by an indented code block enclosed within braces, even if they only contain one statement. This allows the block to be easily expanded in the future. For example:

```
if (value == 0)
{               // Correct
    doSomething();
}

if (value == 0) doSomething();     // not recommended – no block, not indented

if (value == 0)
    doSomething();              // not recommended - no block
```

4.4.2 Conditional Statements in C/C++

In C and C++, conditional statements do not have to explicitly evaluate to TRUE or FALSE. Any expression that evaluates to zero is considered FALSE, and everything else is TRUE. For clarity, it is recommended that you write your conditional statements so they always evaluate to a Boolean value. Also, do not abuse notation by, for instance, comparing non-pointer values to *null*. Here are a few examples:

```
bool boolValue = getValue();

if ( !boolValue )                  // Correct
{
    doSomethingElse();
}

int intValue = getValue();

if ( intValue == 0 )               // Correct
{
```

```
    doSomething();
}

if ( intValue == null )             // Not recommended – null used for pointers
{
    doSomethingElse();
}
```

```
if ( !intValue )
{

// Not recommended - not explicit test
    doSomethingElse();


}
```

4.4.3 Include Statements and Package Imports

Includes and package imports should be grouped together at the top of each file, after the prolog. They should be logically grouped together, according to system includes, application includes, related packages, etc., with the groups separated by a blank line. Absolute path names should never be explicitly used in `#include` or `import` statements, since this is inherently non-portable.

For C/C++, system includes should use the `<file.h>` notation, and all other includes should use the `"file.h"` notation. For example:

(C/C++)

```
                                  //
#include <ltstdlib.h>             Correct
#include <ltstdio.h>              //
#include <ltXm/Xm.h>              //
#include "meaningfulname.h"       //
#include                          // wrong
"/proj/util/MeaningfulName.h"     -         absolute path given
                                  // wrong
#include <ltstdlib.h>             -         out of order
                                  // wrong path given for system
#include </usr/include/stdio.h>   -         file
```

(Java)

```
import java.awt.*;                // Correct
import java.awt.event.*;
import java.awt.event.KeyEvent.*;

import javax.swing.*;

import aLibrary.*;
```

4.5 Declarations

In this section the standards relating to declarations are described. Section 4.5.1 discusses variables and attributes in C, C++, and Java. Sections 4.5.2-4.5.6 describe rules that relate to C and/or C++, but not Java.

4.5.1 Variable and Attribute Declarations

Readability and understandability are the goals. Each variable/attribute should be individually declared on a separate line. Variables/attributes can be grouped by permission (public, private, protected and package) or by type (int, float, Applet, etc.), with groups separated by a blank line. Use whichever makes the code clear. The names should be aligned vertically for readability. There is no required ordering of types. A brief comment describing what the variable/attribute is for should be included. Here is one Example:

```
                    // The area of the     in square
int    area;        object,                inches
                    // The width of
int    width;       the              object, in inches
                       The height of the object, in
int    height;      // inches
doubl                  An             of
e      pi;          // approximation the   constant 'pi'


double e;           // An approximation of the constant 'e'
```

4.5.2 External Variable Declaration in C/C++

All external variables should be placed in header files. The actual allocation should take place in the implementation file (.c / .cpp). In general, the use of global variables is discouraged (consider creating a singleton class).

4.5.3 Enumerated Type Declaration in C/C++

The `enum` type name and enumerated constants should each reside on a separate line. Constants and comments should be aligned vertically. In general, the `enum` should be within a class. If the user of your class needs direct access to it, then put it in the public section and the user will then simply have to scope it with your class name. This will help reduce the pollution of the global namespace.

Consider using explicit values if these values might be saved to permanent store. If they are not explicit, then they will change if someone inserts a new value. Then, when the values are restored, they may not match the newer enum. In these cases, explicit assignment may help keep the sanity of data on permanent store. Here is an example.

```
enum CompassPoints { // Enums used to specify direction.
    North = 0,     // explicit values not necessary, but recommended.
    South = 1,
    East = 2,
    West = 3
};
```

4.5.4 Class Declarations in C++

All class definitions must include a constructor (either default, or at least one parameterized one), (virtual) destructor, copy constructor, and assign (=) operator. If any of these four are not currently needed, create stub versions and place them in the private section so they will not be automatically generated, then accidentally used. (This protects from core dumps and other errors.) It is advisable to put the public section first since the class should represent a concept and the public section holds the services of that concept. The user should not have to know the implementation details. It is suggested that `friend` declarations appear before the public section. All member variables should be either protected or private. It is recommended that definitions of inline functions follow the class declaration (in the .h file), although trivial inline functions (e.g., `{}` or `{ return x; }`) may be defined within the declaration itself.

4.5.6 Function Declaration in C/C++

All functions must be prototyped, with the prototypes residing in header files. If this is not done (especially in C) the parameters may not get checked at compile time, opening the door wide for run-time errors. In general, each class (or struct) will have its own header file (.h) and implementation file (.cpp).

All parameters should either have a meaningful name (even in the prototype) or the use of the parameter must be very clear by the type name. For example, "int" does NOT make the use clear, so you must have something like "int count" that makes the use much more clear.

5 Java Documentation Comments

Java programs can have two kinds of comments: implementation comments and documentation comments. Implementation comments are those delimited by /*… */, and //. Documentation comments (also known as " doc comments") are Java-only, and delimited by /**… */. Javadoc comments can be extracted to HTML files using the Javadoc tool, creating a very nice overview, tree structure and description of all the Java files. In fact, you can use HTML in your comments.

As mentioned previously, Javadoc comments are legal in C/C++. In fact, there are several applications available to process them in much the same way as Javadoc does for Java.

Javadoc comments describe Java classes, interfaces, constructors, methods, and fields. Each Javadoc comment is set inside the comment delimiters /**… */, with one comment per class, interface, or member. The general format of Javadoc comments are list as below:

```
(line 1)  /**
(line 2)   *
           * Beginning of
(line 3)   description
...
(line x)   * end of description
(line
x+1)       *
(line      *
x+2)       @Tag   tagValue
(line      *      tagValue
```

```
x+3)       @Tag
...
(line      *
x+y)       @Tag   tagValue
(line x+y+1)*/
```

5.1 Description

Provide a detailed description of the class, interface, method, or field. This field may include the following items.

- intent
- preconditions and/or postconditions
- dependencies
- collaboration with other class, or method
- modification history

You should always give a brief description of the class, interface, method, or field. Unless there are no explicit preconditions to a method, always specify the preconditions. Likewise, postconditions should be included unless they are absolutely obvious. The other items should be used when appropriate, and you can add other items to increase the understandability of the source code.

5.2 Tags

Below are all of the Javadoc tags, listed in the order they should appear.

```
author     (classes and interfaces only, required)
version    (classes and interfaces only, required)

           (method
param      s       and constructors only)
           (method
return     s       only)
           (@throw is a synonym added in
exception  s       Javadoc 1.2)
see
since
serial     (or @serialField or @serialData)
deprecate  (see
d          How     and When To Deprecate APIs)
```

Please reference Javadoc homepage http://java.sun.com/j2se/Javadoc/index.html for more detailed descriptions.

See Chapter 6 for an extensive example of Javadoc comments

6 Examples

The next two sections give a complete example of code that follows the standard given in this document. For Java, a single class is given as an example. For C++, there are two examples. The first is a header/source file for a class, and the second a header/source file for a group of related functions.

6.1 Java

The following is an example Java class called **BinaryTree**.

```
//-----------------------------------------------------------------
----------------
// BinaryTree.java
//
//   Written By  : Chuck Cusack
//   Written For : The fun of it
// Date   : June 12, 2001
//   Version: 1.0
//
//   Modified on : July 12, 2002
//   Version: 1.3
//-----------------------------------------------------------------
----------------
import blah.foo.TreeNode;
import blah.foo.Connectives;
//-------------------------------------------------
--------------------------- /**
 *BinaryTree is a class that implements the binary tree
  <i>ADT</i>.
 *The Binary tree is constructed using objects of type TreeNode.
 *Since TreeNodes have char key values, the BinaryTree stores
  nodes which
 *have char key values.
 *Most of the standard binary tree operations are implemented.
 *<p><font color=red>
 *Keep in mind that this is only serving as an example of the
  coding standard,
 *and is not meant to be an actual useful class.  You will notice
  that some
 *methods are not present, and some attributes are not used.  Do
  not worry
 *about this.  Use this as a guide for the standard, not of
  proper object
 *oriented design, etc. </font>
 *
```

```
 *@author Chuck Cusack
 *@version 1.3
 *@see blah.foo.TreeNode
 *@see
 blah.foo.Connecti
 ves */
public class BinaryTree
{
   //-------------------------------------------------------------
   -------------
   //The Class Attribute(s)
//


   /**
    *A reference to the root of
    the binary tree. */
   private TreeNode root;

   /**
    E.     Whether or not the tree is
    a binary search tree. */
   private boolean isBinarySearchTree;

   /**
    *The number of nodes
    currently in the tree. */
   public int numberOfNodes;


   //-----------------------------------------------
   -------------------------- /**
    *The default constructor.  It simply creates an empty tree.
*
    */
   public BinaryTree()
   {
      root=null;     // Since the tree is empty, the root should
      point to null.
   }

   //-----------------------------------------------
   -------------------------- /**
    *A single-node constructor, which creates a new binary tree
     with one node.
*
    *@param  rootNode The node which will become the new root.
    *
```

```
    */
   public BinaryTree(TreeNode rootNode)
   {
      root=rootNode; // Clearly the only thing that needs to be
      done.
   }

   //-------------------------------------------------
   -------------------------- /**
    * A three-node constructor, which creates a new binary tree
      with the first
    * argument as the root, the second argument as the root©s left
      child, and
    * the third argument as the root©s right child. If the
      leftNode or
    * rightNode have children, they will be preserved.  If
      rootNode has any
    * children, they will be lost.
    *
    * @param  rootNode The node which will become the new root.
    * @param  leftNode The node which will become the root©s left
      child.
    * @param  rightNode The node which will become the root©s
      right child.
*
    */
   public BinaryTree(TreeNode rootNode, TreeNode leftNode,
   TreeNode rightNode)
   {
      root = rootNode;
      root.setLeft(leftNode);
      root.setRight(rightNode);
   }

   //-------------------------------------------------
   -------------------------- /**
     * @return A reference to the
     root of the tree. */
   public TreeNode getRoot()
   {
      return root;
   }

   //-------------------------------------------------
   -------------------------- /**
     * Replaces the root with the given node.  This orphans the
      old tree, and
```

```
  * maintains any children newRoot might have.

  *
  * @param newRoot The node that will become
  the new root of the tree. */
public void setRoot(TreeNode newRoot)
{
   root = newRoot;
}

//-------------------------------------------------
-------------------------- /**
  * Performs an in-order traversal of the binary tree.
  * <p>
  * Precondition: startingNode is a node in the BinaryTree,
  traversalString
  *   references a valid StringBuffer.
  * <p>
  * Postcondition: traversalString has been appended with the
  in-order
  *   traversal of the tree rooted at startingNode, including
     parentheses
  *   if addParentheses was set to true.
  * <p>
  *
  * @param startingNode The node to start the traversal at.
  * @param traversalString Stores the string that is created as
  the tree is
  *     parsed.
  * @param AddParentheses Whether or not to include parentheses
  in the
  *     appropriate places during a traversal.
  *
  */
private void InOrderTraversal(TreeNode startingNode,
                    StringBuffer traversalString,
                    boolean AddParentheses)
{
   if(startingNode != null)
   {
      //     Check for the base case
      //Add a beginning parenthese if
      appropriate if(AddParentheses ==
      true && !startingNode.isLeaf())
      {
         traversalString.append(©(©);
```

```
      }

      //Process the left subtree
      InOrderTraversal(startingNode.getLeft
      (), traversalString,
                 AddParentheses);

      //     Add the current node©s key to the string
      traversalString.append(startingNode.getKey());

      //Process the right subtree
      InOrderTraversal(startingNode.getRight
      (), traversalString,
                 AddParentheses);

      //Add an ending parenthese if
      appropriate if(AddParentheses ==
      true && !startingNode.isLeaf())
      {
         traversalString.append(©)©);
      }
   }
}

//-----------------------------------------------
--------------------------- /**
 * Splice the first node into the tree as the parent of
 * the second node, making it the right child, and setting
 * the left child to null.
 *
 * @param currentNode the node in the tree.
 * @param newNode the node add.
 *
 */
public void spliceInRight(TreeNode currentNode, TreeNode
newNode)
{


   //   Special case: The root
   is being replaced
   if(currentNode == root)
   {
      root=newNode;
   }
   else
   {
```

```
            //Normal case: An internal node is
            being spliced in TreeNode
            parentNode =
            currentNode.getParent();

            //     Determine whether newNode should be the left or
              right child of
            //its new parent. Then, set the
            appropriate references.
            if(currentNode ==
            parentNode.getLeft())
            {
               parentNode.setLeft(newNode);
            }
            else
            {
               parentNode.setRight(newNode);
            }
         }

         //Make the currentNode the right child
         //   and null the left child
         of the new node
         newNode.setRight(currentNode
         ); newNode.setLeft(null);
      }
      //-----------------------------------------------------------
      --------------
}
```

6.2 C++

The following is an example C++ class called **Queue**.

6.2.1 queue.h

```
#if !defined(QUEUE_000001)
#define QUEUE_000001

/*
 *queue.h
*
 *Written by : Nick Steinbaugh
 *Written for: JDE310
 * Date   : September 3, 2003
 IV. Version
 : 1.0 */

/**
 *A class for storing data, specifically integers, in a queue.
```

```
*
 *Version 1.0
 *
 * @AuthorNick Steinbaugh solarflare@gmx.net
 * @Version    1.0, 09/03/2003
 */

#define MAX_SIZE 100

class Queue
{
private:
   //Pointer to the head of the queue
   int head;

   //Pointer to the tail of the queue
   int tail;

   //Array to store the queue data
   int data[MAX_SIZE];

   //Boolean to keep track of whether or
   not the queue is empty bool empty;
public:

   /**
    *Default
    constructor */
   Queue();

   /**
    *Add an integer to the back of the queue
*
    * @param    item, The integer to add to the queue
    *@return Whether or not adding the
    item was successful */
   bool enqueue(int item);

   /**
    *Remove and return the first item in the queue
*
    * @param    item, The integer to store the value in
    *@return Whether or not removing and storing
    the item was successful */
   bool dequeue(int &item);

   /**
    *Get the pointer to the first array element of the queue
*
```

```
    * @return The pointer to the location in the
    array of the first item */
   int *getHead();

   /**
    * Check to see if the queue is empty
*
    * @return Whether or not the
    queue is empty */
   bool isEmpty() const;

   /**
    * Check to see if the queue is full
*
    * @return Whether or not the
    queue is full */
   bool isFull() const;

   /**
    * Get the size of the queue
*
    * @return The current number of
    items in the queue */
   int size() const;
};

#endif
```

6.2.1 queue.cpp

```
/*
 * queue.cpp
*
 * Written by : Nick Steinbaugh
 * Written for: JDE310
 *  Date   : September 3, 2003
 * Version :
 1.0 */

/**
 * A class for storing data, specifically integers, in a queue.
*
 * Version 1.0



 *
 * @AuthorNick Steinbaugh solarflare@gmx.net
 * @Version    1.0, 09/03/2003
 */
```

```
#include "queue.h"

Queue::Queue()
{
   //Set the head and tail pointers to zero and empty to true
   head = 0;
   tail = 0;
   empty = true;
}

bool Queue::enqueue(int item)
{
   //Check to see if the queue is full
   if(isFull())
      return false;

   //Adjust tail pointer
   data[tail] = item;
   tail++;
   if(tail >= MAX_SIZE)
      tail -= MAX_SIZE;
   empty = false;
   return true;
}

bool Queue::dequeue(int &item)
{
   //Check to see if the queue is empty
   if(isEmpty())
      return false;

   //Copy queue data to item
   item = data[head];

   //Adjust the head pointer
   head++;
   if(head >= MAX_SIZE)
      head -= MAX_SIZE;
   if(head == tail)
      empty = true;
   return true;
}

int *Queue::getHead()
{
   //Get the pointer to the head of the queue
   return data + head;
}
```

```
bool Queue::isEmpty() const
{
   return empty;
}

bool Queue::isFull() const
{
   return !empty && head == tail;
}

int Queue::size() const
{
   if(tail >= head)
      return tail - head;
   else
      return tail + MAX_SIZE - head;
}
```

SOFTWARE TESTING LAB:

Program: 1

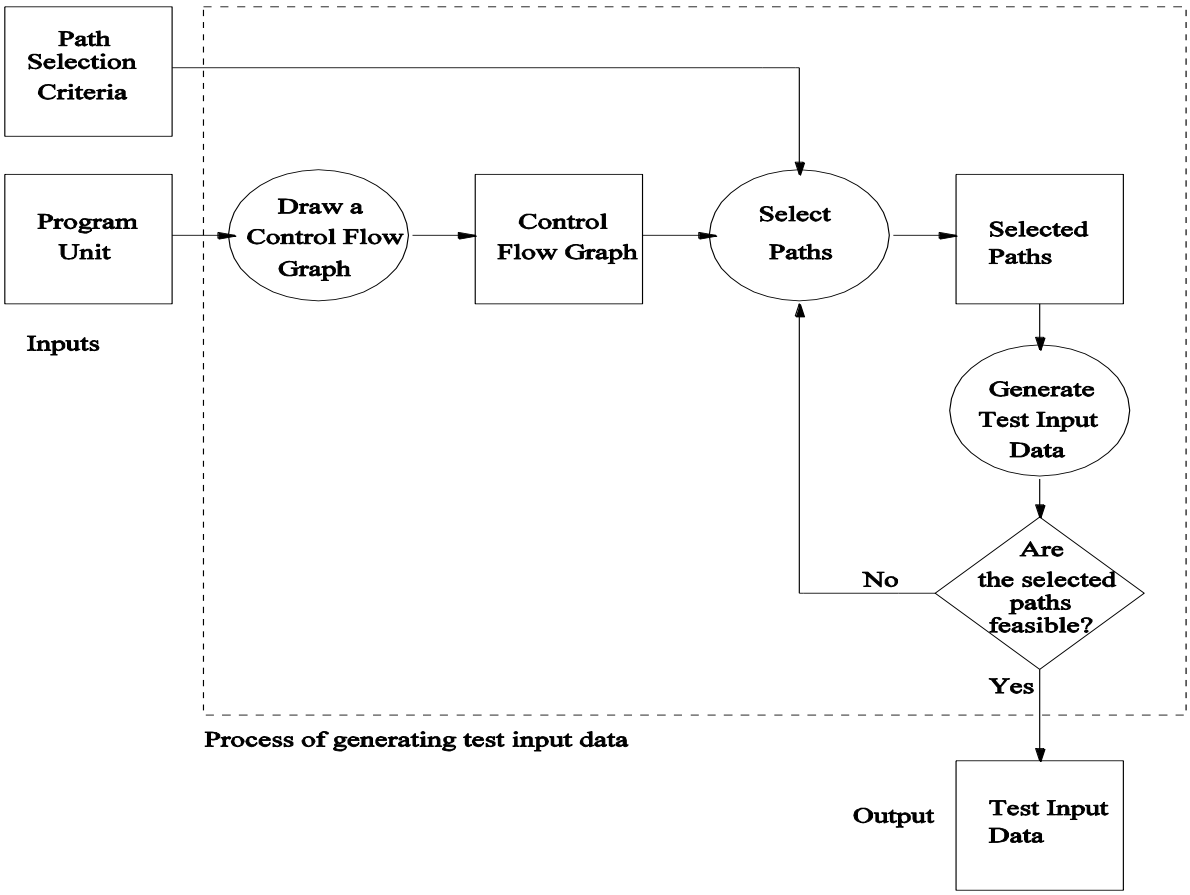
AIM : Program in "C" Language to demonstrate the working of the following constructs:
i)do-while ii) while-do iii) if-else iv) switch v) for

THEORY:

- Two kinds of basic program statements:
  - Assignment statements (Ex. x = 2*y;)
  - Conditional statements (Ex. If(), for(), while(), …)
- Control flow
  - Successive execution of progra–m statements is viewed as flow of control.
  - Conditional statements alter the default flow.
- Program path
  - A program path is a sequence of statements from entry to exit.
  - There can be a large number of paths in a program.
  - There is an (input, expected output) pair for each path.
  - Executing a path requires invoking the program unit with the right test input.
  - Paths are chosen by using the concepts of path selection criteria.
- Tools: Automatically generate test inputs from program paths.

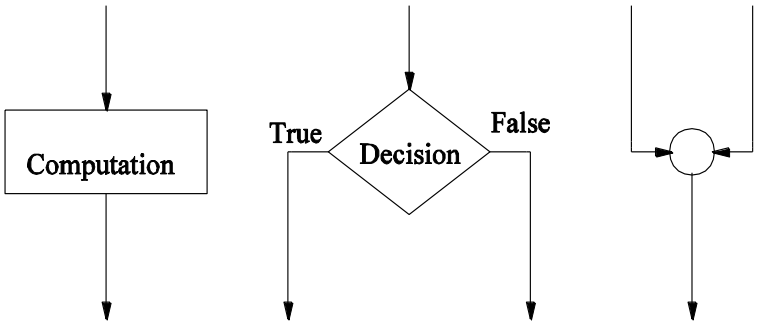
Outline of Control Flow Testing

- Inputs to the test generation process
  - Source code
  - Path selection criteria: statement, branch, …
- Generation of control flow graph (CFG)
  - A CFG is a graphical representation of a program unit.
  - Compilers are modified to produce CFGs. (You can draw one by hand.)
- Selection of paths
  - Enough entry/exit paths are selected to satisfy path selection criteria.
- Generation of test input data
  - Two kinds of paths
    - Executable path: There exists input so that the path is executed.
    - Infeasible path: There is no input to execute the path.
  - Solve the path conditions to produce test input for each path.



Outline of Control Flow Testing

- Symbols in a CFG



Path Selection Criteria:

- Program paths are selectively executed.
- Question: What paths do I select for testing?

- The concept of path selection criteria is used to answer the question.
- Advantages of selecting paths based on defined criteria:
  - Ensure that all program constructs are executed at least once.
  - Repeated selection of the same path is avoided.

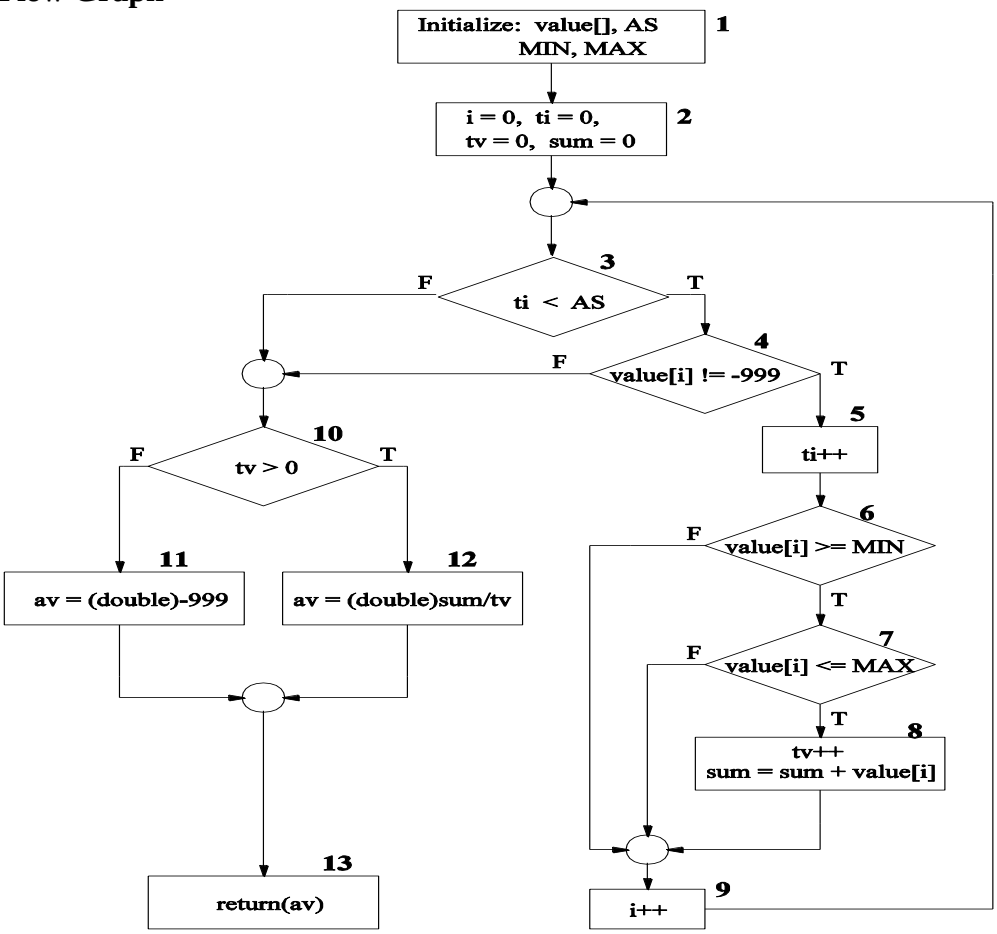
- One can easily identify what features have been tested and what not.

- **Path selection criteria**
  - Select all paths.
  - Select paths to achieve complete statement coverage.
  - Select paths to achieve complete branch coverage.
  - Select paths to achieve predicate coverage
- **Statement coverage criterion:**
  - Statement coverage means executing individual program statements and observing the output.
  - 100% statement coverage means all the statements have been executed at least once.
    - Cover all assignment statements.
    - Cover all conditional statements.
  - Less than 100% statement coverage is unacceptable.
- **Branch coverage criterion**
  - A branch is an outgoing branch (edge) from a node in a CFG.
    - A condition node has two outgoing branches – corresponding to the True and False values of the condition.
  - Covering a branch means executing a path that contains the branch.
  - 100% branch coverage means selecting a set of paths such that each branch is included on some path.

Example: A function to compute the average of selected integers in an array.
code: ReturnAverage()

```
public static double ReturnAverage(int value[],  int AS, int MIN, int MAX){
int i, ti, tv, sum;
double av;
i = 0; ti = 0; tv = 0; sum = 0;
while (ti< AS && value[i] != -999) {
ti++;
if (value[i] >= MIN && value[i] <= MAX) {
tv++;
sum = sum + value[i];
}
i++;
}
if (tv> 0)
av = (double)sum/tv;
else
av = (double) -999;
return (av);
}
```

Control Flow Graph



Paths in a Control Flow Graph:

- A few paths in above flow graph.
  - Path 1: 1-2-3(F)-10(T)-12-13
  - Path 2: 1-2-3(F)-10(F)-11-13
  - Path 3: 1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
  - Path 4: 1-2-3(T)-4(T)-5-6-7(T)-8-9-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13

Expected output:
Statement Coverage:
Inputs:

SCPath1	1-2-3(F)-10(F)-11-13
SCPath2	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13

Output:
100% statement coverage is achived.

Branch Coverage:
Inputs:

BCPath 1	1-2-3(F)-10(F)-11-13
BCPath 2	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
BCPath 3	1-2-3(T)-4(F)-10(F)-11-13
BCPath 4	1-2-3(T)-4(T)-5-6(F)-9-3(F)-10(F)-11-13
BCPath 5	1-2-3(T)-4(T)-5-6(T)-7(F)-9-3(F)-10(F)-11-13

Output:
100% branch coverage is achieved

Path Coverage:
Inputs:

PCPath1	1-2-3(F)-10(F)-11-13
PCPath2	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
PCPath3	1-2-3(F)-10(F)-11-13
PCPath 4	1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10(T)-12-13
PCPath 5	1-2-3(T)-4(F)-10(F)-11-13
PCPath 6	1-2-3(T)-4(T)-5-6(F)-9-3(F)-10(F)-11-13
PCPath 7	1-2-3(T)-4(T)-5-6(T)-7(F)-9-3(F)-10(F)-11-13

Output:
100% Path Coverage has been achieved

Program No: 2

AIM: "A Program written in "C" language for matrix multiplication fails" Introspect the causes for its failure and write down the possible reasons for its failure

THEORY:

Validation Testing:

- It succeeds when the software functions in a manner that can be reasonably expected by the customer.

- Reasonable expectations are defined in the SRS, the specification contains a section called validation criteria, information contained in that section forms the basis for a validation testing.

1)Validation Test Criteria: After each validation test case has been conducted, one or two possible conditions exists.

1. The function or performance characteristic confirms to specification and is accepted
2. A deviation from specification is uncovered and a deficiency list is created.
3. Deviation or error discovered at this stage in a project can rarely be correct prior to scheduled delivery.

2)Configuration Review: The intent of review is to ensure that all elements of the s/w configuration have been properly developed, are cataloged. Sometimes called audit.

3)Alpha testing: is conducted at the developer's site by end-users.

- The s/w is used in a natural setting with the developer "looking over the shoulder" of typical users and recording errors and usage problems
- Alpha tests conducted in a controlled environment

4)Beta testing: is conducted at the end-user site, unlike alpha testing, developer is generally not present.

- Beta test is a "live" application of the software in an environment that can't be controlled by the developer.
- End-users records al the problems and reports to the developer at regular intervals
- As a result of problems reported during beta tests, s/w engineers make modifications and then prepare for release of the s/w product.

System Testing: Its primary purpose is to test the complete software.

1)Recovery Testing: Many computer –based systems must recover from faults and resume processing within a prespecified time.

- Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed
- If recovery automatic (performed by the system itself), reinitialization, checkpointing mechanisms, data recovery and restart are evaluated for correctness
- If recovery is human intervention, the mean-time- to – repair(MTTR) is evaluated to determine whether it is within acceptable limits.

2)SecurityTesting: Security testing verifies that protection mechanism built into a system infact protect it from improper penetration.

3)Stress Testing: Stress testing refers to a type of testing that is so harsh, it is expected to push the program to failure. For example, we might flood a web application with data, connections, and so on until it finally crashes. Buffer overflows are classic examples of stress test results.

4)Performance Testing: Performance testing is designed to test the run-time performance of software within the context of an integrated system.

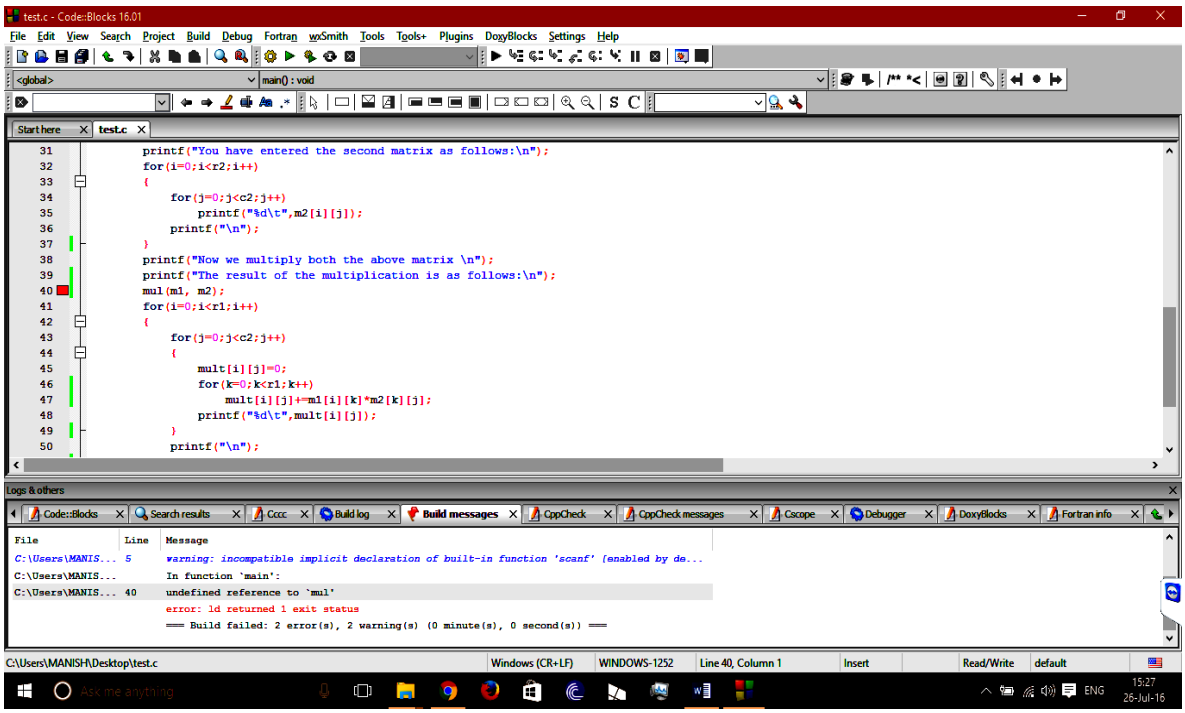
- It occurs throughout all steps in the testing process, even at unit level also.
- Performance tests are often coupled with stress testing.

Program: Write the Matrix multiplication Program
OutPut:

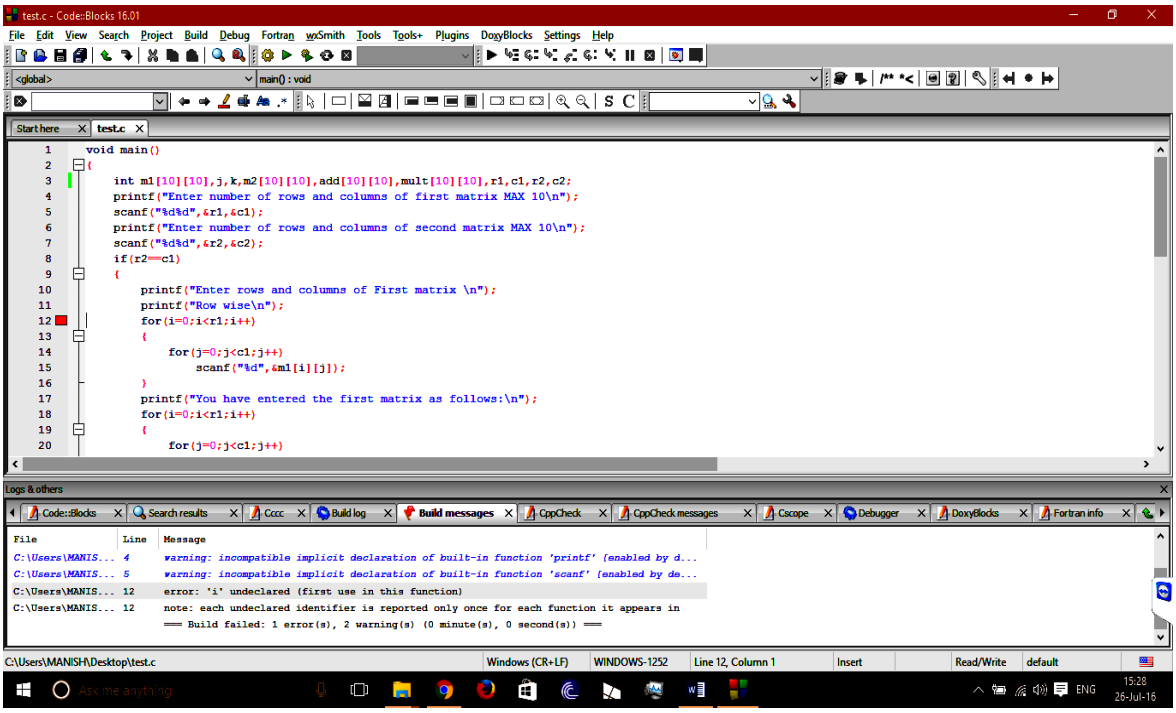
Causes for failure:

Possible reasons for its failure:

Screenshots:



Missing function in line 40



Using undefined variable i

Program No.: 3

AIM: "Take any system (e.g ATM system) and study its system specifications and report the various bugs

THEORY:

Bug Def: *Deviates from expected behavior is a bug.*

Consequences: *(how bugs may affect users)*

These range from mild to catastrophic on a 10 point scale.

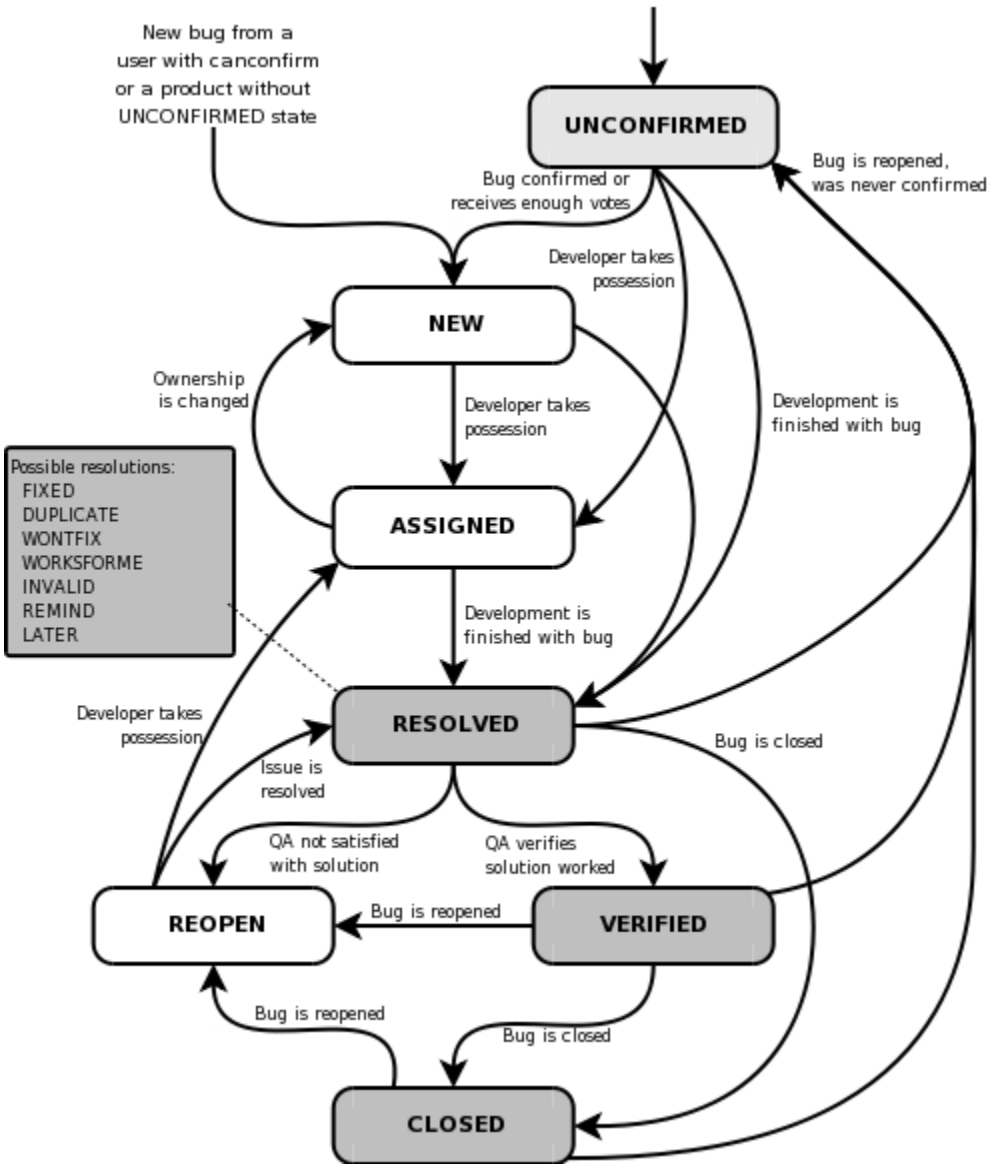
- ***Mild***
  - *Aesthetic bug such as misspelled output or mal-aligned print-out.*
- ***Moderate***
  - *Outputs are misleading or redundant impacting performance.*
- ***Annoying***
  - *Systems behavior is dehumanizing for e.g. names are truncated/modified arbitrarily, bills for $0.0 are sent.*
  - *Till the bugs are fixed operators must use unnatural command sequences to get proper response.*
- ***Disturbing***

    - *Legitimate transactions refused.*
    - *For e.g. ATM machine may malfunction with ATM card / credit card.*
- ***Serious***
    - *Losing track of transactions & transaction events. Hence accountability is lost.*
- ***Very serious***
    - *System does another transaction instead of requested e.g. Credit another account, convert withdrawals to deposits.*
- ***Extreme***
    - *Frequent &Arbitrary - not sporadic & unusual.*
- ***Intolerable***
    - *Long term unrecoverable corruption of the Data base. (not easily discovered and may lead to system down.)*
- ***Catastrophic***
    - *System fails and shuts down.*
- ***Infectious***
    - *Corrupts other systems, even when it may not fail.*

Tools example:Bugzilla/Issue-Tracker/PR-Tracker etc, Jira,QC(Quality Center)

Bug Life Cycle:

Lifecycle of a Bugzilla Bug:



Sample example for bug report:

Here is a scanariocausea a bug. Let's assume in your application under test you want to create a new user with user information, for that you need to log into the application and navigate to Users-> menu-> new user, then enter all the details in the user form like first name, last name, age, address, phone etc. once you enter all these information, you need to click on "save" button in order to save the user information. Now you can see a success message saying. "New User" has been created successfully

But when you entered all information and pressed save button, the application is crashed and you got one error page on screen (capture this error message window and save as a Microsoft paint file). Now this a bug, that you report to developer your bug tracking tool.

BUG FORMAT:

Bug Name: *Application crash on clicking the save button while creating a new user*

Bug id:(*it will be automatically created by the BUG tracking tool once you save this bug*

Area path*: USER Menu -> New Users*

Build Number: *Version number 5.0.1*

Severity*: High(High/medium/low) or 1*

Priority: *High(High/medium/low) or1*

Assigned to*: Developer X*

Reported by*: Your Name*

Reported on*: date*

Reason*: Defect*

Status*: New/open/active (depends on the tool you are using)*

Environment*: Window 2003/sql server 2005*

Description:

Application crashes on clicking the save button which creating a new user, hence unable to create a new user in the application.

Steps Reproduce*:*

1.Logon into the application

2.Navigate to the user menu-> new user

3. filled all the user information fields

4. clicked on "save" button

5. seen an error page "ORA1090 exception: insert values error…"

6. see the attaced logs for more information

7. and also see the attached screen shot of the error messages

Expected result*: on clicking SAVE button, you should be prompted to a success message "New user has been created successfully"*

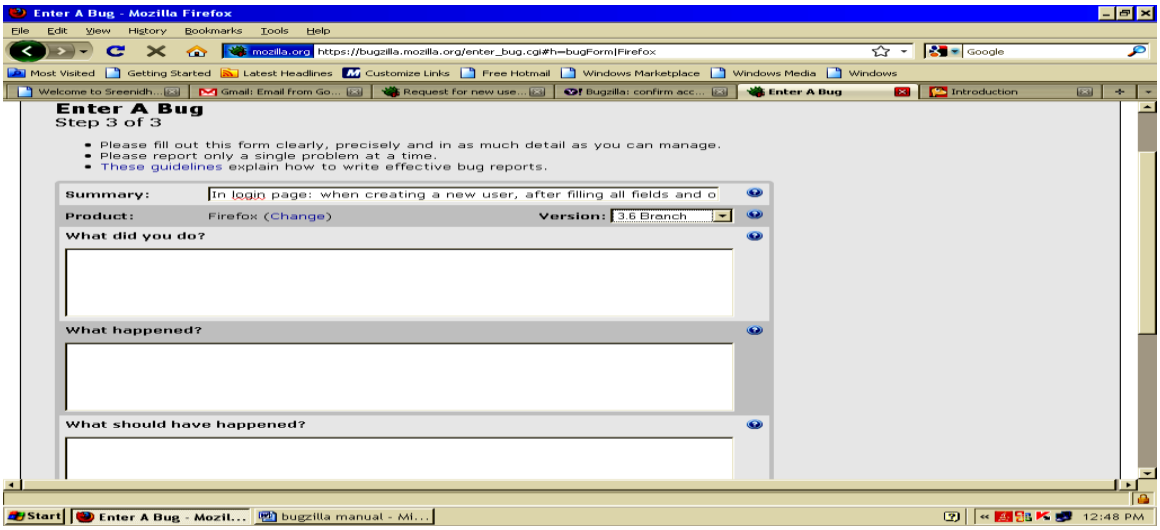
(Attach "application crash screen shots….)

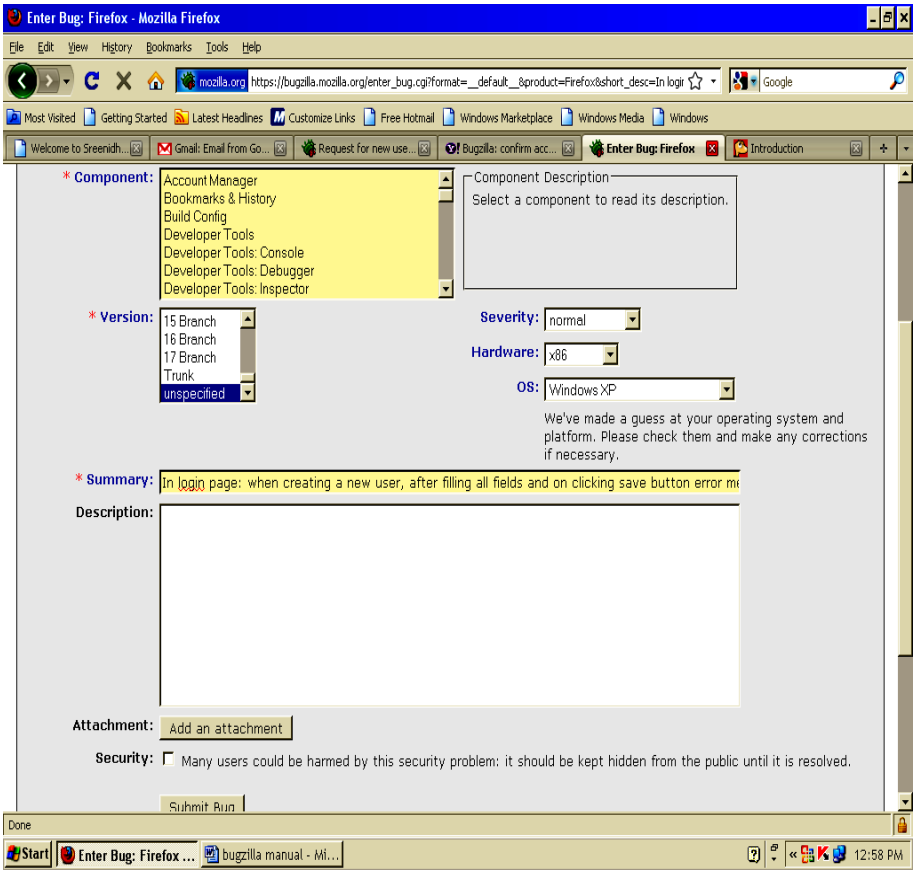
Save the defect/bug in the bug tracking tool. You will get a bug id, which you can use for further bug reference.

Default new bug mail will go to respective developer and the default module owner(team leader or manager) for further action.

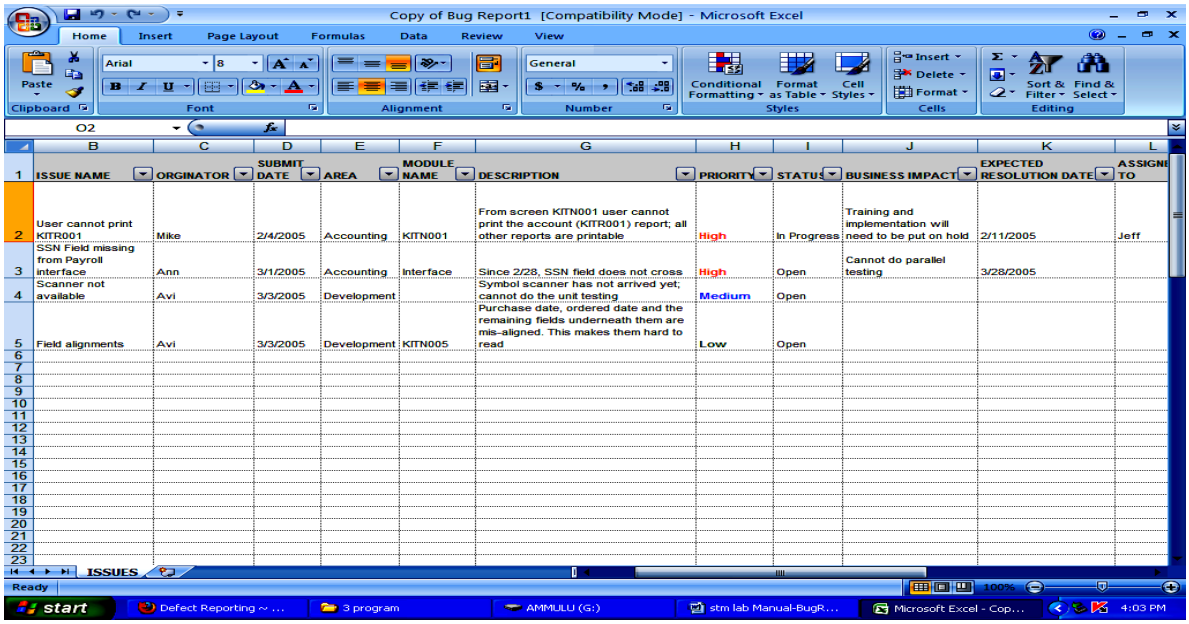
You can report bug details using tool as shown below

Ex: Bugzilla bug summary sheet





You can report bug details using Excel sheet as shown below



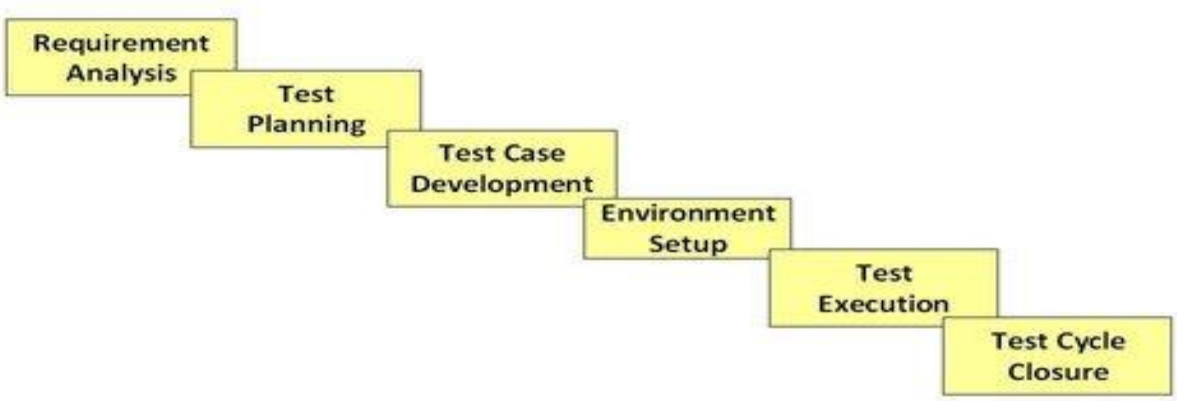
Program No.: 4

AIM: "Write a Test Cases for any known Application (Ex: Banking system, ATM)

THEORY:

Software Testing Life Cycle:

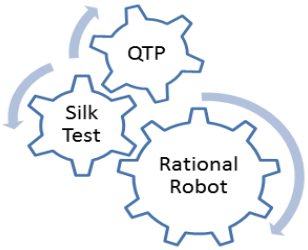
The different stages in Software Test Life Cycle:



Each of these stages have a definite Entry and Exit criteria , Activities & Deliverables associated with it.

Manual Testing: Testing when carried out without the help of any tool is called as Manual testing. When user is required to carry out every activity related to testing manually, we say it is

a Manual testing process.



Steps for manual testing: -
1. Understand the functionality of application
2. Prepare the test plan
3. Write the test cases and execute on the aut.
4. Verify actual and expected results
5. Prepare the bug report

After recieving the modified build from the developer we will go for retesting and regression testing. these process are repeated until the bug status is closed.

Examples:A simple example of testing would be a Login screen - we could test quite a few scenarios. Enlisted below are a few: -
1) Checking if Both Username and Password are entered and not blank
2) Masking of Password is implemented
3) Verifying if screen navigates to next page incase valid Login credentials are provided.
4) Incase of invalid login, ensuring system does not navigate further but displays an error message.
5) Checking the maximum possible length of login and password fields.

Automation Testing:

Using Automation tools to write and execute test cases is known as automation testing. No manual intervention is required while executing an automated test suite.

Testers write test scripts and test cases using the automation tool and then group into test suites.

Benefits of Automation Testing

- Reduction of repetitive work.
- Repeatability
- Greater consistency
- Ease of access of information about tests or testing

Advantages:

- You would have tested software applications or web applications manually, so you might be aware of the drawbacks of manual testing. Manual testing is time consuming, tedious and requires heavy investment in human resources.
- Time constraints often make it impossible to manually test every feature thoroughly before software application or web application is to be released. This leaves you wondering whether serious defects have been detected or not.
- To address all these issues automation testing is done, you can create tests that check all aspects of the software applications and then execute these test cases every time any changes are made in software application.

Benefits of Automation Testing

- **Fast:** Runs tests significantly faster than human users.
- **Repeatable:** Testers can test how the website or software reacts after repeated execution of the same operation.
- **Reusable:** Tests can be re-used on different versions of the software.
- **Reliable:** Tests perform precisely the same operation each time they are run thereby eliminating human error.
- **Comprehensive:** Testers can build test suites of tests that covers every feature in software software application.
- **Programmable:** Testers can program sophisticated tests that bring hidden information.

Automation tools:

Following are the most popular test tools:

- **HP UFT:**HP Unified Functional Testing(now known as HP Functional Test) **is the market leader in Functional Testing Tool**. The tool supports plethora of environments including SAP,Java, Delphi amongst others. **HP UFT can be used in conjunction with Quality Center** which is a comprehensive Test Management Tool. know is light tool which can be recommended for web or client/server applications.
- **Rational Robot:**It's is an IBM tool used to automate regression, functional and configuration tests for client server, e-commerce as well as ERP applications. It **can be used with Rational Test Manage**r which aided in Test Management Activities
- **Selenium**:Its an **open source Web Automation Tool**. It supports all types of web browsers. Despite being open source its actively developed and supported

Automation testing Process:

Simply record and play back

1. Learning
2. Recording
3. Edit script
4. Run
5. Analyze Results
6.

Manual Testing Vs Automation Testing

Manual Testing	Automation Testing
1. Time consuming and tedious: Since test cases are executed by human resources so it is very slow and tedious.	**1. Fast** Automation runs test cases significantly faster than human resources.
2. Huge investment in human resources: As test cases need to be executed manually so more testers are required in manual testing.	**2. Less investment in human resources:** Test cases are executed by using automation tool so less tester is required in automation testing.
3. Less reliable: Manual testing is less reliable as tests may not be performed	**3. More reliable:** Automation tests perform precisely same operation each time

with precision each time	
4. Non-programmable: No programming can be done to write sophisticated tests which fetch hidden information.	**4. Programmable:** Testers can program sophisticated tests to bring out hidden information.

Test case definition: A test case is step by step instructions to test a specific requirement. Step by step driving direction to check specific functionality.

- one test case can contain one or more steps based on the complexity of requirement.
- Simple and clear steps or driving directions to test the s/w functionality.

Test Scenario: *A document specifying a sequence of actions for the execution of a test*
Test Case: *One or more input values, execution preconditions, steps for execution, expected results and execution post-conditions, developed for a particular objective or test condition.*
Test Data: *Data that exists before a test is executed, and that affects or is affected by the component or system under test.*

How to create Effective test case:

1. Create test case based on requirements:
   - Business requirements are the input for test case. Define test conditions based on detailed requirements
   - Develop multiple cases to test individual business requirements when necessary.
2. Validate Test Case:
   - Validate the test cases are defined correctly to test requirements and that cases provide adequate coverage of requirements
   - Review with business Functional expert
3. Target high risk functions:
   - It is not always possible to test everything. cases should target high risk functions first for more extensive testing.
4. Maintain Treaceability:
   - Map test cases to requirements, components and scripts
   - This ensure that the test documents are mainatainable
   - Traceabilllity is very important when the requirements or test conditions change and it becomes necessary to update the test scripts
5. Repeability:
   - Manual scripts need to be detailed enough for everyone to follow cases
6. Manual and Automated:
   - Test cases may be manual or automated

IEEE FORMAT FOR TEST CASE:

1. **Test Case Id:** Unique Name or ID.
2. **Test Case Name:** Name of the test condition
3. **Features to Be Tested:** Corresponding Module/ Function/ Service
4. **Test Suite Id:** In which batch, is this case is a member of
5. **Priority:** Importance of the test case in terms of functionality

P0: Basic Functionality
P1: General Functionality (Input Domain, Error Handling, Compatibility)
P2: Cosmetic Testing (User Interface)

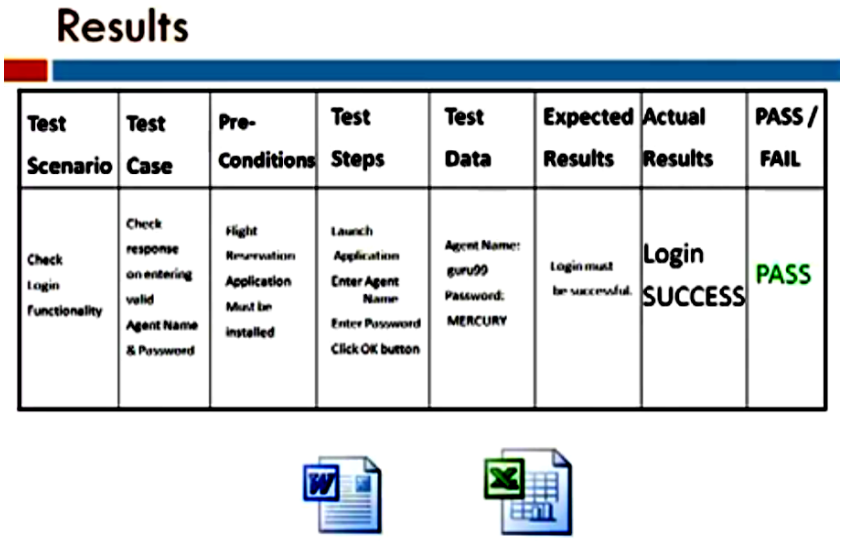
6. **Test Environment:** Required Hardware and Software to execute this test case
7. **Test Effort (Person/Hour):** Time required to execute this test case (E.g. 20 minutes on an average)
8. **Test Duration:** Date and Time (Scheduling)
9. **Test Setup:** Necessary tasks to do before starting this test case execution
10. **Test Procedure:** A step-by-step process to execute these test cases from base state to end state

S. No.	Action	Input Required	Expected	Actuals	Result	Comments

Filled during test design stage filled during test execution

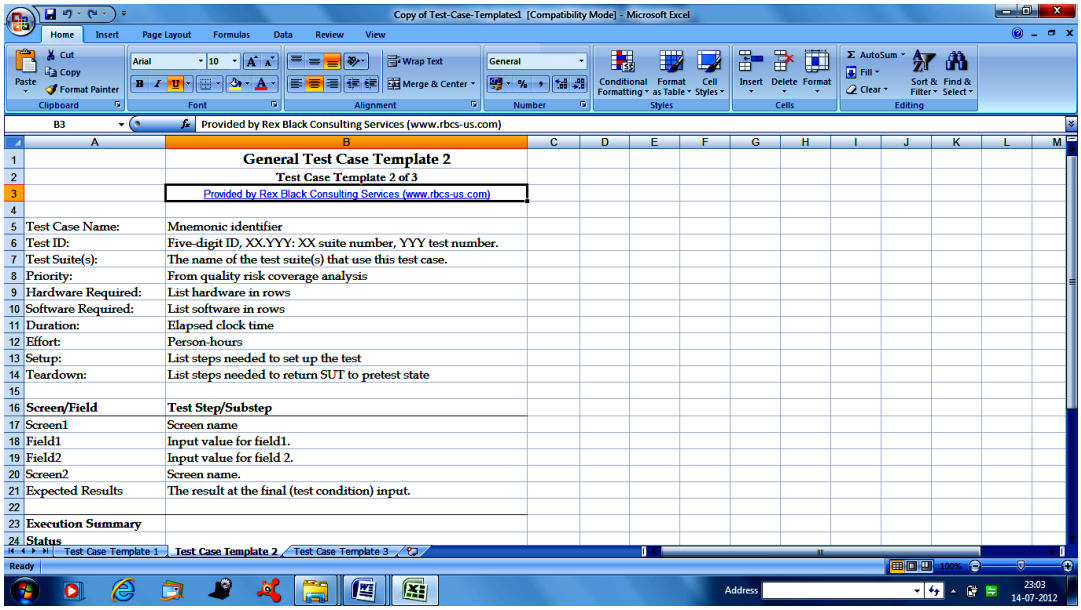
11. **Test Case Pass/Fail Criteria:** when this test case is pass and when this test case fails

Test Case General Template:



Test Case Document:

Test Case Document (Banking application)							
Project Name < >	**Test case version no.** < >	**Module Name** < >	**Executed by** < >	**Prepared by** < >	**Executed on** < >	**Prepared on** < >	**Reviwed on** < >
Test case ID	***Requirements ID***	***Test case description***	***Test procedure***	***Test data***	***Expected result***	***Actual result***	***Status***



	A	B
1		**General Test Case Template 2**
2		**Test Case Template 2 of 3**
3		Provided by Rex Black Consulting Services (www.rbcs-us.com)
4		
5	Test Case Name:	Mnemonic identifier
6	Test ID:	Five-digit ID, XX.YYY: XX suite number, YYY test number.
7	Test Suite(s):	The name of the test suite(s) that use this test case.
8	Priority:	From quality risk coverage analysis
9	Hardware Required:	List hardware in rows
10	Software Required:	List software in rows
11	Duration:	Elapsed clock time
12	Effort:	Person-hours
13	Setup:	List steps needed to set up the test
14	Teardown:	List steps needed to return SUT to pretest state
15		
16	**Screen/Field**	**Test Step/Substep**
17	Screen1	Screen name
18	Field1	Input value for field1.
19	Field2	Input value for field 2.
20	Screen2	Screen name.
21	Expected Results	The result at the final (test condition) input.
22		
23	**Execution Summary**	
24	**Status**	

Test Cases in Excel Sheet

Test Case for ATM

TC 1 :- succesful card insertion.
TC 2 :- unsuccessful operation due to wrong angle card insertion.
TC 3:- unsuccesssful operation due to invalid account card.
TC 4:- successful entry of pin number.
TC 5:- unsuccessful operation due to wrong pin number entered 3 times.
TC 6:- successful selection of language.
TC 7:- successful selection of account type.
TC 8:- unsuccessful operation due to wrong account type selected w/r to that inserted card.
TC 9:- successful selection of withdrawl option.
TC 10:- successful selection of amount.
TC 11:- unsuccessful operation due to wrong denominations.
TC 12:- successful withdrawl operation.
TC 13:- unsuccessful withdrawl operation due to amount greater than possible balance.
TC 14:- unsucceful due to lack of amount in ATM.
TC 15:- un due to amount greater than the day limit.
TC 16:- un due to server down.
TC 17:- un due to click cancel after insert card.
TC 18:- un due to click cancel after insert card and pin no.
TC 19:- un due to click cancel after language selection,**account** type selection,withdrawl selection, enter amount

Program No.: 5

AIM: "Create a Test Plan document for any application(ex: library management system)

THEORY:

Def: Test Plan is a systematic approach to testing typically contain detailed information of what the work load will be, it deals with detailed information about upcoming testing efforts, scope of testing, testderiverables, staff and risk etc.

Template: A template is a file that serves as a strating point for a new document, when we open a template, it is pre-formatted in some way

Document: A piece of written, printed or electronic matter that provides information or evidence or that serves as an official record.

TEST PLAN IEEE FORMAT:

1. **Test Plan ID:** Unique No. or Id or Name of the test plan
2. **Introduction:** About the Project and testing
3. **Test Items:** Names of Modules/ Functions/ Services/ Features
4. **Features to Be Tested:** Responsible Modules for the Test Design
5. **Features Not to Be Tested:** Which ones to test and which ones not to test (e.g. Features of previous version of the Software)
6. **Approach:** List of testing techniques to be applied on the modules (prepared by QA/PM)
7. **Features Pass/Fail Criteria:** When above features are pass and when they fail
8. **Suspension Criteria:** Possible abnormal situations arose during testing of above features. Without recovering from these situations, you are not able to conduct testing. (Technical problems with respect to project)
9. **Test Environment:** Required hardware and software including testing tools to conduct testing
10. **Test Deliverables:** Required test documents to be prepared during testing (Test Cases, Test Procedures, Test Log, Test Report)
11. **Test Tasks:** necessary tasks to do before starting of every project testing
12. **Staff and Training Needs:** The names of test engineers and required training sessions
13. **Responsibilities:** Work allocation in terms of test engineers Vs Modules
14. **Schedule:** Dates and Times
15. **Risks and Mitigations:** Analyze risks and possible solution to overcome them
16. **Approvals:** Signatures of Test Plan Author and PM/QA

A Sample Test Plan Document for Internet Banking Application:

1. **Test Plan Id: IBS_ST_TP_001**
2. **Introduction:**

- ✓ It is system Test Plan for internet Banking System, internet web application, provides access to account holders and guest users from any where in the world.
- ✓ It has two interfaces one is Admin interface another is user interface.
- ✓ Admin can be accesses by Bank authorized users, user interface can be accessed by Bank account holders and guest users.
- ✓ The purpose of the system(Application) is to provide bank information and services online (through internet), Bank account holders can get banking services from anywhere, without visiting the bank branches.

3. Test Items:

- ✓ Admin Interface:
- ✓ Master Data
- ✓ User Interface
- ✓ Information
- ✓ Personal Banking
- ✓ Corporate Banking
- ✓ Business

4. References:

- ✓ Requirments
- ✓ Project Plan
- ✓ Test Strategy
- ✓ Use cases (if available)
- ✓ High level Design Documents
- ✓ Low Level Design Documents
- ✓ Process Guide line document
- ✓ Prototypes

5. Features to be tested:

a) Admin Interface:

I) Master Data

1. Add new branch, edit branch/Delete branch
2. Add new ATM
3. Add new loan type
4. Add new account type
5. Add new deposit type

II) User Management

1. Create New user
2. Edit user
3. Delete user

Etc…

III) Reports

1. Branch wise report

2. User wise report
3. Day, month,yearly reports
4. Service wise report(only loans, only new account, fixed deposits)

b) User Interface:

I)Information

1. Branch locators
2. ATM locators
3. Loans information
5. Bank history
6. Bank Financial details
7. Fixed deposits information
8. Calculators

Etc….

II)Personal Banking

Login
Balance enquiry
Bill payment (utilities, subscriptions0
Fund transfer (transfer to same bank, other banks)
Statement generation (mini statement, detailed report)
Etc….

IV) Corporate Banking

1. Add user, edit user, delete user
2. Balance enquiry
3. Money transfer
4. Payroll
5. Reports

Etc…

6. Features not to be tested: NA

7. Entry Criteria:

a) Test Design:

- Team formation, Responsibilities,schedule,requirements,test case template
- Training on domain, on automation tools

b) Test Execution

Readiness of test tab
Readiness of AUT
Requirements
Test case Documents
Test data
Defect Report Template
Etc….

8) Exit Criteria:

All possible test cases executed
Maximum defect fixed, final regression performed successfully

Confidence on test process
Time limitations
Budget limitations

9) Suspension criteria:

Show –stopper bug found
Supplier issues
Vast changes in requirements
If resolving defects are more

10) Roles and Resposibilities:

S.NO	NAME	ROLE	RESPONSIBILITIES	REMARKS
1	B.Vasundhara Devi	Test Lead	Test planning, guidance, Monitoring and test control	
2	A Ritvikasai	Sr. Tester	Test data collection, Generating test scenarios	
3	A vinay	Tester	Test case documentation, testexecution, defect reporting and tracking for admin module	
4	Srinivas V	Tester	Test case documentation, testexecution, defect reporting and tracking for Personal banking module	
5	Suneetha B	Tester	Test case documentation, testexecution, defect reporting and tracking for Corporate banking module	

11) Schedule:

SNO	TASK	DAYS	DURATION	REMARKS
1	Understanding and Analyzing requirements	5	2nd july to 6th july	
2	Review meeting	1	9th july	
3	Generating Test scenarios	10	11th july to 22nd july	
4	Reviews	02	25th july to 26th july	
5	Test case Documentation	40	29th july to 12th august	
6	Reviews	04	14th august to 18th august	
7	Test data collection	6	20th august to 26th august	
8	Reviews	1	28th august	
9	Verifying Test Environment Setup	1	29th august	
10	Create Test Batches	02	30th 31st Aug	
11	Sanity Testing	1	3rdseptember	
12	Comprehensive testing	25	4thsep to 2nd October	
13	Sanity Testing	1	3rd October	

14	Selecting Test Cases	2	4th and 5th October	
15	Regressing Testing	05	8th October to 12th October	
16	Sanity Testing	1	15th October	
17	Selecting Test Cases	1	16th October	
18	Regression Testing cycle -2	4	17th October to 22nd October	
19	.			
.	.			
.	.			
28	Final Regression	8	19th November to 28thnovember	
29	Evaluating Exit Criteria	1 or 2	29th , 30thnov	
30	Collecting all artifacts	2	3rd ,4th Dec	
31	Test Summary Report	1	5th Dec	

Note: Regression Testing depends on Application and strength of Development team.

12)Training:

- Training program on Banking Domain
- Test Automation Training Using HP UFT Tool

13) Risks and Miligations

- Team member's issues
- Vendors issues
- Time
- Budget

14) Test Environment/ Lab:

Application Type: Web Application, Internet and public

Server Side:

- Windows 2003 server
- UNIX server
- MS Exchange server a) webserver b) EDP c) Data storage
- Bugzilla tool
- VSS
- MS Office
- HP UFT Tool etc
- Browser IE 7.0

Client side:

- Windows xp+sp2
- VSS
- Ms-Office
- HP UFT

AUT Environment:

- .NET(c#,VC++,ADO)

- IIS- web server
- COM+ - APP server
- SQL server 2005 for database server

15) Test Deliverables:

- Test Plan
- Review reports
- RTM
- Test Scenario docs
- Test Case Docs
- Test data
- Opened, closed defect report
- Test summary report
-

16) Approvals:

SNO	TASK/S	AUTHOR/ RULE	DATE & SIGNATURE
1	Test plan documentation	Kareemulla (Test Lead)	
2	Review	Hari Prasad (Quality analyst)	
3	Approval	Vinod Rao (Project Manager)	

17) Glossary

AUT- Application Under Test
PIN- Project initiation note
SRS- Software Requirement Specification

Program No.: 6

AIM: Overview of any Test Management Tools (e.g. Test Director)

Introduction to TestLink

This is most widely used web based open source test management tool. It integrates both requirements specification and Test specification together. User can create test projects and document test cases using this tool. We can create account for multiple users/testers and assign different user roles. Admin user can manage test cases assignment task.

It supports both manual and automated execution of Test cases. With this tool the testers can generate Test Report and Test Plan Documents within a minute. It supports generation of Test reports in MS Word, Excel and HTML formats.

TestLink also support integration with many popular Defect Tracking systems like Mantis, BugZilla, Jira, Youtrack and TRAC. We can link a specific bug ticket with test cases. It also support and maintain multiple Test projects. Since it is a web based tool, multiple users can access its functionality at the same time with their credentials and assigned roles.

Benefits of TestLink

- Supports Multiple Projects.
- Easy Test Cases import or export.
- Easy to integrate with many defect management tools.
- Automated Test cases execution through XML-RPC.
- Easy to filter test cases with keywords, version and Testcase ID.
- Can provide credentials to multiple users and assign roles to them.
- Easy to assign test cases to multiple users.
- Easy to generate Test plan and Test reports in various formats.

TestLink Installation

Important Note: If you want to try your hand at TestLink demo before installing or using it for your project, head to below demo page. All features are available in this online latest demo version:
=>Go to TestLink Online Demo here.
Login name: admin
Password: admin

Pre-requisites:

1. Apache Web server
2. PHP
3. MySQL
4.

Note: *If you have System Admin assigned for your project you can ask sysadmin to install this for your project and skip directly to "Creating a Test Project" step. (If you have time to do these installations I will suggest try your hand at it. This will add to your experience. :))*
To install TestLink, You need to install Apache web server, PHP and MySQL server. If your system or server already has Apache, PHP and MySQL installed then you can directly install this tool as mentioned in "***Installation of* TestLink**" section below.
Otherwise, follow below steps one by one to install Apache, PHP, and MySQL first.

Installation of Apache:
This section contains steps to be followed to install Apache web server.

Step – 1
Download Apache from here.
Step – 2
Extract the Zip file to C:/
Step- 3
Copy C:\Apache24\bin path and append this with path environment variable.
Step – 4
Open Command Prompt. (Run as administrator)
cd to C:/Apache24/bin and run the following commands one by one
httpd -k install
httpd -k start

Installation of PHP
This section contains steps to be followed to install PHP and configuring PHP with Apache.

Step – 1
Download php from this page (thread is safe).
Extract this to *C:\php*
Step – 2
Rename php.ini-development to php.ini
Step -3
Add *C:\php* to path environment variable.
Step -4
Edit *php.ini* and add/edit the following lines,
display_errors =On
log_erros = On
doc_root = C:/Apache24/htdocs
extension_dir = "C:/php/ext"
Step – 5
Edit *httpd.conf* and add/edit the following lines,
LoadModule php5_module "C:/php/php5apache2_4.dll"
AddType application/x-httpd-php .php .html .php4 .php5
At the end of *httpd.conf* file add,
PHPIniDir "C:/php"
Step – 6
Create a file named *phpinfo.php* inside *C:/Apache24/htdocs* & add the following line inside the file.
<?php phpinfo();?>
Step – 7
Copy *php-mysql.dll* in *C:/php/ext* and put it inside *C:/Windows/System32*
Step – 8
Restart Apache.

Installation of MySQL
This section contains steps to be followed to install MySQL.

Step – 1
Download MySQL from this page and install it.

Step – 2
Create a Database named "testlink" in your MySQL server.
Step -3
In *php.ini* uncomment MySQL extension.
And set date time zone as "PRC".

Installation of TestLink

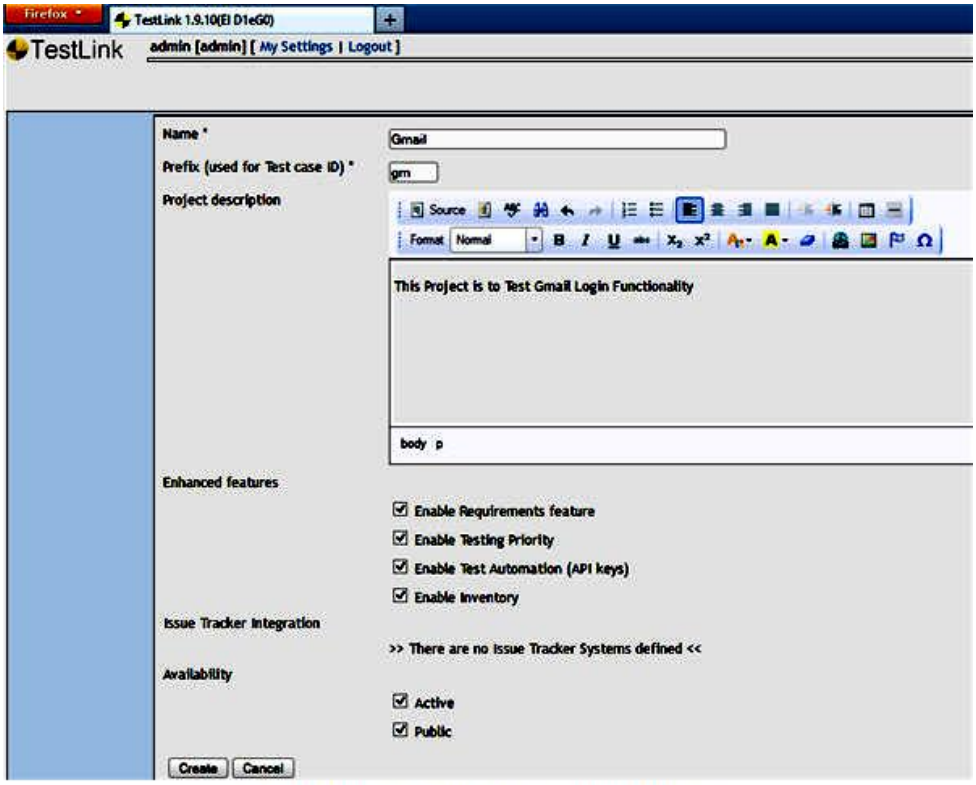
This section contains steps to be followed to install TestLink in Apache web server.

Step – 1
Download TestLink from this page.
Step – 2
Extract the package, rename it to "testlink" and put it inside "*C:/Apache24/htdocs*"
Step – 3
Edit *config.inc.php* and edit the following lines,
$tlCfg->log_path = 'C:/Apache24/htdocs/testlink/logs/';
$tlCfg->config_check_warning_mode = 'SILENT';
$g_repositoryPath = 'C:/Apache24/htdocs/testlink/upload_area';
Save the changes, and open http://localhost:80/testlink from the browser.
Click "next" button until installation completed.
Step – 4
Now you can log in to TestLink By accessing http://localhost:80/testlink.
Username: admin
Password: admin

Creating a Test Project

Test Project is a project created for a specific product/project. It contains Test plans, Test suites, Test cases and Builds. We can maintain multiple projects in TestLink.

This section clearly explains how to create a Test project:
Step-1: Creating a new Test Project requires "admin" rights. (So login with admin user)
Step-2: When logging for first time, it will directly navigate to Test project creation page. Otherwise select "Test Project Management" link on main page and click on "Create" button.
Step-3: Test Project section will be opened. Click on "Create" button.
Step-4: Enter Test Project name, description and all the required fields and click on "Create" button.
[Click to enlarge image]



TestLink Tutorial © www.SoftwareTestingHelp.com

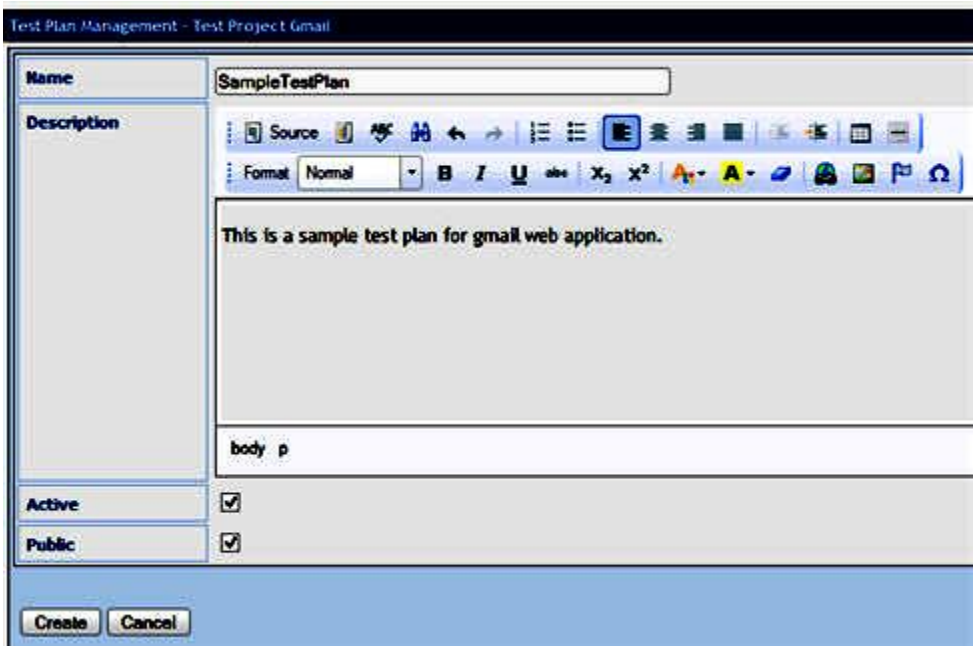
Test project should be created successfully.

Creating Test plan

Test plan document contains complete information like scope of testing, milestones, test suites and test cases. Once you've created a Test Project, Next step is creating a Test plan.

This section clearly explains creating Test plan:

Step-1: Click on "Test Plan Management" link on the desktop page.
Step-2: Click on "Create" button on the test plan management page.
Step-3: Enter all the required details in the page.



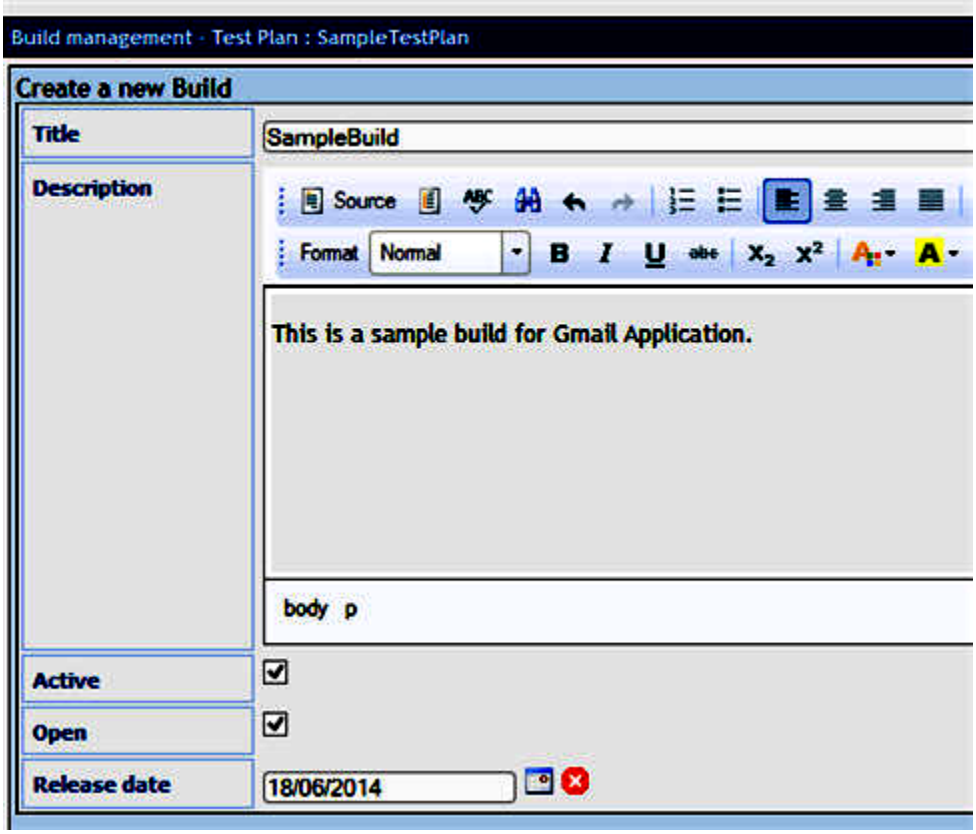
Step-4: Click on "Create" button. It should be saved.

Build Creation

Build is a specific release of software.

This section clearly explains how to create a build:

Step-1: Click on "Builds/Releases" as shown in figure.
Step-2: Click on "create" button.
Step-3: Enter the details about the build as shown in below figure and click on "Create" button.



Build should be created and saved.

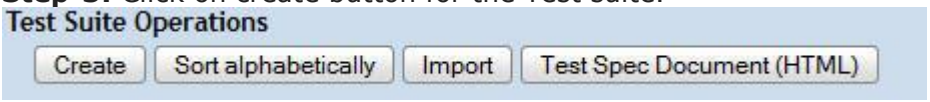
Creating Testsuite

Testsuite is a collection of test cases that may be validating/testing same component. This section clearly explains how to create a testsuite. Follow the below steps one by one to create a Testsuite.

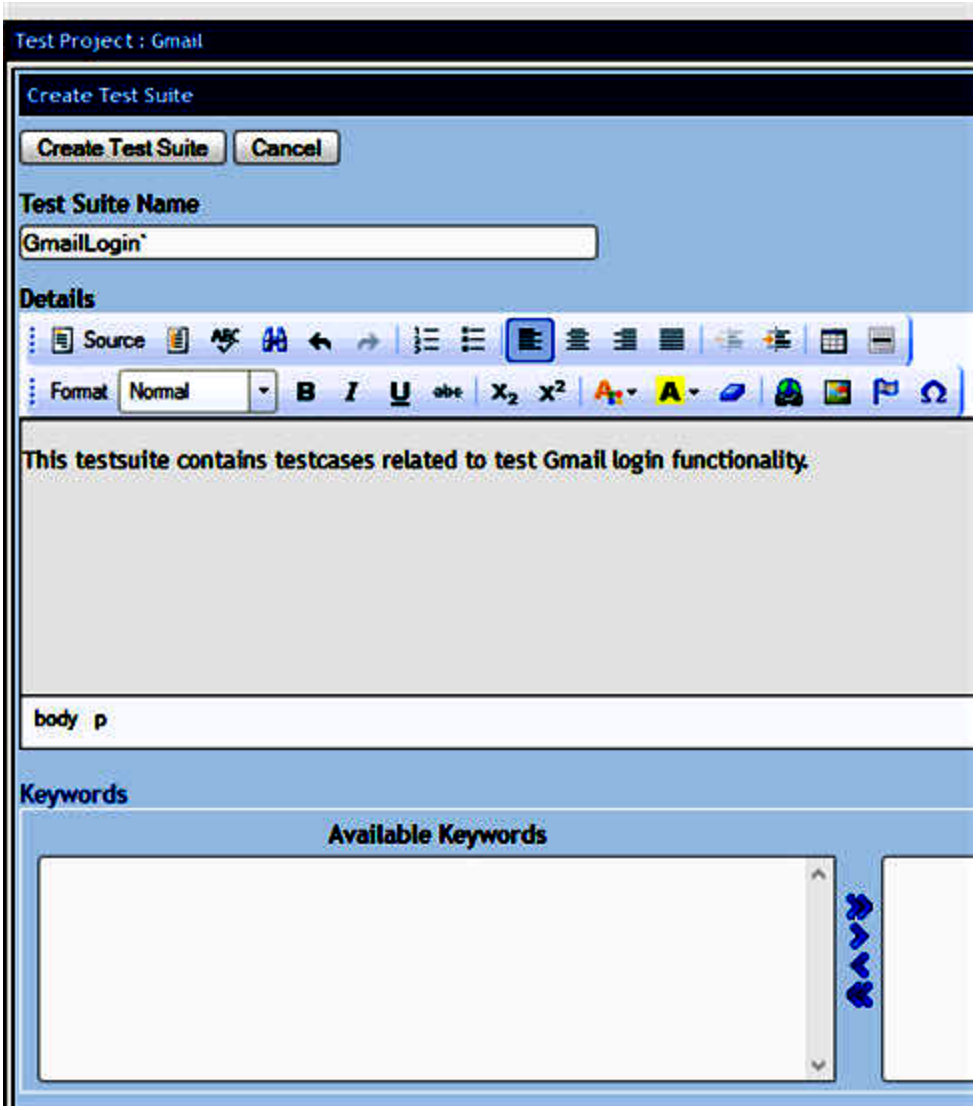
Step-1: Click on "Test specification" link on the Desktop. It should navigate to the Test specification page of the Project.

Step-2: Click on the settings icon  on the right side panel. It will display a series of Test suite operations.

Step-3: Click on create button for the Test suite.



Step-4: Test suite specification page should be opened. Fill the details related to Test suite and click "Create Test suite" button.



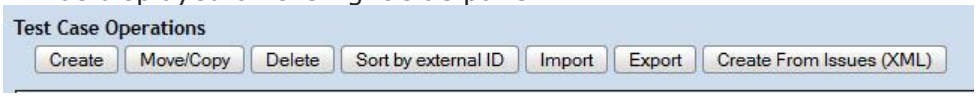
Test suite will be created and appear on the left side, folder tree structure.

Creating a Testcase

Testcase contains a sequence of test steps to test a specific scenario, with expected result. This section explains how to create a testcase along with test steps. Follow the below steps one by one to create Test cases.

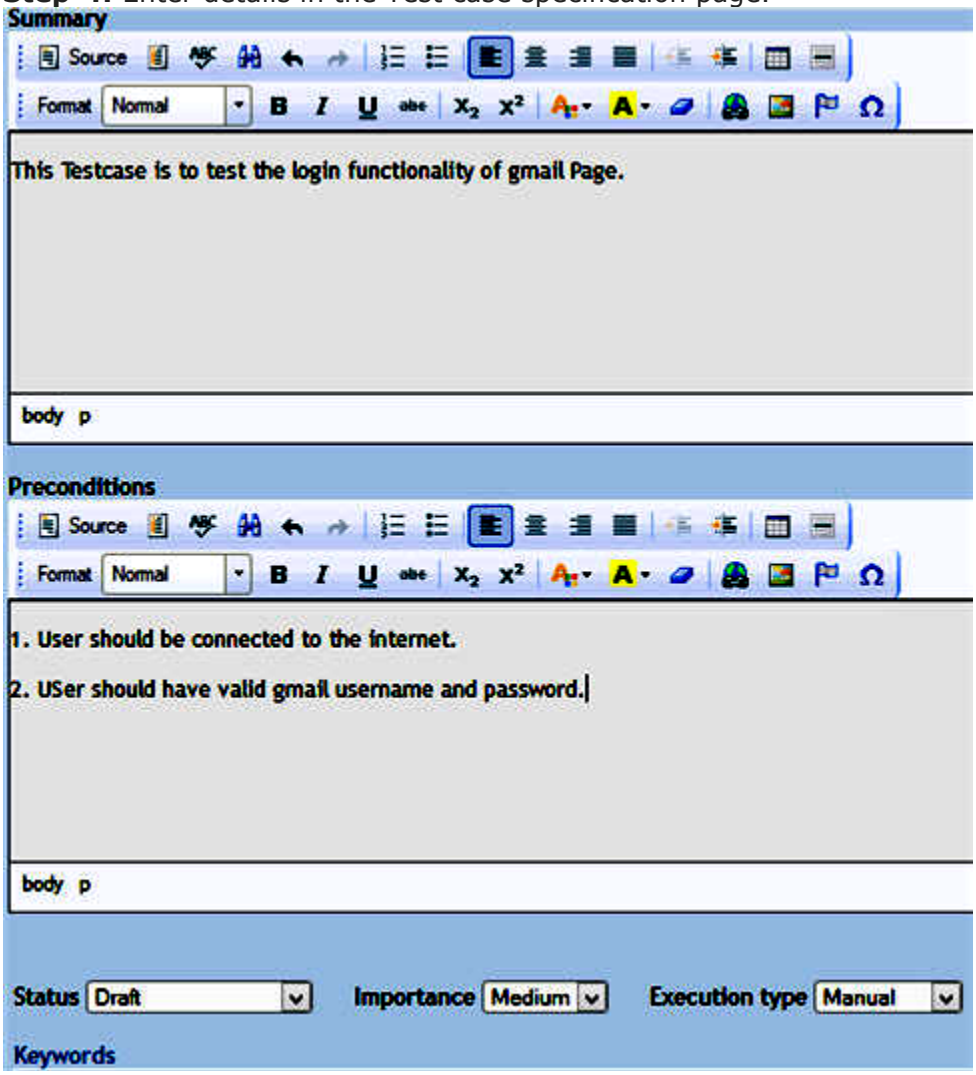
Step-1: Click on the Test suite folder on the left side tree structure

Step-2: Click on the settings icon  in the right side panel. List of Test case operations will be displayed on the right side panel.

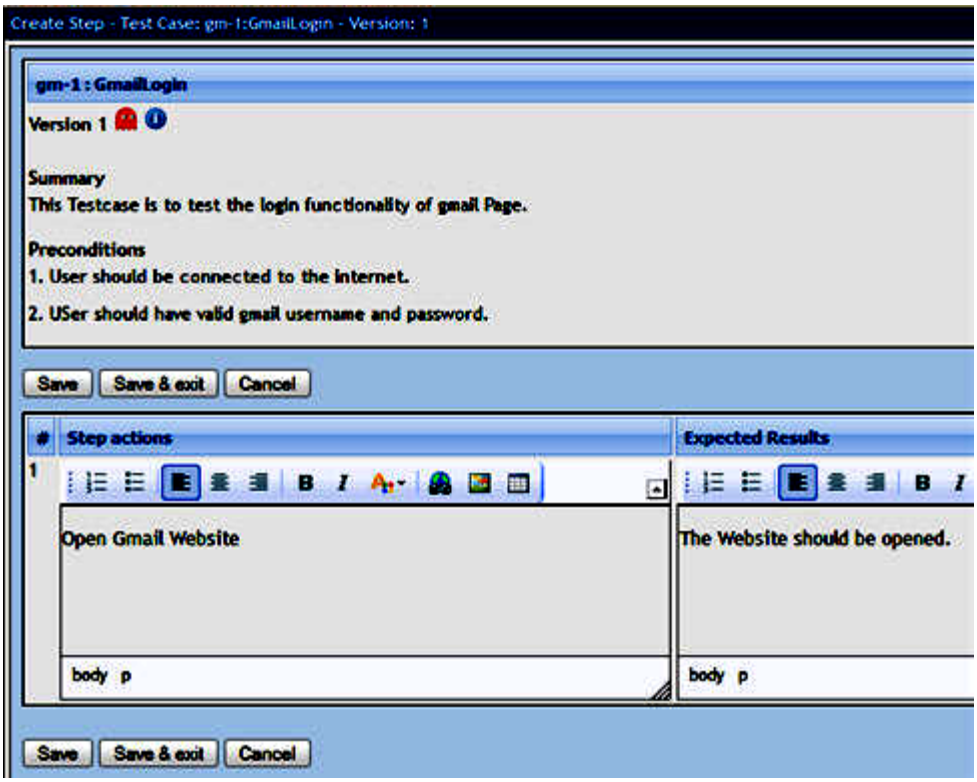


Step-3: Click on Test case "Create" button. It will open Test case specification page.

Step-4: Enter details in the Test case specification page.

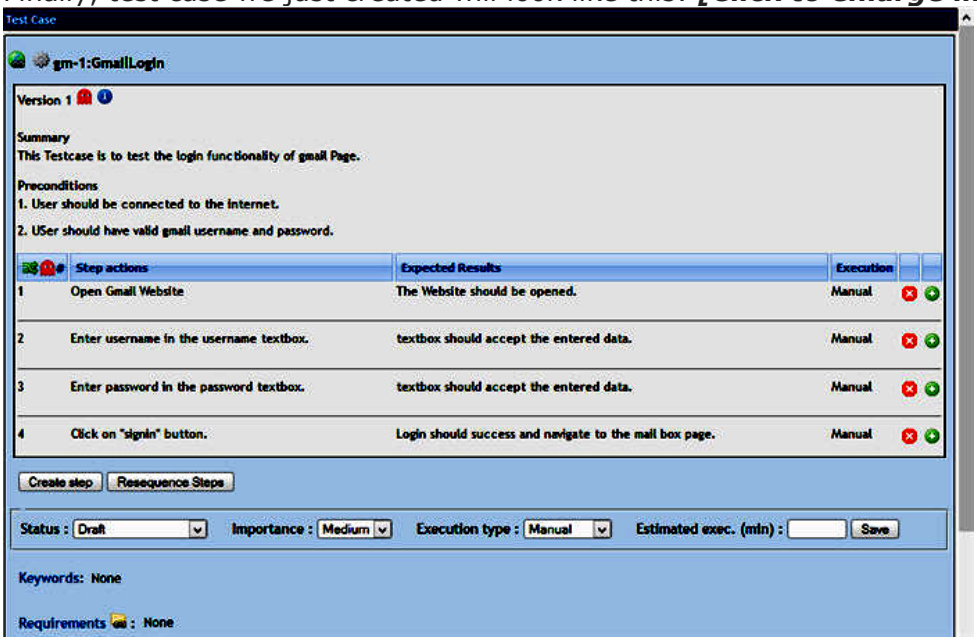


Step-5: After entering the details, click on "Create" button. It will save the test case.
Step-6: Now click on the "Create steps" button in test case. It will show test case step editor.
Step-7: Start adding the Test steps as shown in the picture.



Step-8: Click "Save" button to add further steps, or click "Save & exit" button to save step and exit from the editor.

Finally, test case we just created will look like this: ***[Click to enlarge image]***

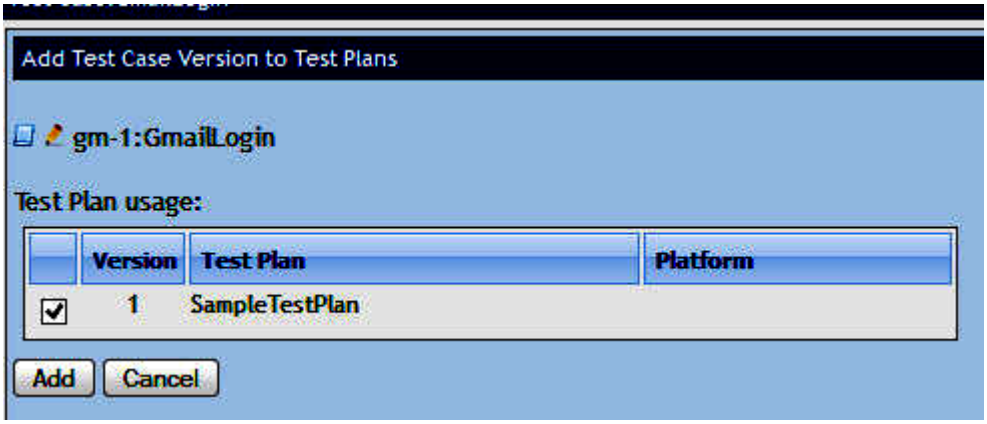


Assigning Testcase to Test Plan:

To Execute a Test case, it should be assigned to a Test plan. This section describes how to assign a testcase to a Test Plan. Follow the below mentioned steps to assign a test case to a Test Plan.

Step-1: Click on the settings icon on the test case panel. It will display list of operations.
Step-2: Click on "Add to TestPlans" button.

Step-3: Click on the check box of test plan, to which you want to assign the test case. And then click on "Add" button.



Test case will be successfully added to the Test Plan.

Program No.: 7

AIM: Study of any Testing Tool (ex:UFT,QTP)

HP UFT:

THEORY:

- HP Unified Functional Testing, popularly know by its acronym HP UFT is the flagship functional automation testing toolfrom Mercury Interactive now acquired by HP.
- It is an icon based tool, which automates the functional & regression testing of an application
- HP UFT is easier to use and implement for both technical &non technical testers in comparison to other functional testing tools available.
- HP UFT's Scripting Language is VB Script which is easy to use, understand and program
- HP Unified Functional Testing enables Business Process Testing (BPT)
- Supports large pool of software development environments like SAP, Web, Oracle etc.
- The trainings have been recorded using HP UFT version 9.5 but you may use any higher or lower versions for your learning purposes
- For All Hands On in these trainings, we will be using "Flight Reservation" Application which comes bundled with HP UFT
- Uses "Active Screen "Technology to record script which aids the tester in reffering to the screens object properties.

- Library files contain VBScript functions and subroutines that can be added to the test.

How to Launch HP UFT and Tool Bar and Menu bar in HP UFT:

- To launch HP UFT, In Start Menu, Choose Programs >HP Unified Functional TestingFolder >HP Unified Functional Testing
- The first time you start HP UFT, the **Add-in Manager dialog box** opens. It **Displays list of all installed add-in along with license expiry date**.
- It is recommended you **select only the add-ins required for your particular testing session.** Because at times, different add-in interferes with each other degrading object identification and HP UFT's performance. HP UFT will remember the add-ins you load so that the next time you open HP UFT the add-ins you selected in the previous session are selected by default. Also, if you do not want this dialog box to open the next time you start HP UFT clear the Show on startup check box. Click OK.
- The **Start Page describes the new features** in this release—including links to more information about these features.It also provides links to Process Guidance, a tool that offers best practices for working with HP UFT.You can open a document from the list of Recently Used Files,or you can click the buttons in the Welcome! area to open new or existing documents.If you do not want HP UFT to display the Start Page when you next open HP UFT ,select the "Don't show the Start Page window on startup" check box.When you select this option, the Start Page is also automatically hidden for the current HP UFT session as soon as you open another HP UFT document.
- To display the Start Page again, select View > Start.
- **Title Bar** Displays the **name of the active document.** If changes have been made since it was last saved, an asterisk (*) is displayed next to the document name in the title bar.
- **Menu Bar** Displays menus of **HP UFT commands**.
- Toolbars Contains buttons to assist you in managing your document
- **Document Tabs** Enables you to **navigate open documents**
- **Keyword View** Displays test steps in a **graphical representation**
- **Expert View** Displays test steps as a **VB Script line.**
- **Active Screen** Provides a **snapshot of your application** as it appeared when you performed a certain step during the recording session.
- **Data Table** Assists you to **parametrize** your test.
- **Test Flow** Displays the **hierarchy of actions** and action calls in the current test, and shows the order in which they are run.

Objects Identification in HP UFT:

- HP UFT also uses a **"human" like technology** for object identification
- During Record Time HP UFT tries to learn properties of a GUI object on which operation is performed.

- During Run-Time HP UFT **compares** the **stored object properties with actual properties of object** available on screen and uniquely identifies an object independent of its location on screen
- The stored object and together with its properties is called **TEST Object**
- During Run-Time, the actual object available on the application under test is called **Run-Time Object**
- This is Quick Tests **"Test Object Model"**
- Information about the Test Objects is stored in **Object Repository**
- **Add-ins help in instructing Quick Test in advance of the nature of object to be recorded** so it as to reduce the time required to learn its properties

Understanding Expert View:

- In the **Expert View, each line represents a Test Step in VB Script**
- An **Object's Name is displayed in parentheses following the Object Type**.Herethe Object Name is Login and Object Type is Dialog
- **Objects in Object Hierarchy are separated by a "dot". Here** Dialog and WinEdit are fall in the same Object Hierarchy
- Just to put things in perspective, Object Hierarchy is Object Oriented Concept where set of objects that are grouped together in a parent-child relationship. In our case Dialog Box is the Parent Object and WinEdit is the Child Object
- The **Operation performed on the object is always displayed at the end of the statement** followed by any values associated with the operation. Here the word "Mercury" is inserted in the AgentName Edit Box using the Set Method
- **Syntax for a statement in expert view is** GUI object on which the operation is performed along with its complete hierarchy followed by the Operation on the Object and value associated with that Operation

ParentObject(Name).ChildObject1(Name)….ChildObjectN(Name).Operation"Value"

Keyword view Vs Expert View:

- The **Keyword View is comprised of a table-like view** where
- **Each step** is a **separate row** in the table and
- **Each column** represents **different parts of the steps**.
- **Item Column** contains the **item on which you want to perform the step**. This column uses icons displays the hierarchy of the GUI object on which operation is performed
- **Operation Column** contains the **operation to be performed** on the item.
- **Value Column** contains the **argument values** for the selected operation,
- **HP UFT automatically documents** each step for easy understanding in the **Documentation Column**

- These 4 columns are default but you can also use assignment & comment columns in Keyword View
- You will observe that the same object hierarchy is displayed in both Expert & Keyword Views and they map to the same operation and argument value.
- Essentially, **Keyword & Expert view contain the same data but arranged in different format**.

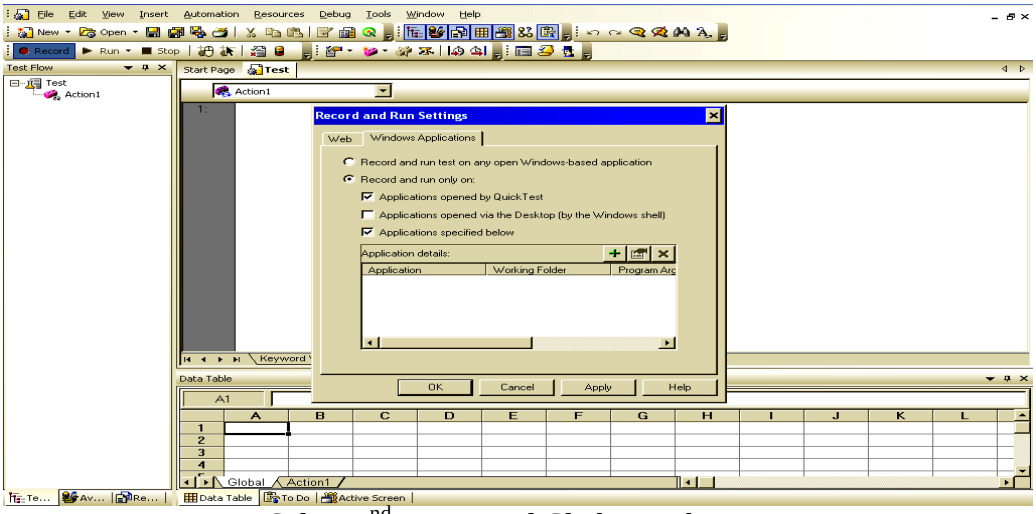
Program 7.1

AIM: Demonstrate the working of Recording Modes in HP UFT.

1. Normal Recording Mode or Context Recording Mode:

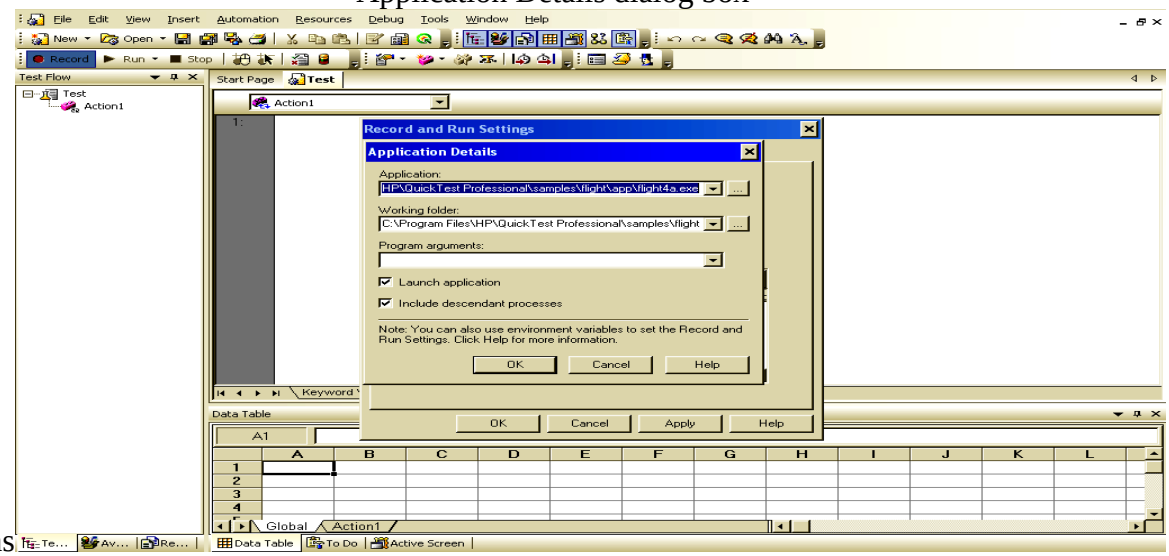
Steps:

1. Click on Record button
2. Record and Run Setting dialog box opens, as shown below



Select 2nd option and Click on + button

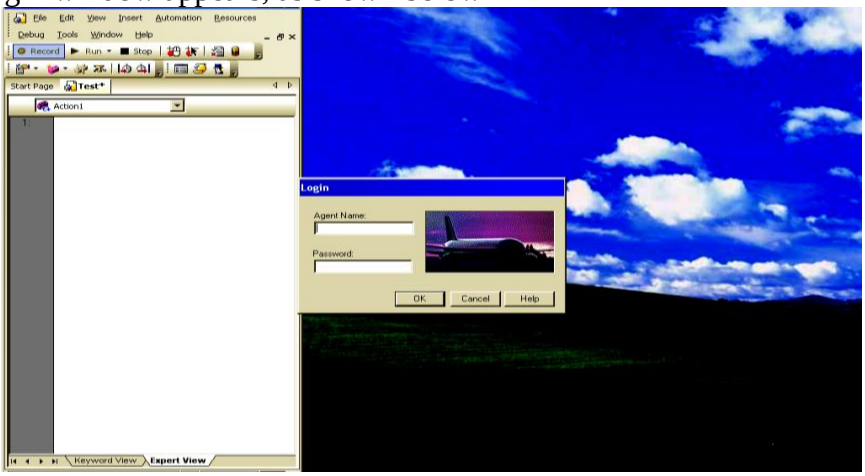
Application Details dialog box



opens

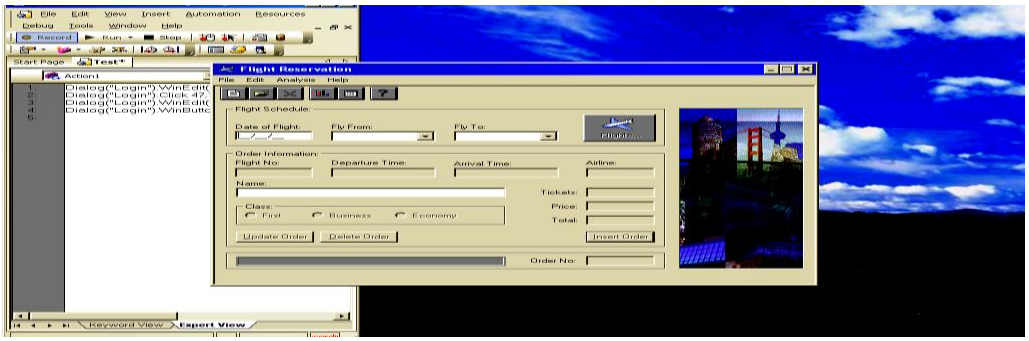
Click on OK button

3. click on Apply button and click on ok button
4. Then Login window appears, as shown below

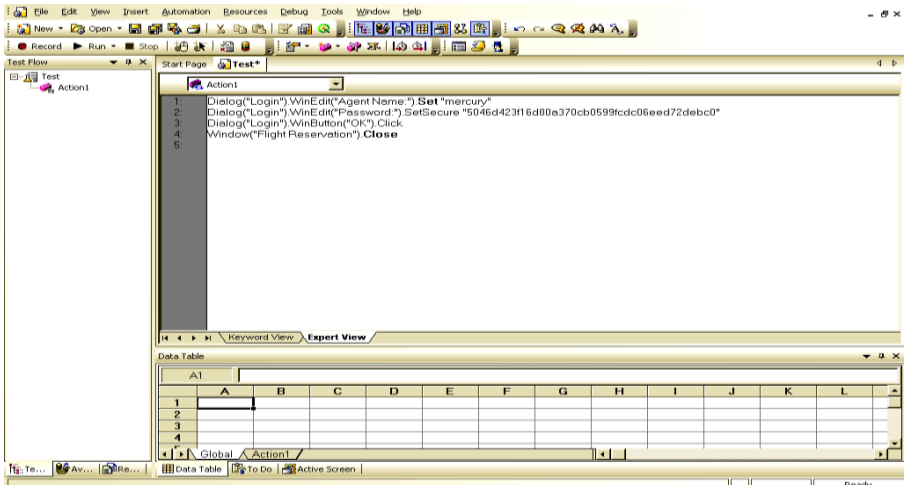


Give Agent name as: mercury, Password as: mercury and Click on OK button

4. Flight Reservation dialog box opens, as shown below

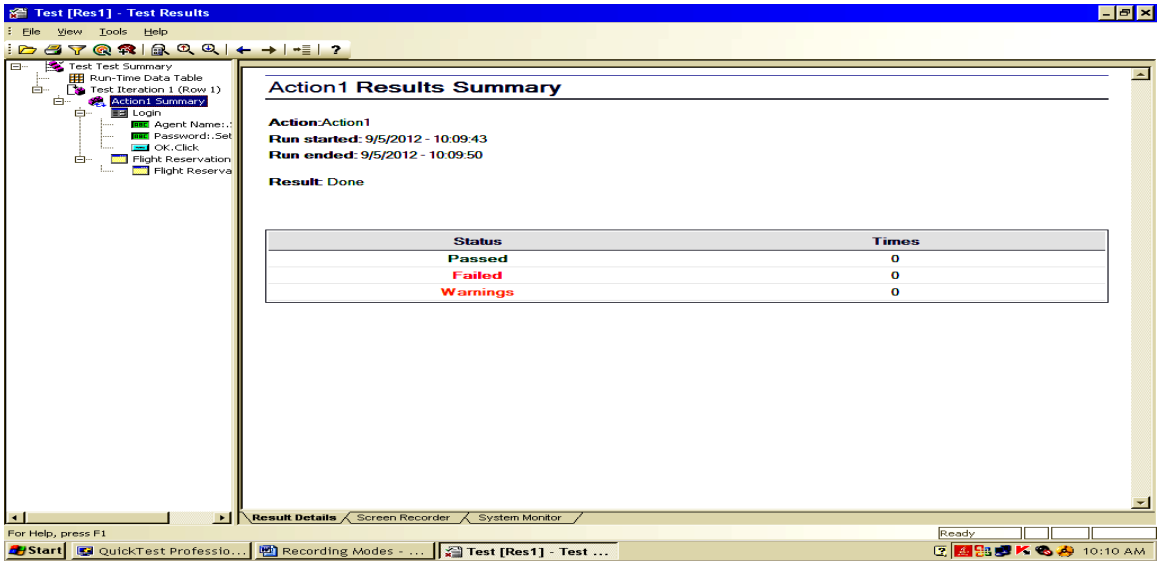


5. Close the flight reservation
6. In HP UFT, STOP recording. You get the script like below



7. Run the script

Results:



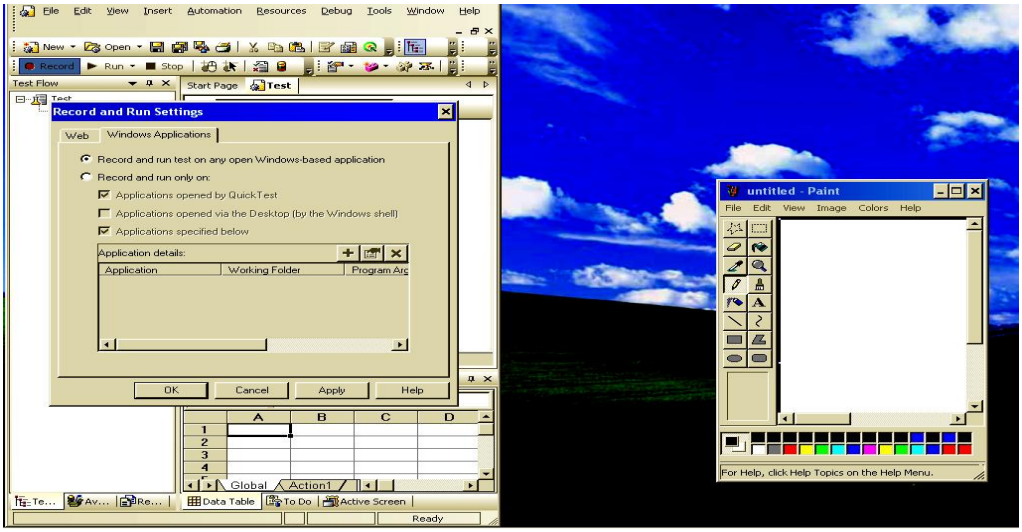
2. Analog Recording mode: For Paint

Steps:

1.To open paint

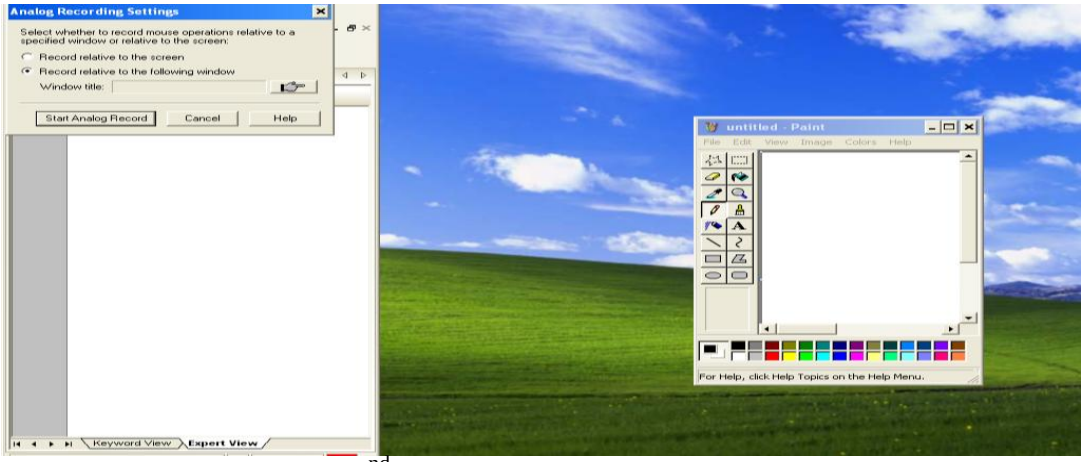
select ->start ->programs -> Accessories -> paint

2. In HP UFT, Click on Record button
3. Record and Run Setting dialog box opens, as shown below



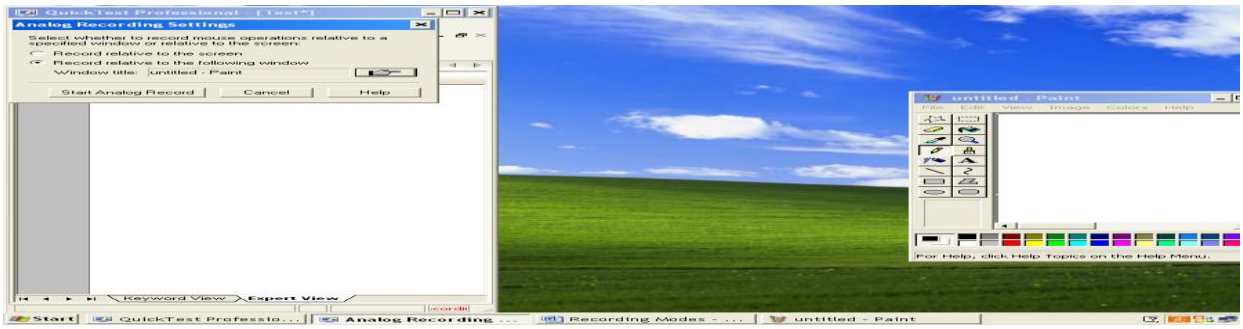
Select window application, select 1st option, Click on ok button

4. In HP UFT, select -> Record -> analog recording mode
5. Then Analog Recording Setting dialog box opens



Select 2nd option and Click on Hand icon

6. Cursor changed to hand icon, so click on untiled paint blue shaded area as shown below

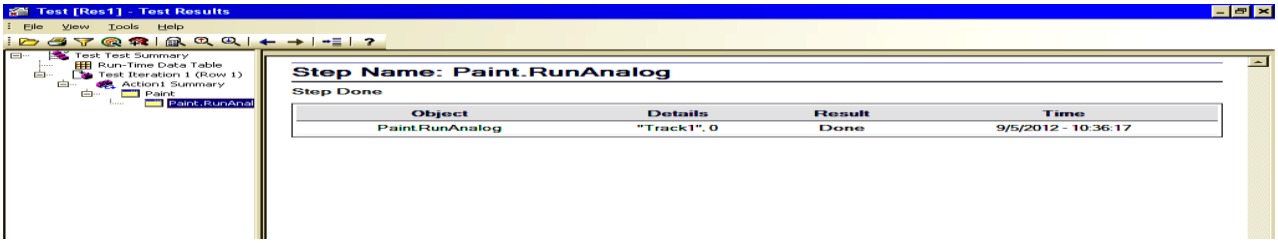


8. click on Start analog record, in analog record setting dialog box
9. In untitled paint, sign your signature and select file -> exit
10. In HP UFT, stop recording. Then you get the script as below

Window("Paint").RunAnalog "Track1"

11. Run the script, analyze the execution and difference between normal recording moto analog recording mode

Results:

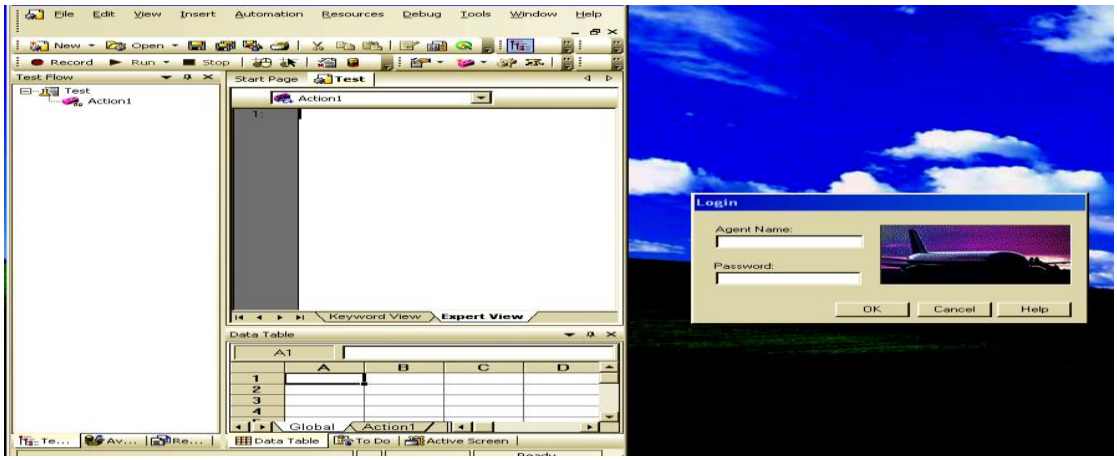


Analog Recording Mode: For Window Application

Steps:

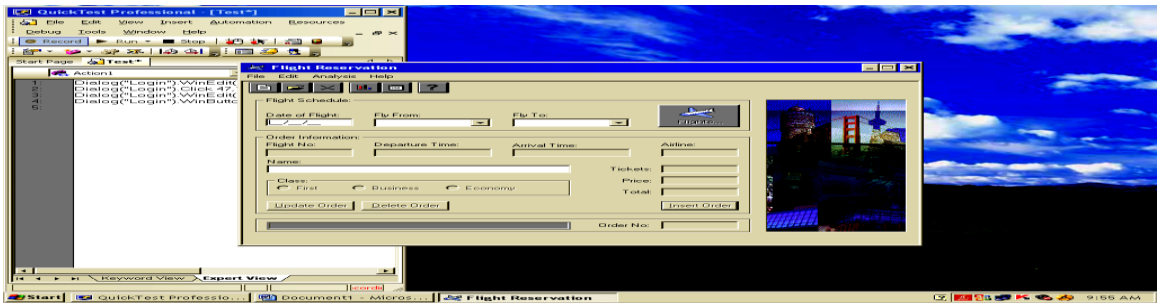
1. Open Flight Reservation window as below

   Start -> Programs ->HP Unified Functional Testing -> Sample Application -> Flight
2. Login dialog box opens as shown below

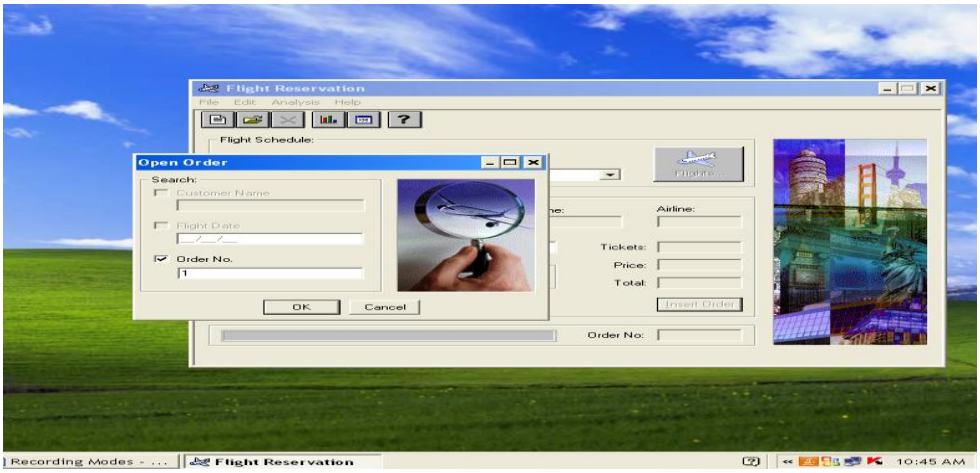


Give Agent name as: mercury, Password as: mercury, Click on OK button

3. Flight Reservation dialog box opens, as shown below

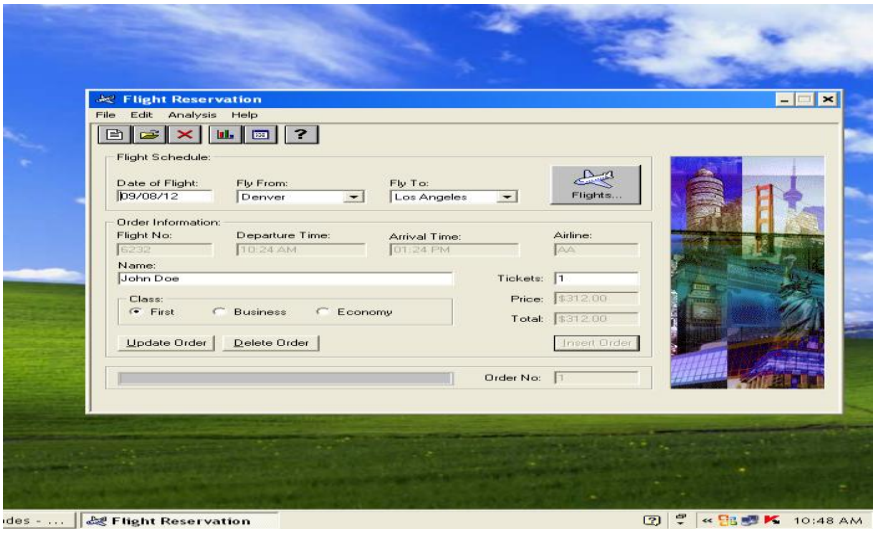


4. In Flight Reservation Select File -> Open Order
5. Open Order dialog box opens, as shown below

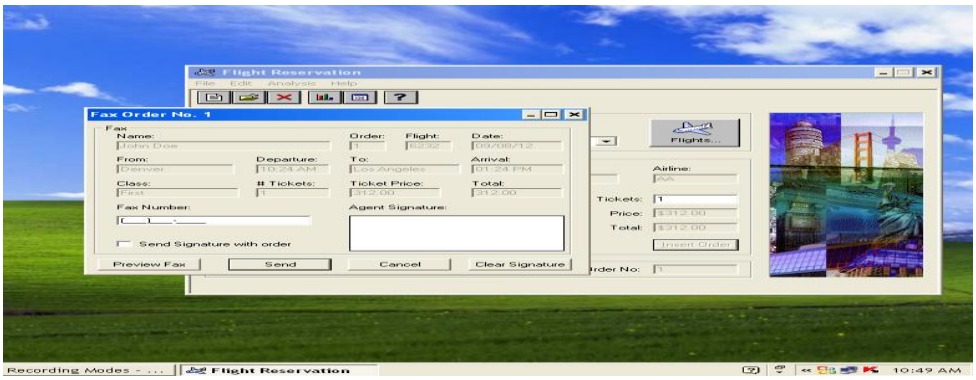


Click on Order No, Give order No :1, Click ok button

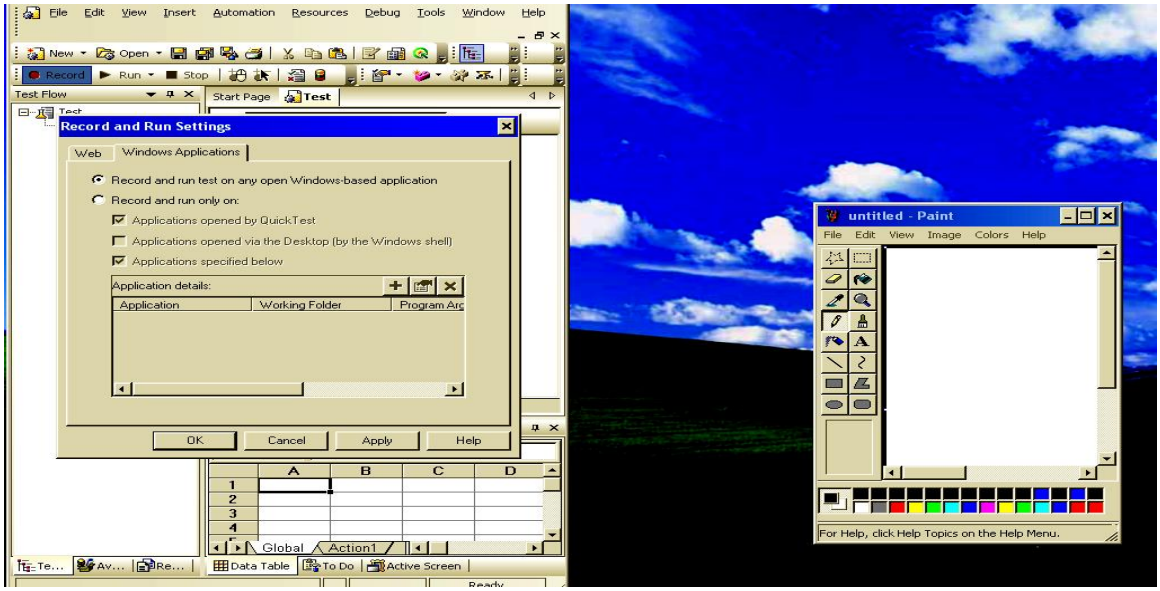
6. Then order no is opened, as shown below



7. select File -> Fax Order as shown below



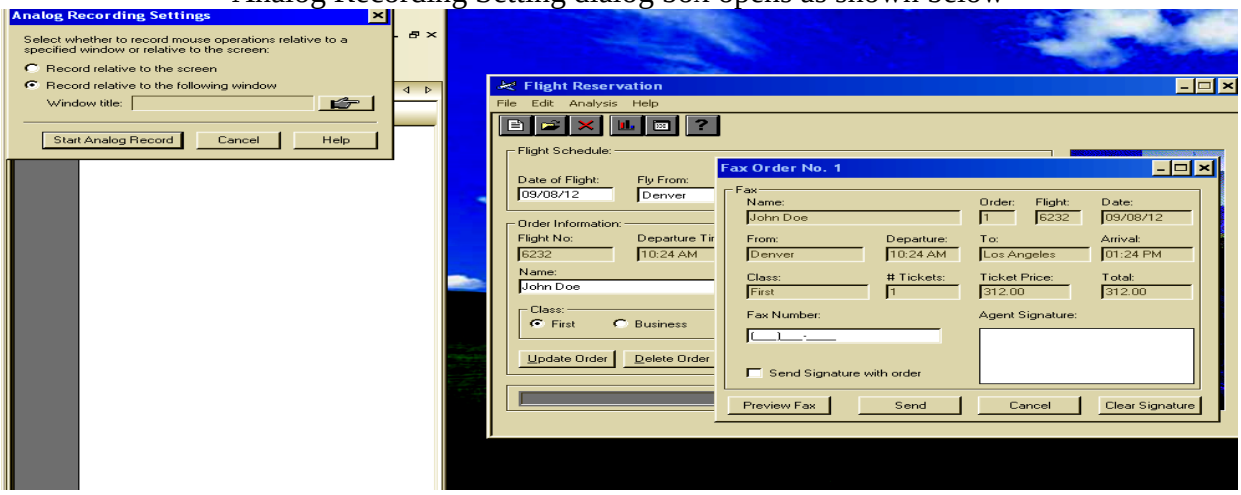
8. In HP UFT, click on Record button
9. Record and Run Setting dialog box opens, as shown below



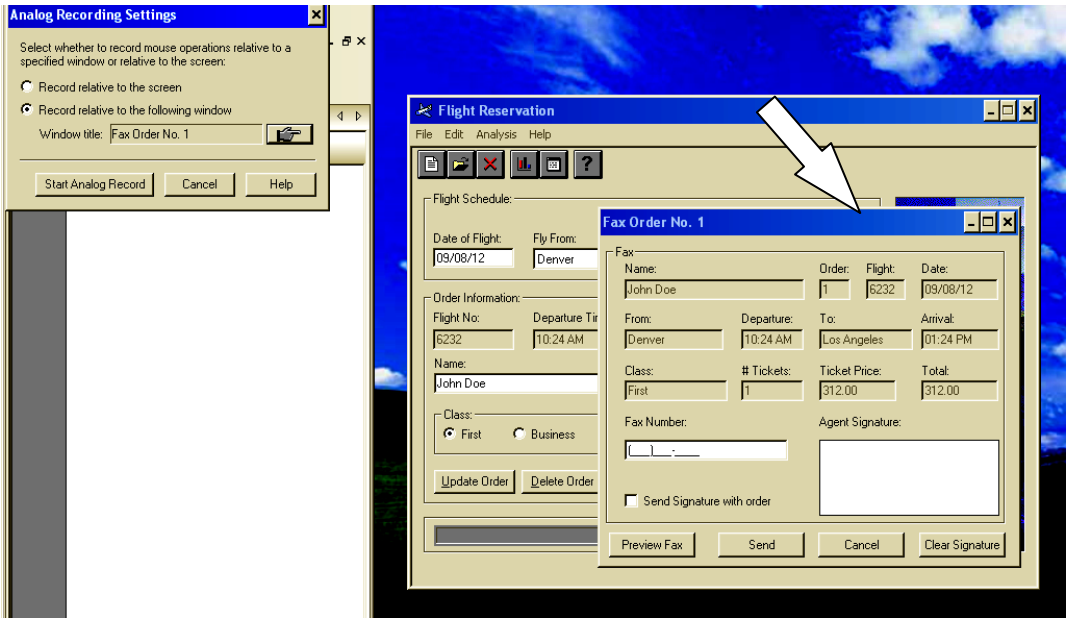
Select window application, Select 1st option, Click on ok button

8. select ->Record -> analog recording mode

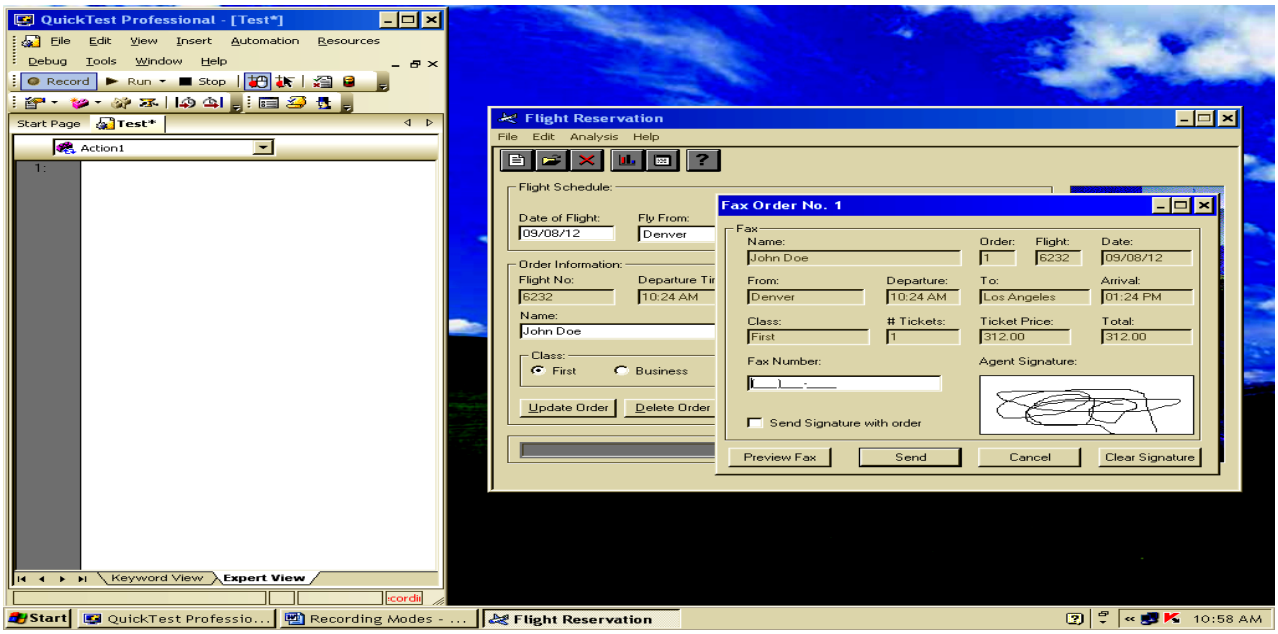
Analog Recording Setting dialog box opens as shown below



10. select the 2nd option (i.e Record Relative to the following window)
11. click on hand icon
12. cursor changed to hand icon, then click on Fax order:1 blue shaded area, as shown below



13. click on Start analog record on AnalogRecording Setting dialog box
14. and sign your signature on Agent signature area in Fax order No :1 dialog box, as shown below

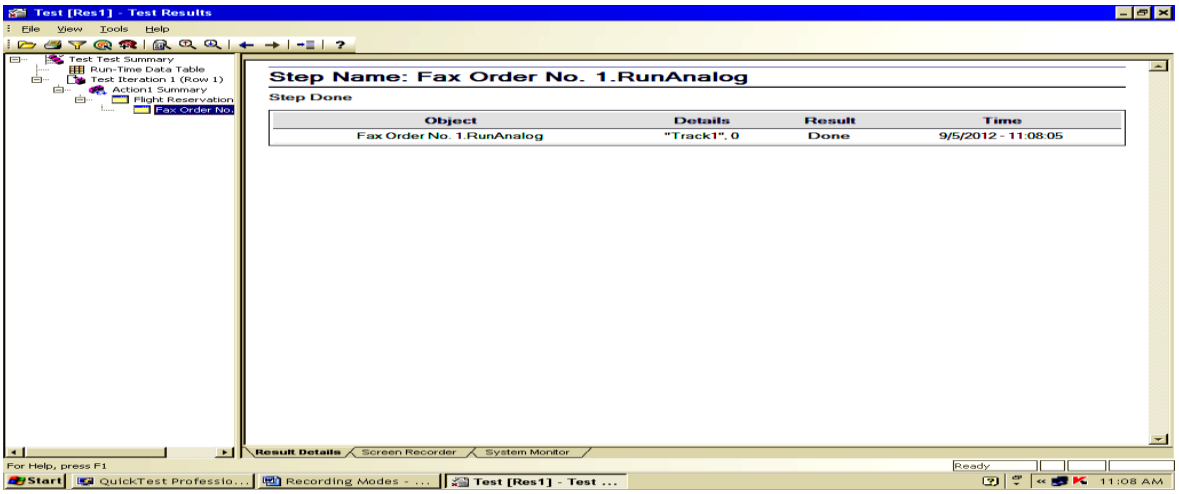


19. In HP UFT, stop Recording and you get the script like below

Window("Flight Reservation").Window("Fax Order No. 1").RunAnalog "Track1"

20. Run the script, analyze the execution and difference between normal recording mode to analog recording mode

Results:



Analog Recording mode: For Multiple window

Steps:

1. First open Paint and Fax order window as below steps
2. To open paint

select ->start ->programs -> Accessories -> paint

3. Open Flight Reservation window as below

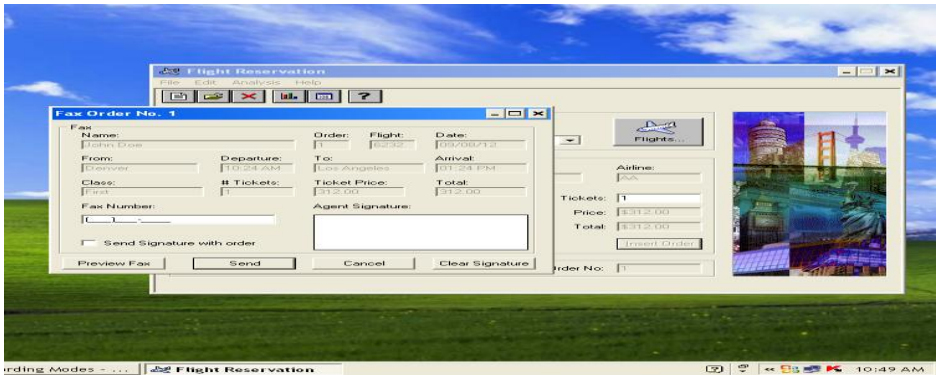
   Start -> Programs ->HP Unified Functional Testing -> Sample Application -> Flight
4. Login dialog box opens

   Give Agent name as: mercury, Password as: mercury, Click on OK button
5. Flight Reservation dialog box opens, as shown below
6. In Flight Reservation Select File -> Open Order

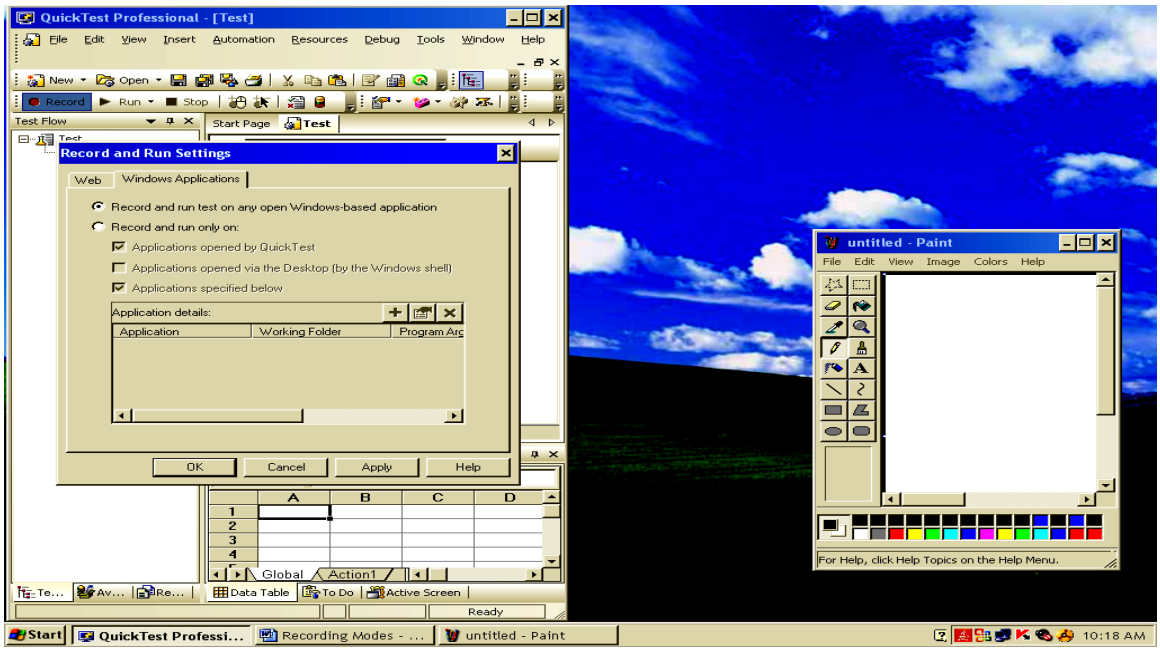
5. Open Order dialog box opens

   Click on Order No, give order No :1, Click ok button

15. then order no is opened, as shown below
16. select File -> Fax Order as shown below



17. In HP UFT, Click on Record button
18. Record and Run Setting dialog box opens, as shown below



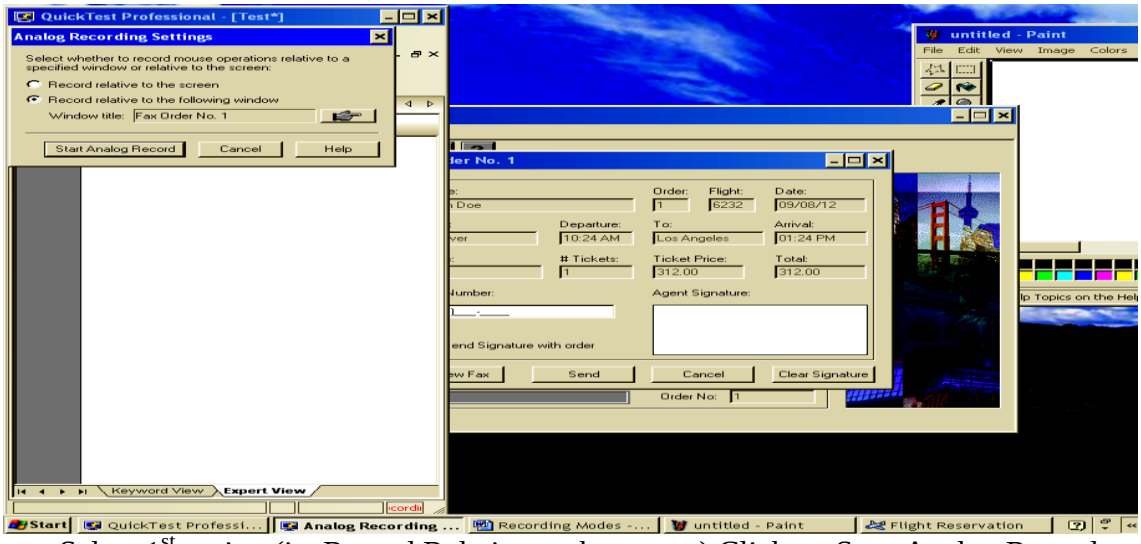
Select window application, Select 1st option, Click on ok button

19. In HP UFT, select ->Record -> analog recording mode

6. Then Analog Recording Setting dialog box opens

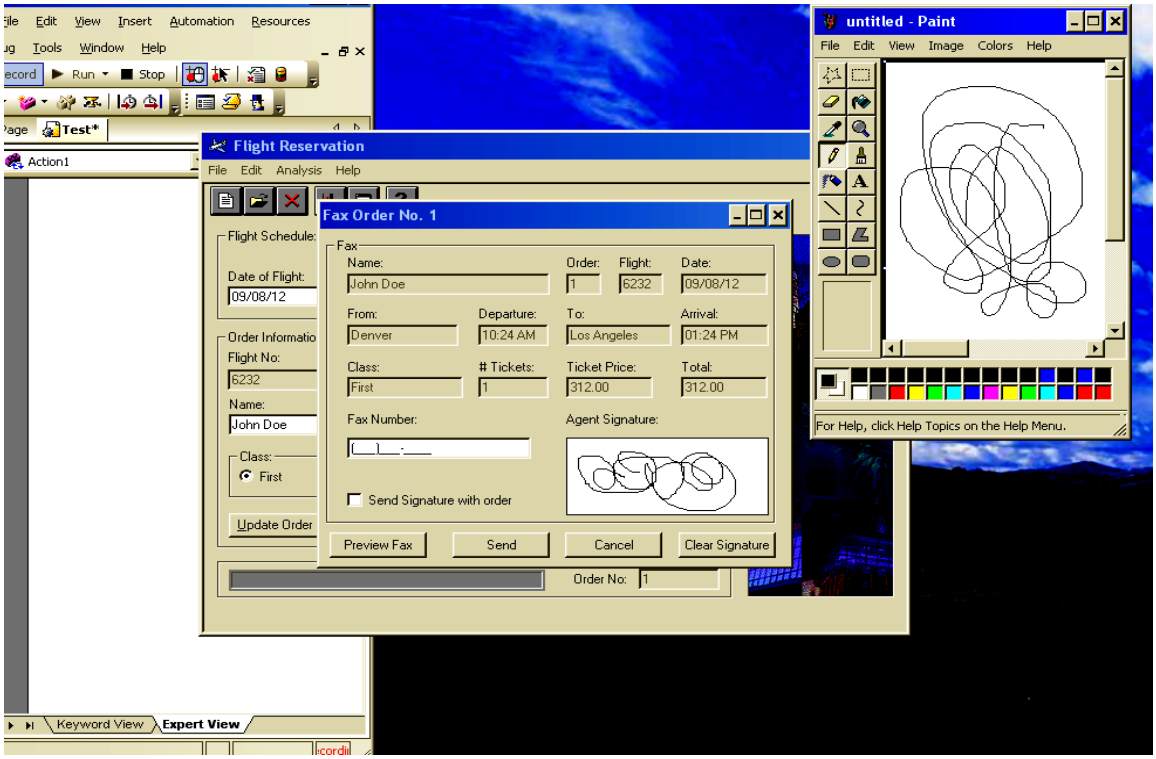
20. In HP UFT, Click on Record button

21. Record and Run Setting dialog box opens, as shown below



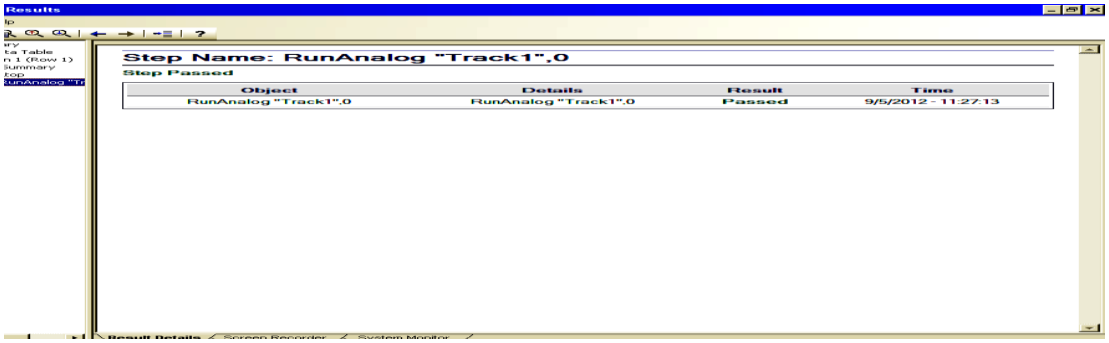
Select 1st option (i.e Record Relative to the screen) Click on Start Analog Record

6. And sign your signatures on Fax order:1 and paint as shown below



7. In HP UFT, Stop Recording
8. Run the script

Results:

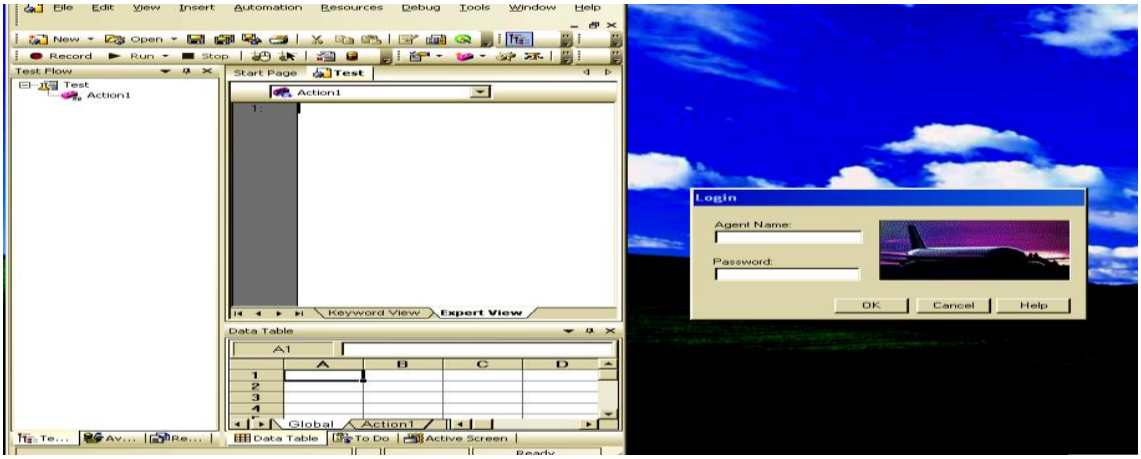


3. Low Level Recording Mode:

Steps:

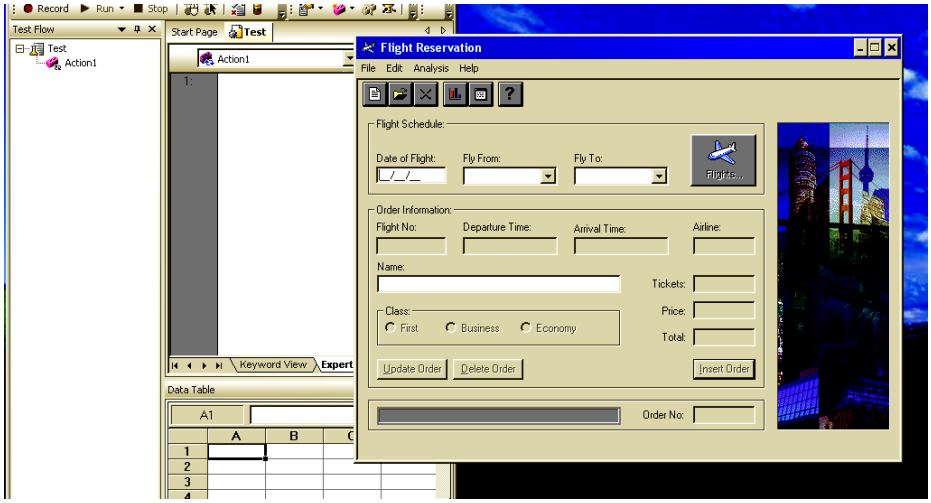
1. Open Flight Reservation window as below
   Start -> Programs ->HP Unified Functional Testing -> Sample Application -> Flight

2. Login dialog box opens as shown below

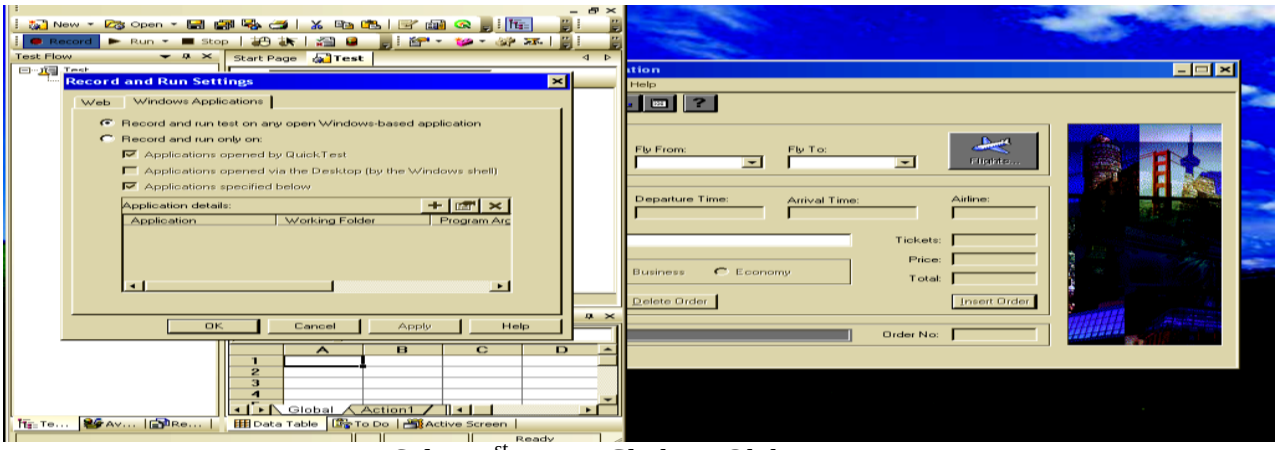


Give Agent name as: mercury or Your name or Your rollno, Password as: mercury, Click on OK

3. Flight Reservation dialog box opens, as shown below



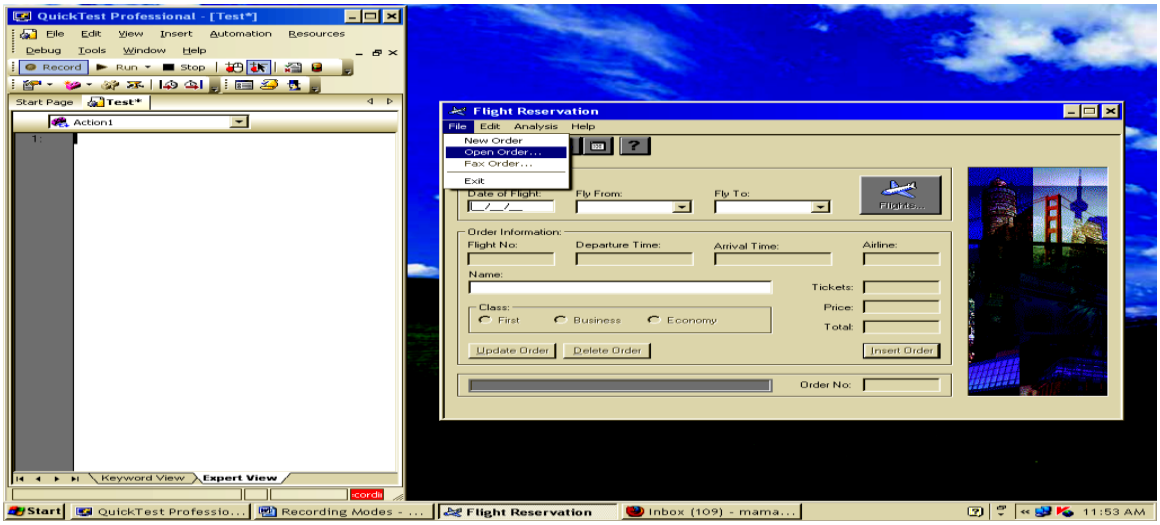
4. In HP UFT, Click on Record button
5. Record and Run Setting dialog box opens, as shown below



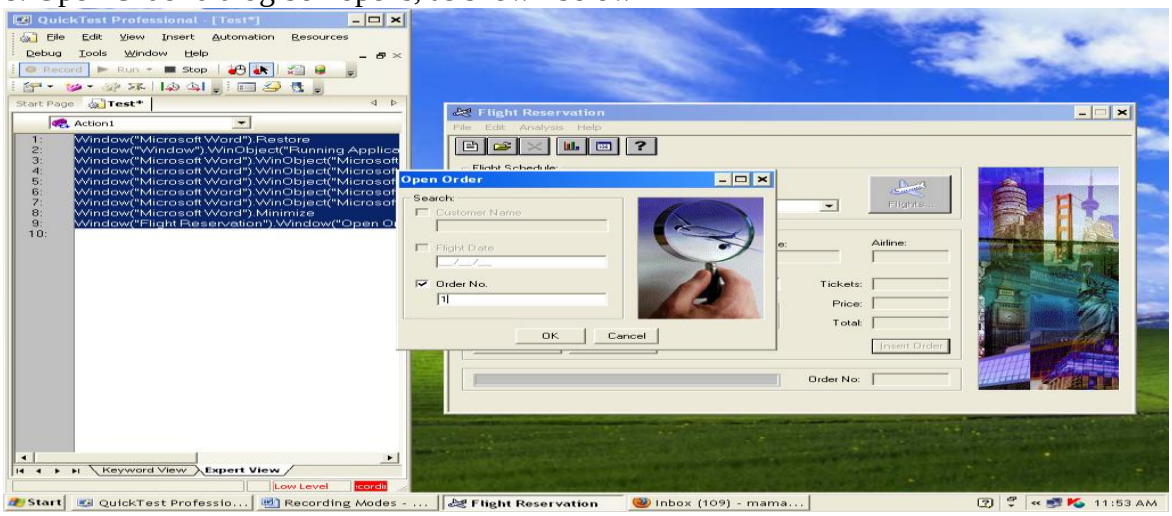
Select 1st option Click on Ok buttion

6. In HP UFT, select ->Record -> Low Level recording mode

7. Then in flight Reservation dialog box,Select file -> open order



8. Open Order dialog box opens, as shown below



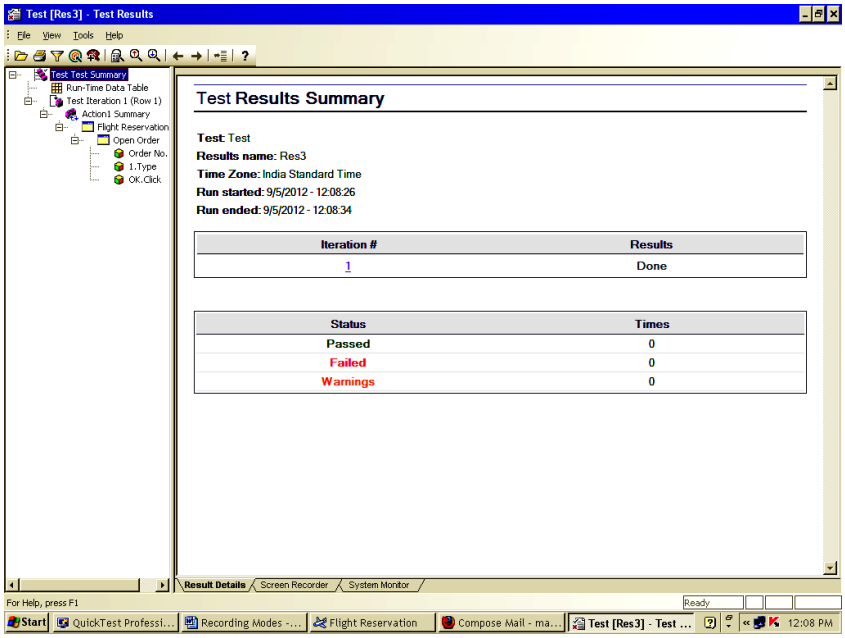
Click on Order No, give order No: 1, Click on OK button

8. In HP UFT stop recording, you get script like this

Window("Flight Reservation").Window("Open Order").WinObject("Order No.").Click 6,9
Window("Flight Reservation").Window("Open Order").WinObject("1").Type "1"
Window("Flight Reservation").Window("Open Order").WinObject("OK").Click 30,11

9. stop recording

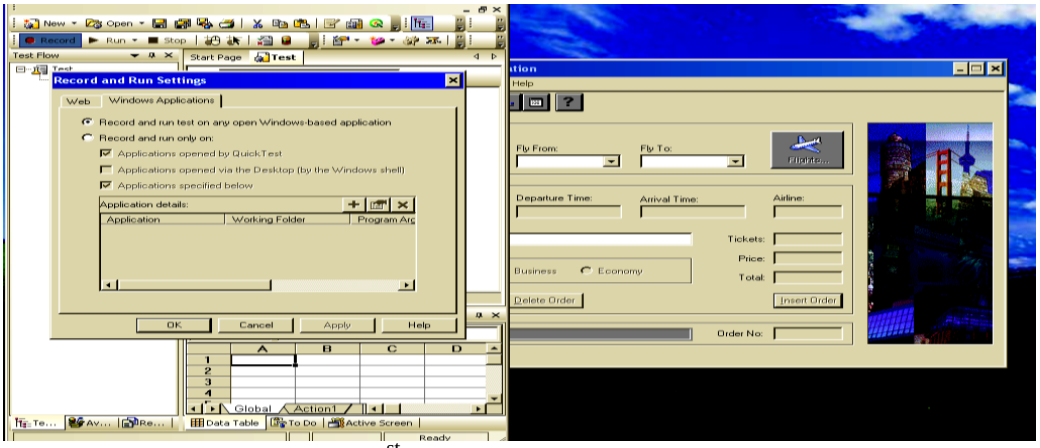
Results:



3. Insight Recording Mode

Steps:

10. In HP UFT, Click on Record button
11. Record and Run Setting dialog box opens, as shown below

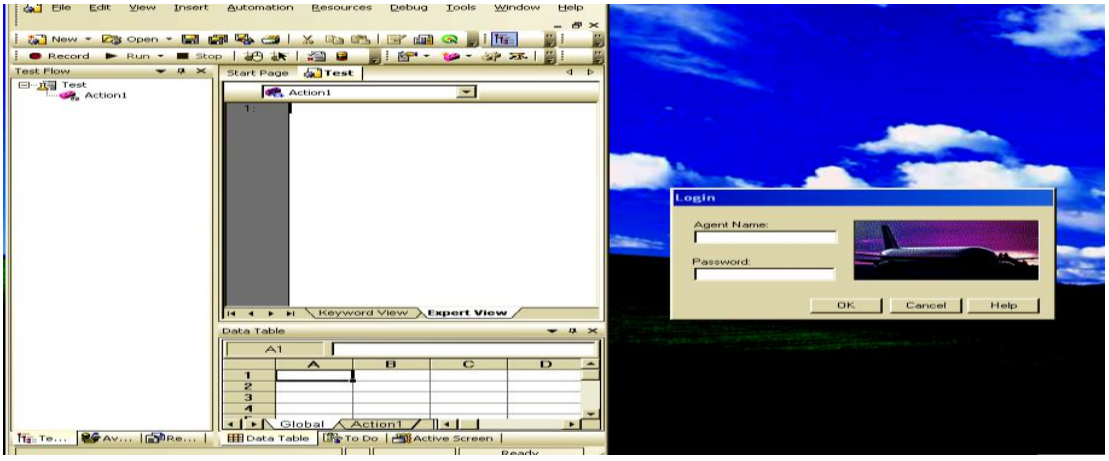


Select 1st option Click on Ok buttion

12. In HP UFT, select ->Record -> Insight recording mode
13. Open Flight Reservation window as below

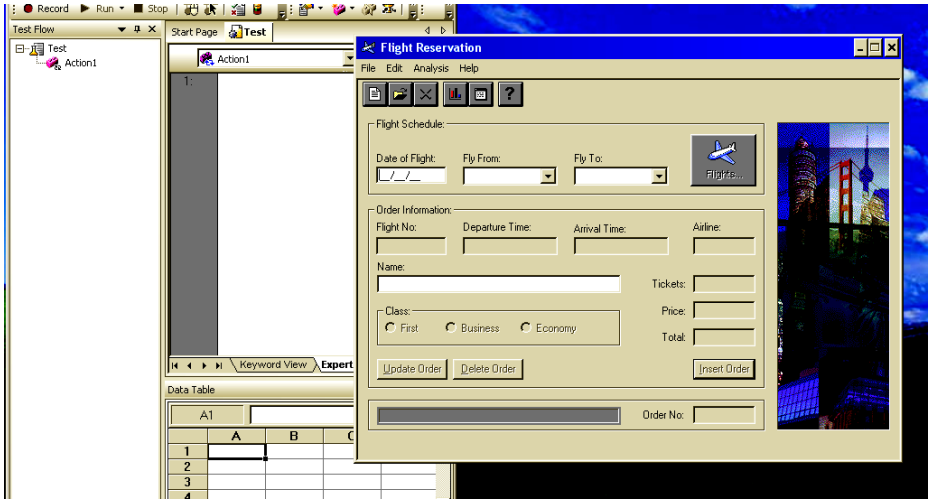
    Start -> Programs ->HP Unified Functional Testing -> Sample Application -> Flight GUI

14. Login dialog box opens as shown below

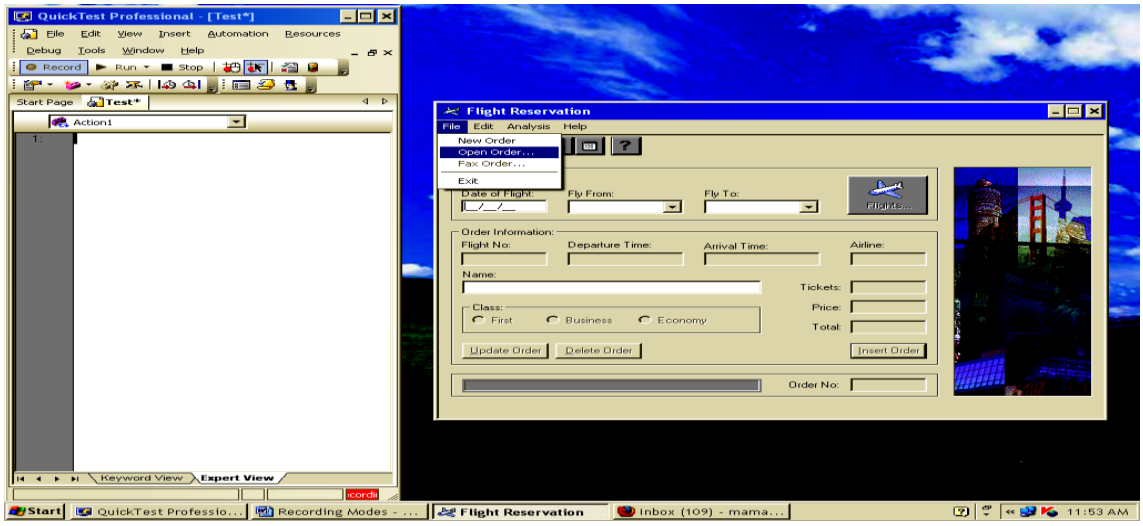


Give Agent name as: mercury or Your name or Your rollno, Password as: mercury, Click on OK

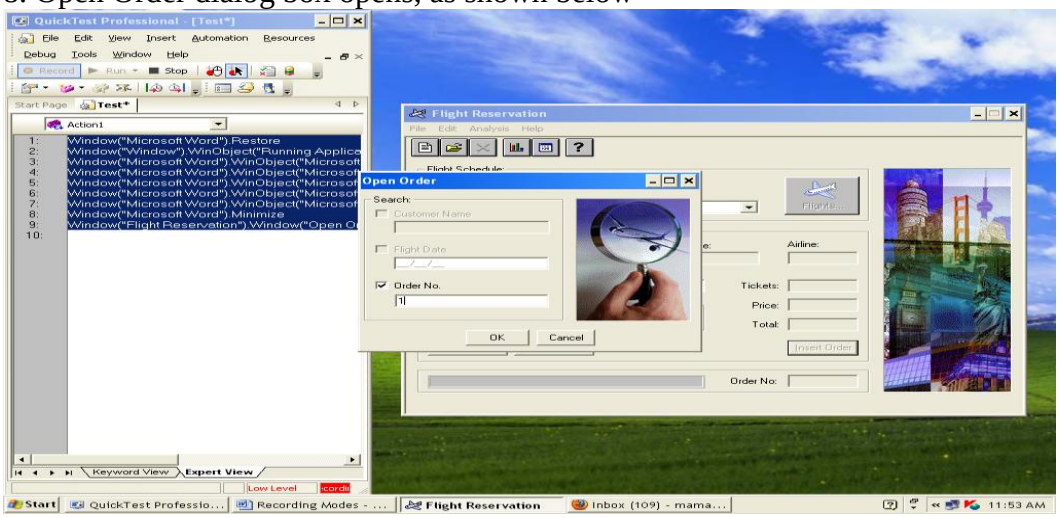
15. Flight Reservation dialog box opens, as shown below



16. Then in flight Reservation dialog box,Select file -> open order



8. Open Order dialog box opens, as shown below



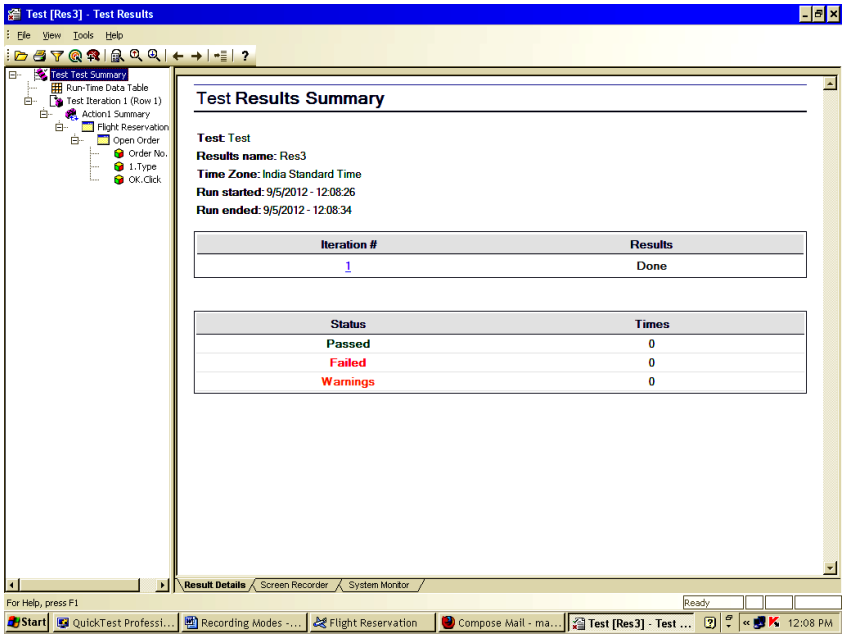
Click on Order No, Give order No : 1, Click on OK button

17. In HP UFT stop recording, you get script like this

```
1  Dialog("Login").InsightObject(          ).Click
2  Window("Login").WinObject("Text").Type "hello"
3  Window("Login").WinObject("Text").Type  micTab
4  Window("Login").WinObject("Text_2").Type "meru"
5  Window("Login").WinObject("Text_2").Type micCtrlDwn + "a" + micCtrlUp
6  Window("Login").WinObject("Text_2").Type "mercury"
7
   Dialog("Login").InsightObject(  OK  ).Click
8
   Dialog("Login").InsightObject(  OK  ).Click
9  Window("Login").WinObject("Text_2").Type "mercury"
10 Dialog("#32770").InsightObject(          ).Click
11
   Window("Flight Reservation").InsightObject(    ).DblClick
12
   Window("Flight Reservation").InsightObject( File ).Click
13
   Window("C:\Users\Mirmo\Documents\Unifi").InsightObject(      ).Click
```

18. stop recording

Results:

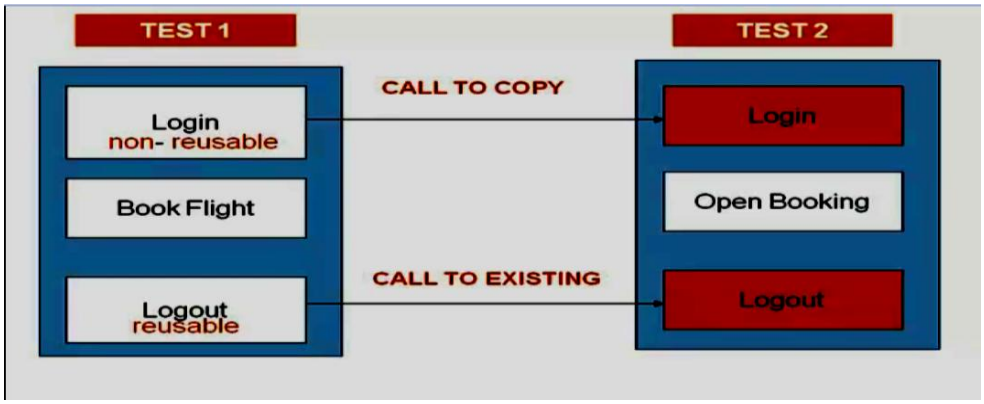


Program No.: 7.2

AIM: Performing operations on Actions:

- Creating Actions
- Splliting Actions
- Renaming Actions
- Deleting Actions
- Call to copy an action
- Calling an existing action
- Making an action Reusable and Non –Reusable

To perform the above operation, we are considering a sample example as below.



THEORY:

Action: A set of statements for performing a task or tasks

- Actions help to divide your test into logical units or Business processes
- Actions help to create a script which is more modular and efficient
- An action consists of its own test script
- When a script is newly created it consists of only one action
- Action consists of one or more actions

We divide our test into actions to streamline the process of testing.

Types of Actions:

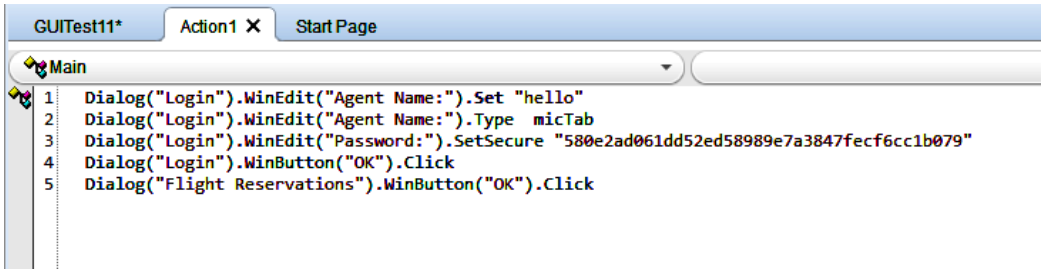
1. Reusable actions:
   - Can be used in other tests
   - Can be used in same test, multiple times
2. Non –reusable actions:
   - Cannt be used in other tests
   - Can be called in same test, only once.

Two methods to import actions:

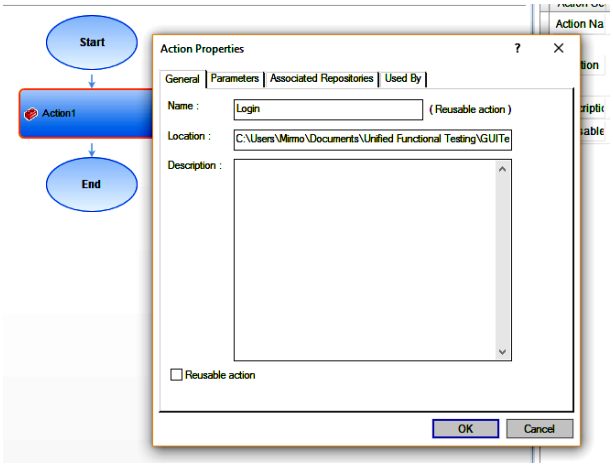
- **Call to copy of an action:**When you make a copy of an action, the action is copied entirely, including checkpoints, parameterization and the corresponding action tab in the data table into the calling test
- When you insert a copy of an existing action, you can make changes to the copied action, and your changes will not affect nor be affected by any other test
- Can insert copies of both reusable and non-reusable actions
- **Call to an existing action:** calls to actions are read only in the calling test. they can only be modified in the test in which they were created
- Enables you to use the same action in several tests and makes it easy to maintain tests
- Can make calls to only "reusable "actions

Steps:

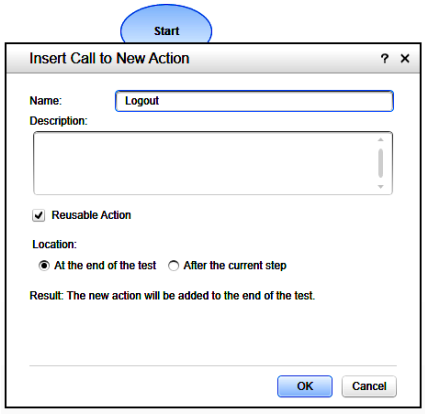
1. Open New Test ➔ GUI Test.
2. Goto Action1 from the flow chart.
3. Record Logging into Flight GUI



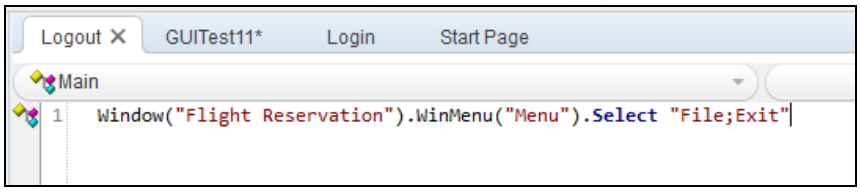
4. Right Click on Action1 ➔ Action Properties. There give name(login) to the action and deselect "Resuable" checkbox to make it non reuseable



5. Now Design ➔ Call to New Action. Give it a name (logout) and let it be reuseable and insert at the end of the test.

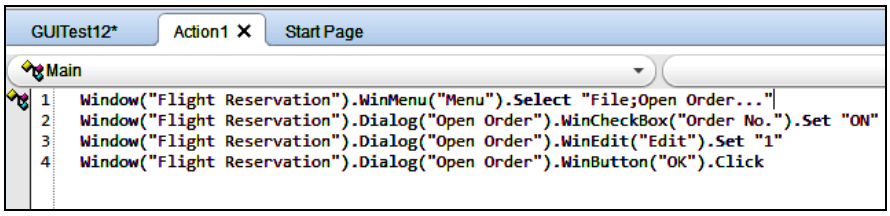


6. Now record the logout.

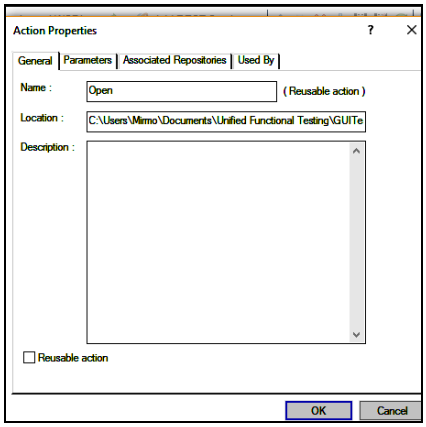


7. Save the test and open new test.

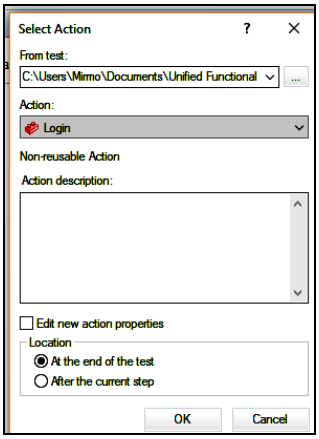
8. Now before recording keep Flight GUI open and ready and only record opening an order.



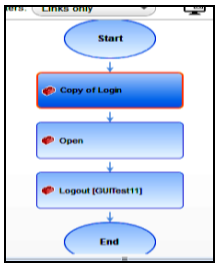
9. Now go to the flowgraph. Right click on Action1 and give name open and make it non – reuseable.



10. Go to design ➔ Call to Copy Action (since non resueable) and import Login action from the previous test and insert action at the end of the test.

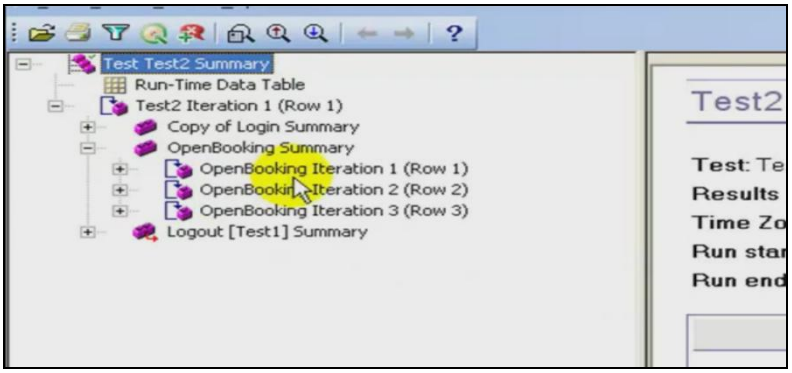


11. Go to design ➔ Call to Existing Action (since reuseable) and import Logout action from previous test and insert action at the end of the test.
12. Now put the actions in proper order of Login ➔ Open Order ➔ Logout and Run the Script.



13. Now we parameterize the open order action by going to the keyword view. (add to local data sheet)
14. Now if we run the script, it will iterate only once, because Open Order, is a non reuseable action.
15. To iterate through all rows of the data sheet, right click on Open Action ➔ Action Call Properties and select "Iterate through all rows"
16. Now it will iterate all the rows after logging in and then logout. Run the script and see the results.

Result:



Program: 7.3

AIM: **Parameterization of tests**

TEST SCENARIO	TEST STEPS	TEST DATA
1.Inputting a combination of valid alphanumeric agent name and password.	1. Open flight reservation application 2. enter valid Agent name 3. enter valid password 4. press ok 5. close application after succesffull.	Agent Name: mercury Password: mercury Agent Name: mercury123 Password: Mercury Agent Name:1234 Password: MERCURY

THEORY:

When you test your applications, you may want to check how theapplication performs the same operations with multiple sets of data.

Parameterization is:

- ✓ For Checkpints
- ✓ In object repository
- ✓ In action steps

Why Parameterization:

- ✓ Parameterization allows us to pick different values at run time
- ✓ Reduces time and effort
- ✓ Usage of data drivers allow us to use the same data for various input boxes.

Data Table: Integrated spread sheet (EXCEL), used for Data Driven Testing .

Data Driven Testing: Testing the same task with multiple sets of test data. Data can be imported to the data table from external files (flat files, excel sheets etc) and data bases (MS-Access, SQL server, oracle etc)

Two types of Sheets in Data Table:

Global Data Sheet:

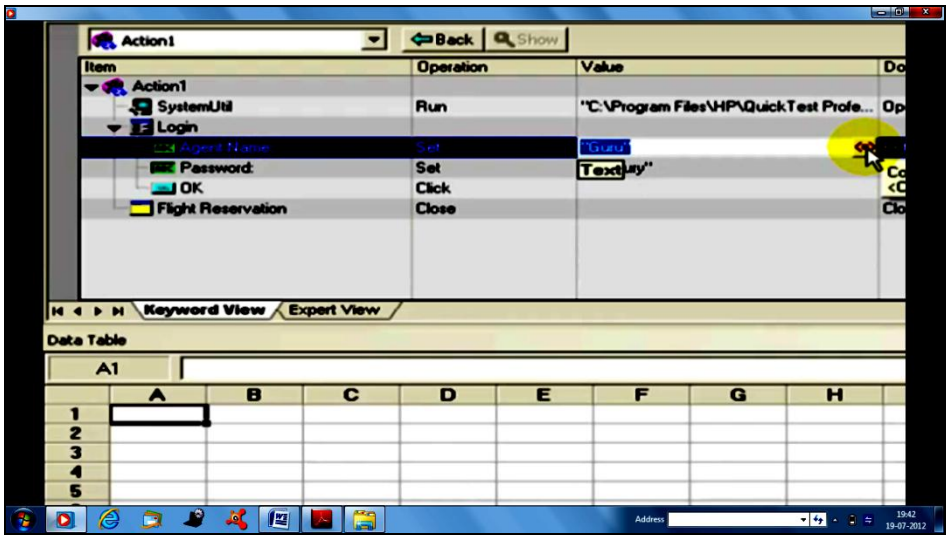
- used for entire test i.e used for all actions in the test.
- Sheet is named as "GLOBAL"
- Any action can access and write data into global sheet.
- Similar like global variables in languages like c

Local or Action Data sheet:

- For specific action only.
- Sheet is named as "ACTION NAME"
- An action can read and write data into its own local data sheet only
- Similar like local variables in languages like c

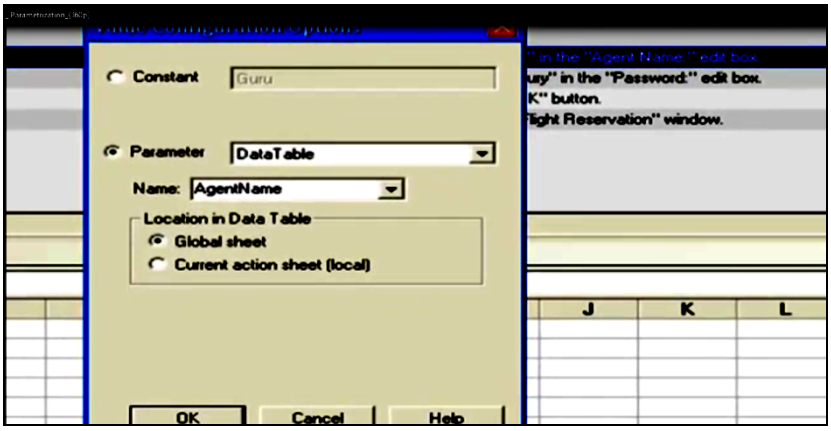
Steps:

1. Click on Record buttion
2. Start-> programs->HP UFT -> sample application-> flight reservation
3. Enter Agent name: mercury
   Password: mercury
   Click –ok button
4. Flight Reservation Window open, then close the window
5. In HP UFT- stop Recording
6. Goto keyword view



And parameterize the argument mercury. On clicking on Parameterization icon <<.>>

7. Then you get the Value configuration option dialog box appear.

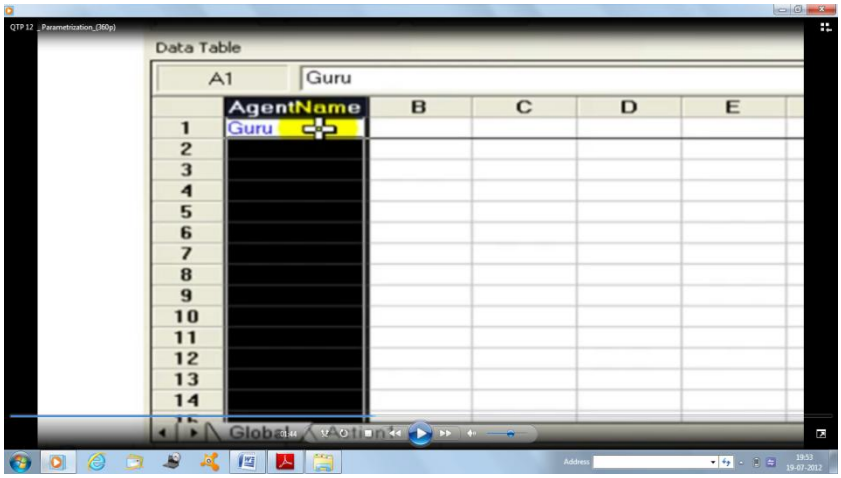


8. Click on parameter

   Name: AgentName

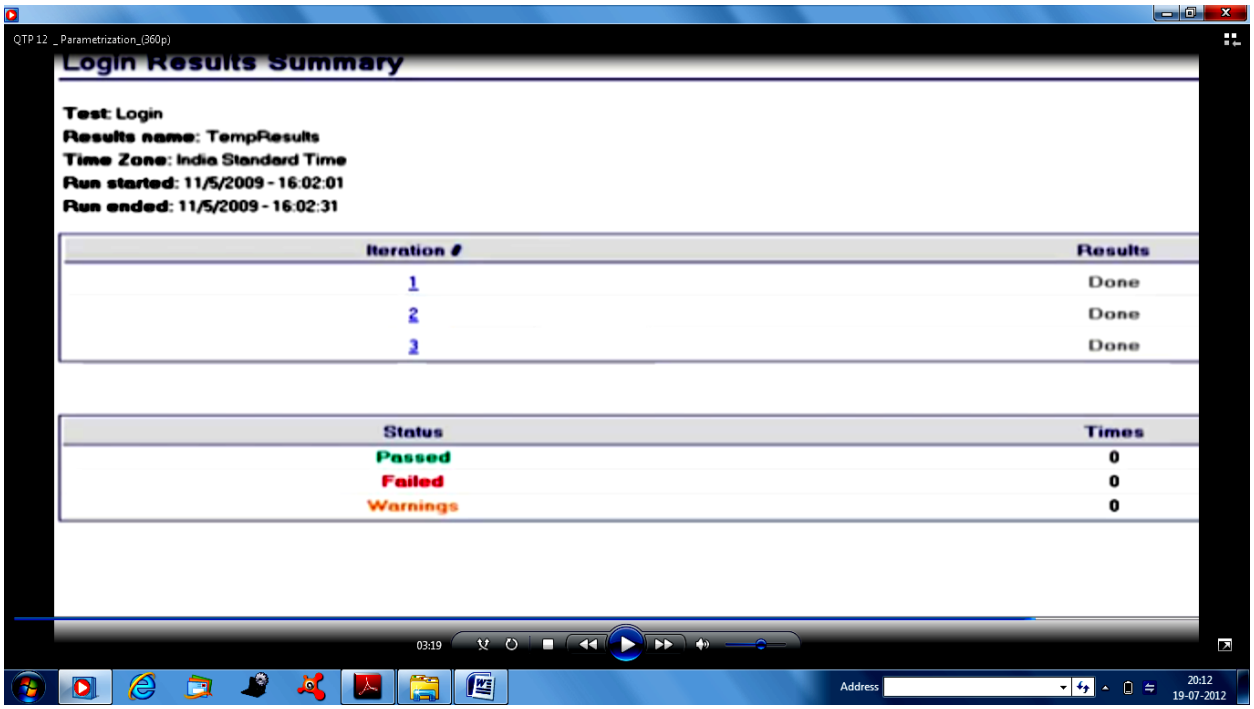
   Click ok

9. Then in Data table you get AgentNamecoloumn, like this



10. Like that you parameterize for password also.
11. Run the script
12. Analyze the results.

Result:



Program: 7.4

AIM: Checkpoints

THEORY:

Checkpoint Definition: A confirmation or verification point in which the value of some property which is expected at a particular step is compared with the actual value which is displayed in the application

Types of checkpoints:

The expected value can be any property of an image or web page or table with values displayed in application, portion of text or text displayed in a specific region

Dynamic value displayed in application from database, bitmap of image displayed or dynamic text displayed from XML

Based on expected value, checkpoints are classified as

1. Standard checkpoint
2. Page checkpoint
3. Text/Text area checkpoint
4. Image checkpoint
5. Data Base checkpoint

6. XML checkpoint
7. Bitmap checkpoint
8. Table checkpoint
9. Accessibility checkpoint

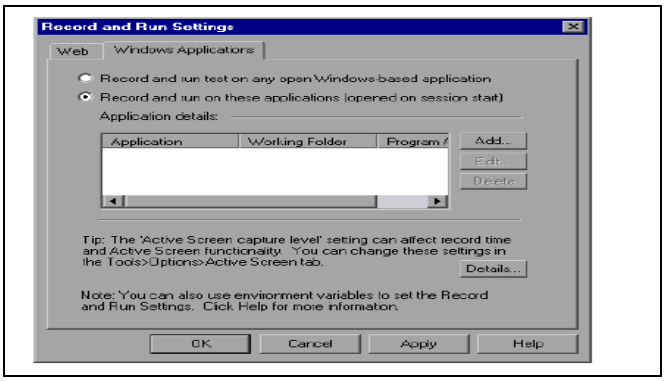
Page checkpoints:

Standard checkpoint created for a web page is called page checkpoint

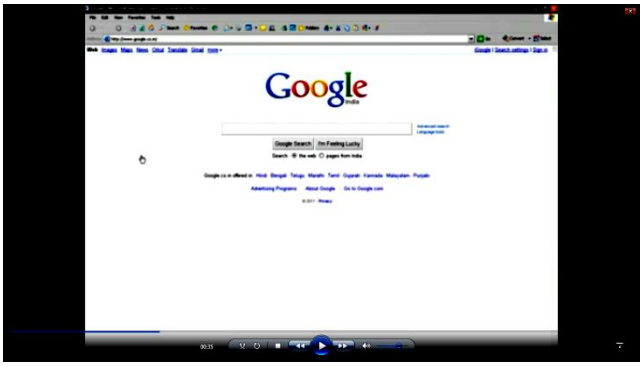
- It is used to check total no. of links and images on a web page
- Ex: suppose in your web page, you have six links andis used to check total no. of links and images on a web page
- Ex: suppose in your web page, you have six links and 4 images
  HP UFT results will give you
  No of links=6
  No of images=4
  - Page checkpoints can also be used to check load time, i.e time taken to load a web page
  - Also checks HTML source tags and broken lines
  - Very useful for Regression Testing

Steps:

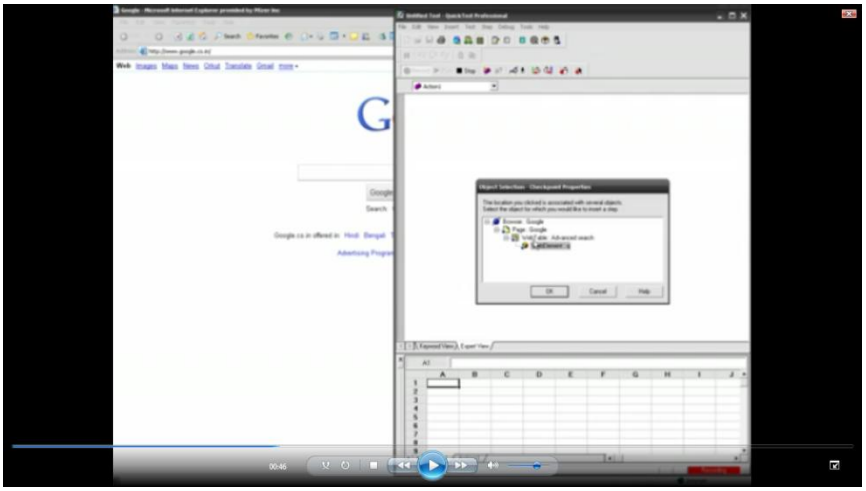
1. Click on Record
2. You get Record and Run Setting dialog box appear.



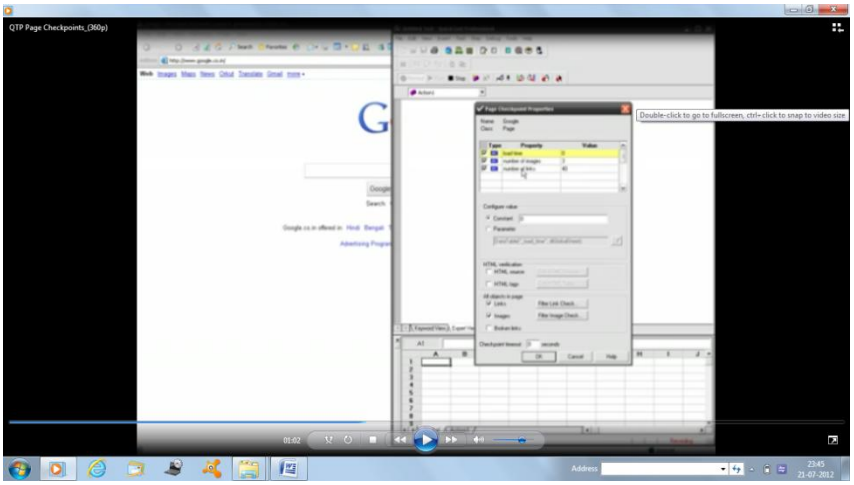
   - select web application
   - select 2nd option
   - Give URL: www.google.com
   - Open in: Internet explorer or firefox
   - click ok
3. Then Google site appears
4. In HP UFTselectDesign -> checkpoint -> standard checkpoint
5. Then cursor changed to hand icon, so click on google page as below



6. In HP UFT you get, checkpoint properties dialog box open. Click on page: google

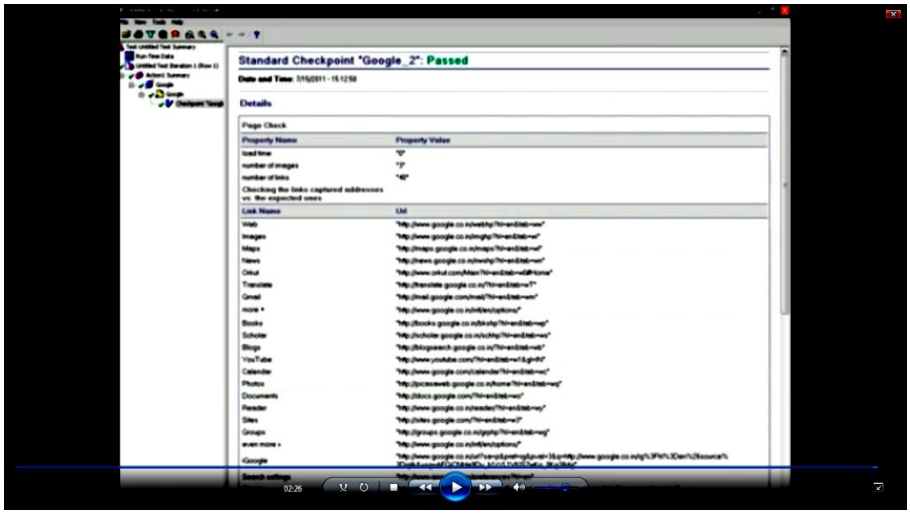


7. Page checkpoint properties dialog boxopens



8. Click on ok
9. Run the script and analyze the results

Results:



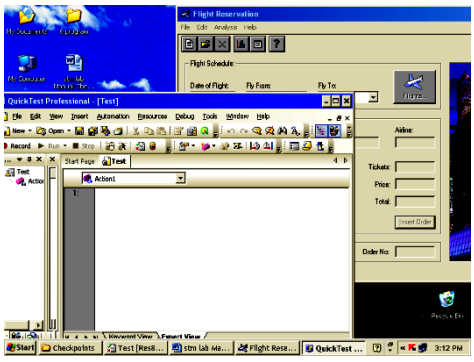
Bit mapCheckpoint:

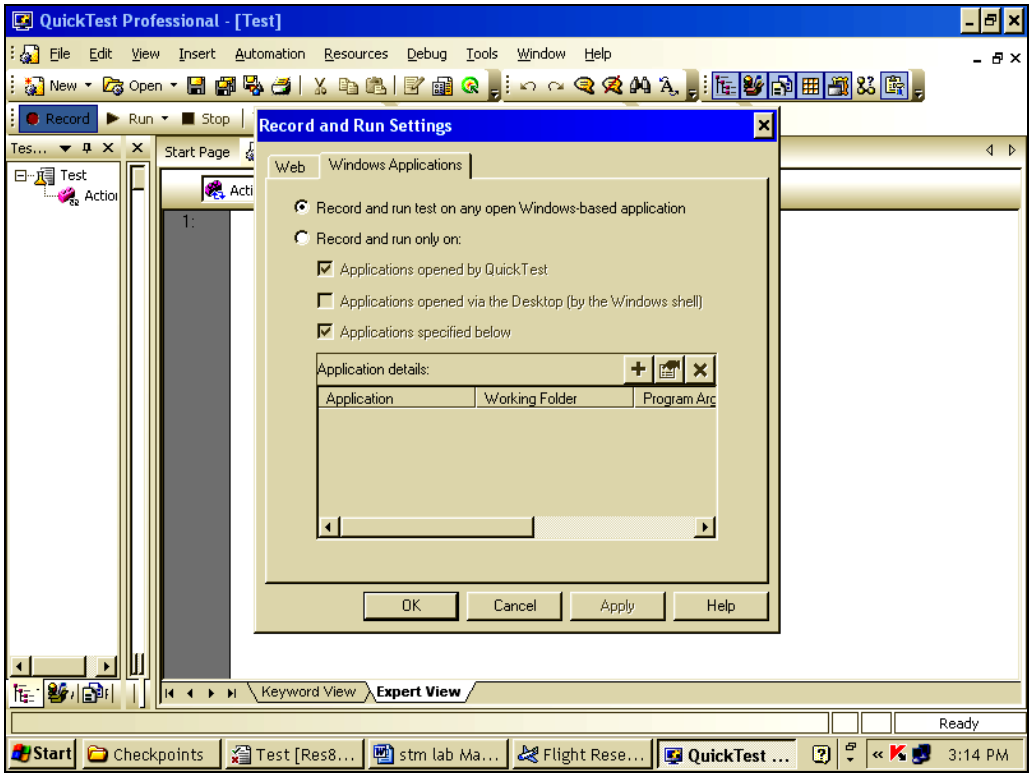
A bitmap checkpoint help a user in checking the bitmap of an image or a full web page

- During recording, you can insert a bitmap checkpoint, HP UFT will store it as expected value
- During Run time, HP UFT will compare the actual bitmap with the expected bitmap and give the appropriate results

Steps:

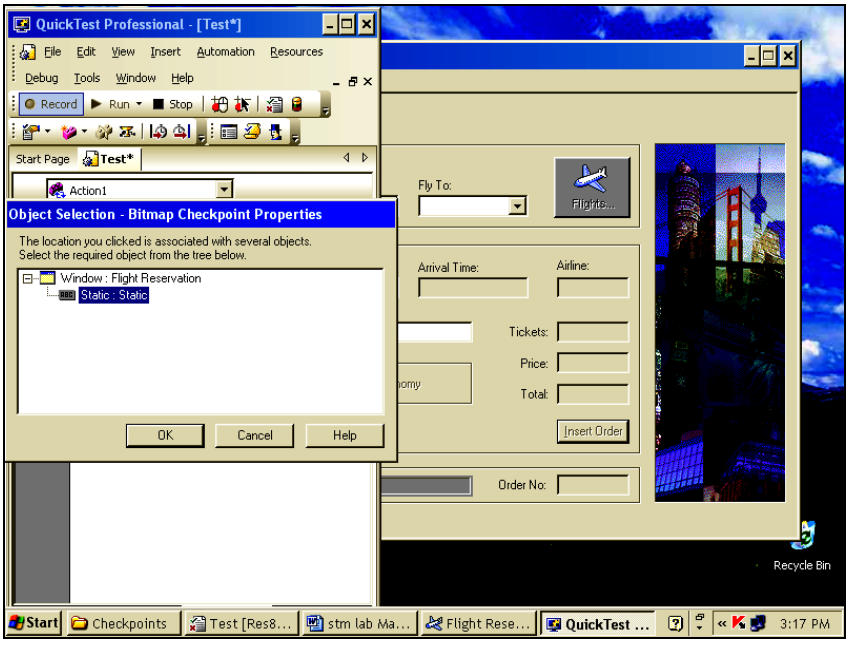
1. Open the flight reservation window
2. Click on record





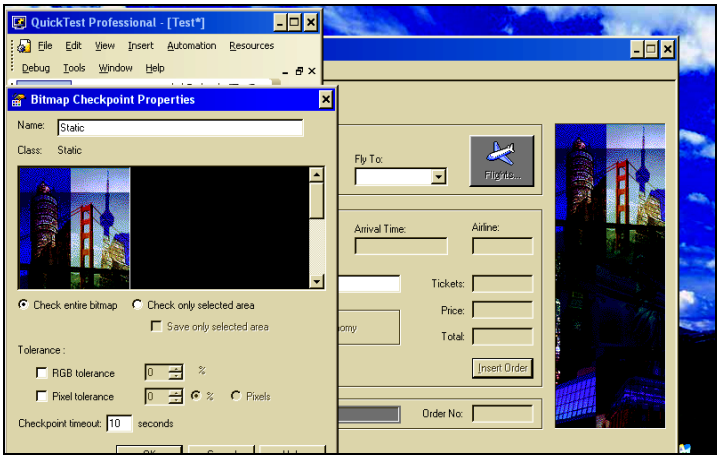
3. design -> checkpoint -> bitmap checkpoint. cursor changed to hand icon so click on flight reservation window image

4.then object selection: bitmap checkpoint properties dialog box opens

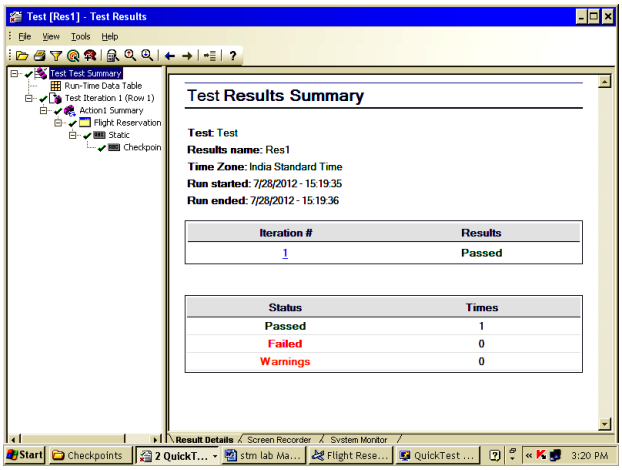


Click ok

7. bitmap checkpoint properties dialog box opens as shown below



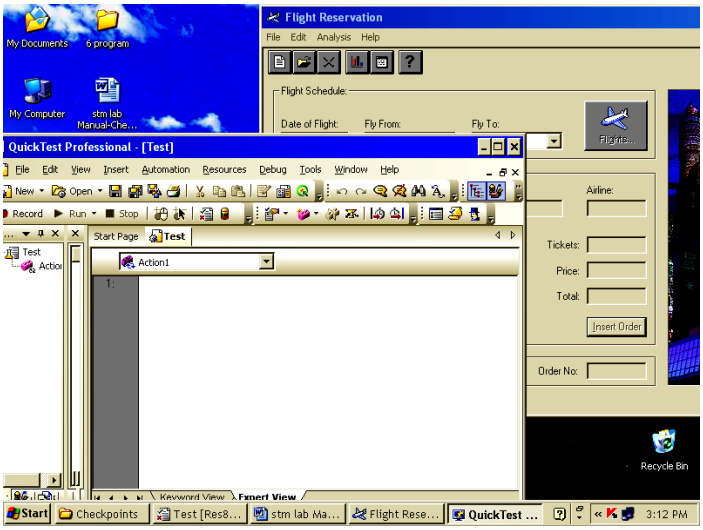
8. In HP UFT, stop recording

Results: 

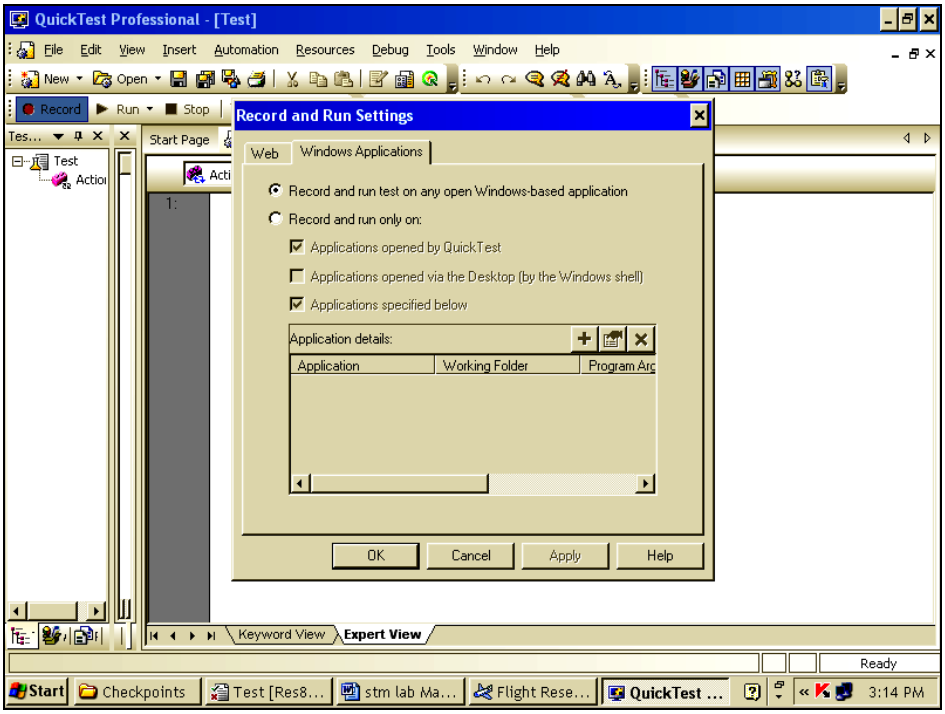
Bitmap checkpoint for fail

Steps:

4. Open the flight reservation window



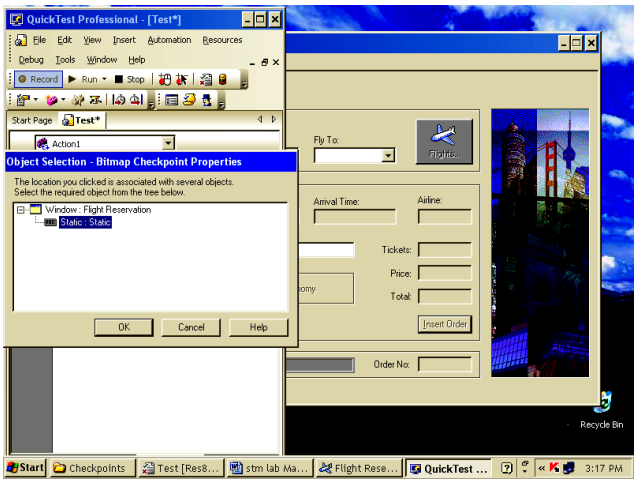
5. Click on record



- Select 1st option
- Click ok

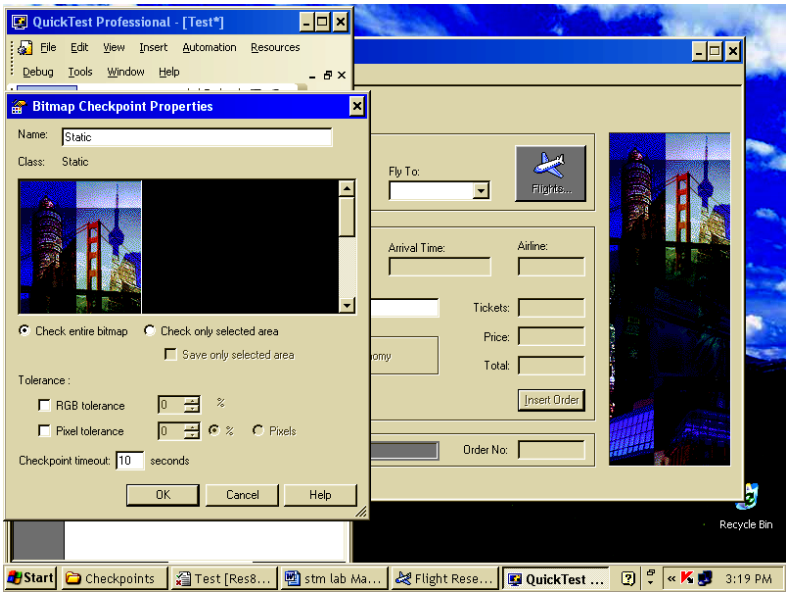
6. design -> checkpoint -> bitmap checkpoint
7. cursor changed to hand icon so click on flight reservation window image

4. then object selection: bitmap checkpoint properties dialog box opens



Click ok

7. bitmap checkpoint properties dialog box opens as shown below



8. In HP UFT, stop recording

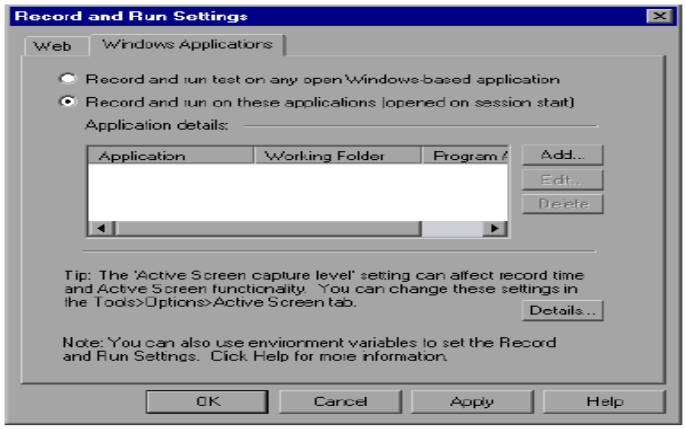
Text checkpoint for web application:

- Used to check expected text in a web page or application
- At run time, HP UFT verifies expected value with an actual value

EX: Open any site, select any text that you want to test.

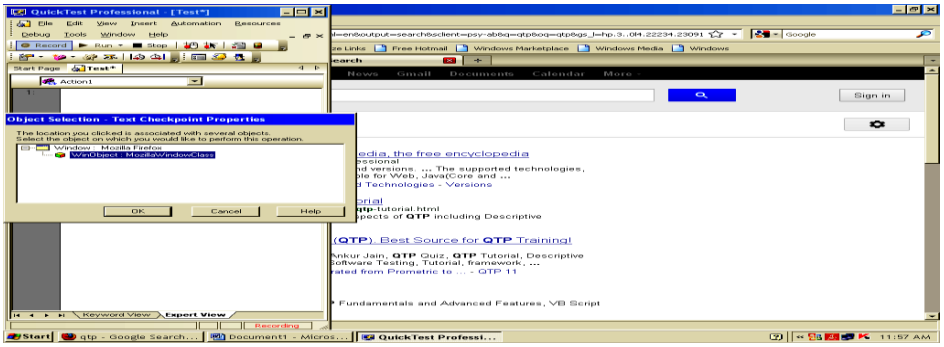
Steps:

1. Before recording, open any site like google, give any query like HP UFT
1. Click on Record
2. You get Record and Run Setting dialog box appear.

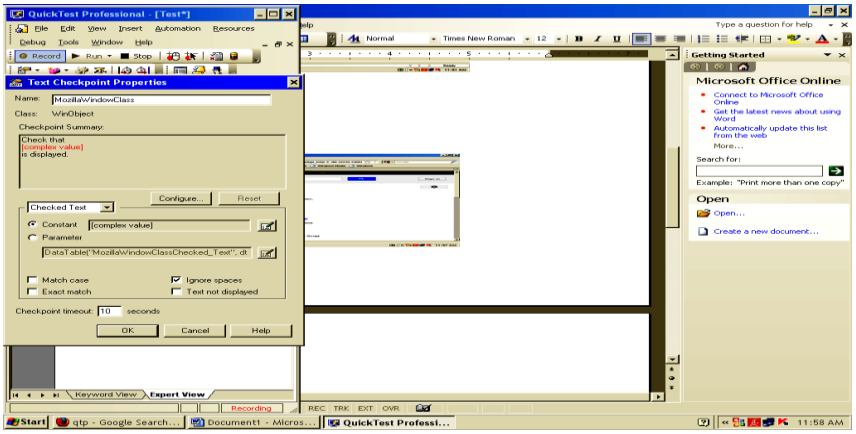


- ❖ select web application
- ❖ select 1st option
- ❖ click ok

3. In HP UFT select
   Design -> checkpoint -> Text Checkpoint
4. Cursor changed to hand icon, place cursor where you want to select the text
5. ThenObjectSelection-Text checkpoint properties dialog box opens



6. Text checkpoint properties dialog box opens



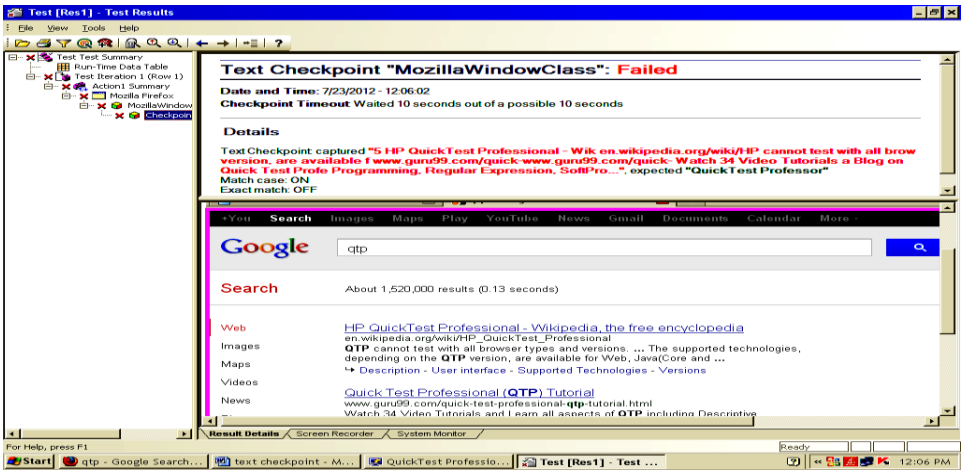
- ▪ On summary you get the selected text, so

- Click on Configure buttion
- Select checked text so your selected text is hightlighted
- Click on Match case
- Click on ok

7.In HP UFT stop recording

8. Run the script

Results:

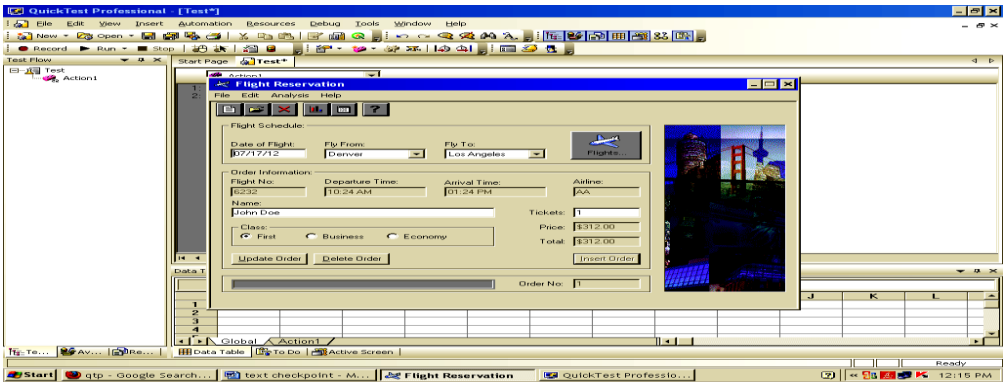


Text checkpoint for window application:

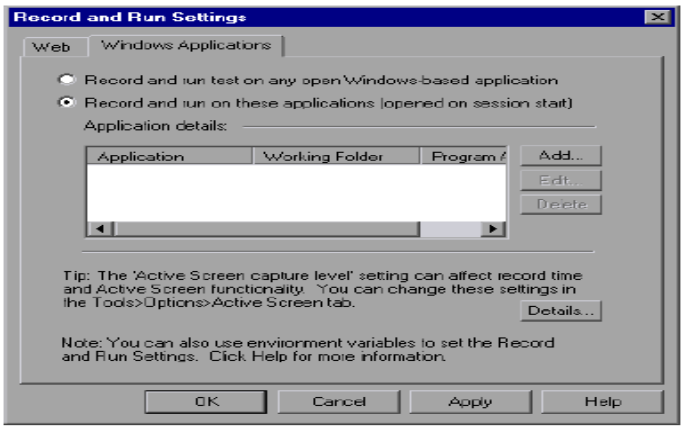
checks that a text string is displayed in the appropriate place on a Web page or application

Steps:

1. Before recording open flight reservation and open an order as shown below



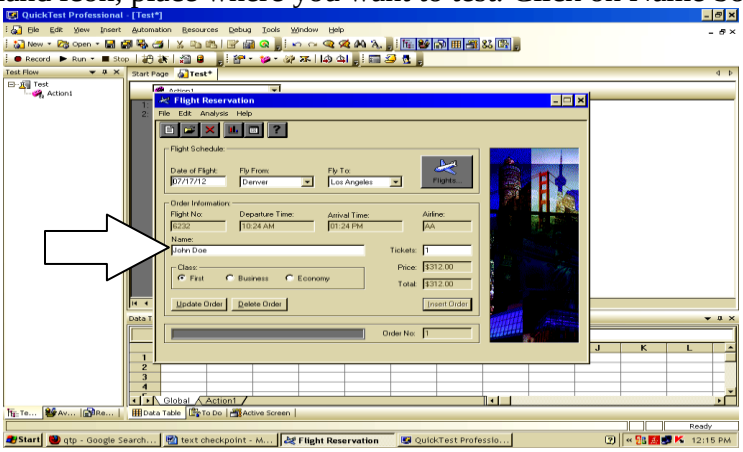
2. click on Record button
3. Record and Run Setting Dialog box open



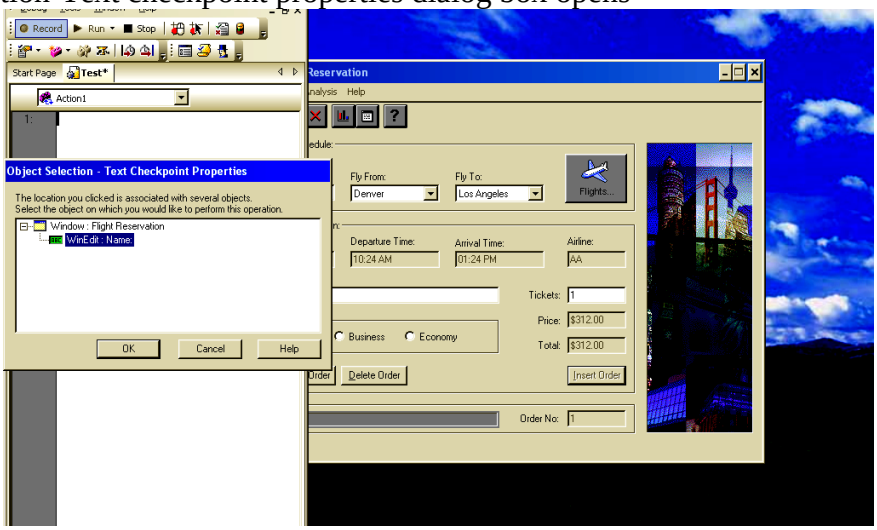
- Select window applicaion
- 1st option
- Click ok

4. design -> checkpoint -> text checkpoint

Cursor changed to hand icon, place where you want to test. Click on Name box



5.Object Selection-Text checkpoint properties dialog box opens

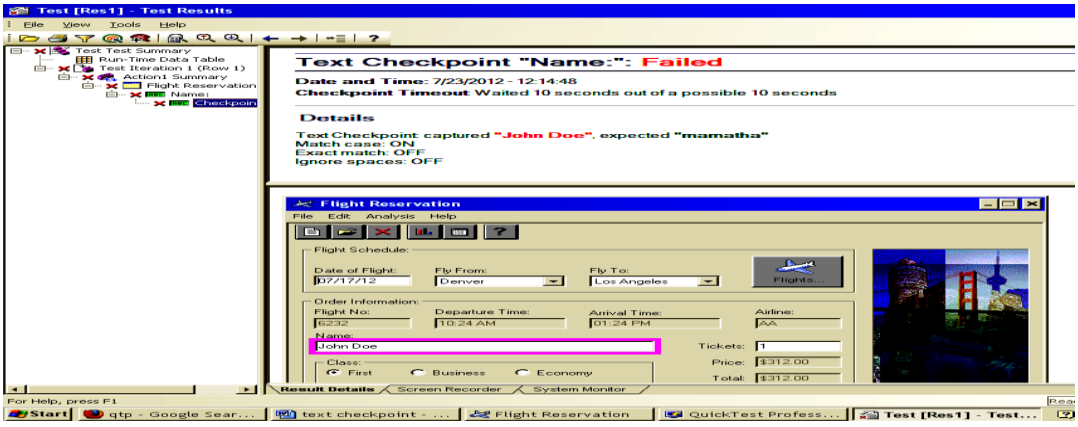


Click ok

6. Text checkpoint properties dialog box opens

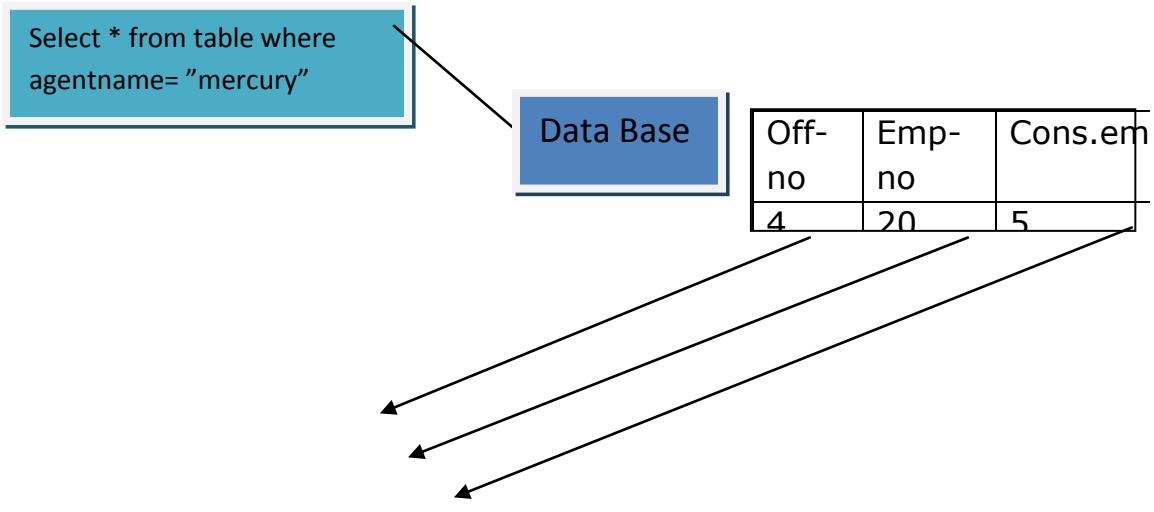
- Constant – mamatha
- Select match case
- Click ok

7. In HP UFT, you get script like this
Window("Flight Reservation").WinEdit("Name:").Check CheckPoint("Name:")
8. stop recording
9. Run the script
Result:



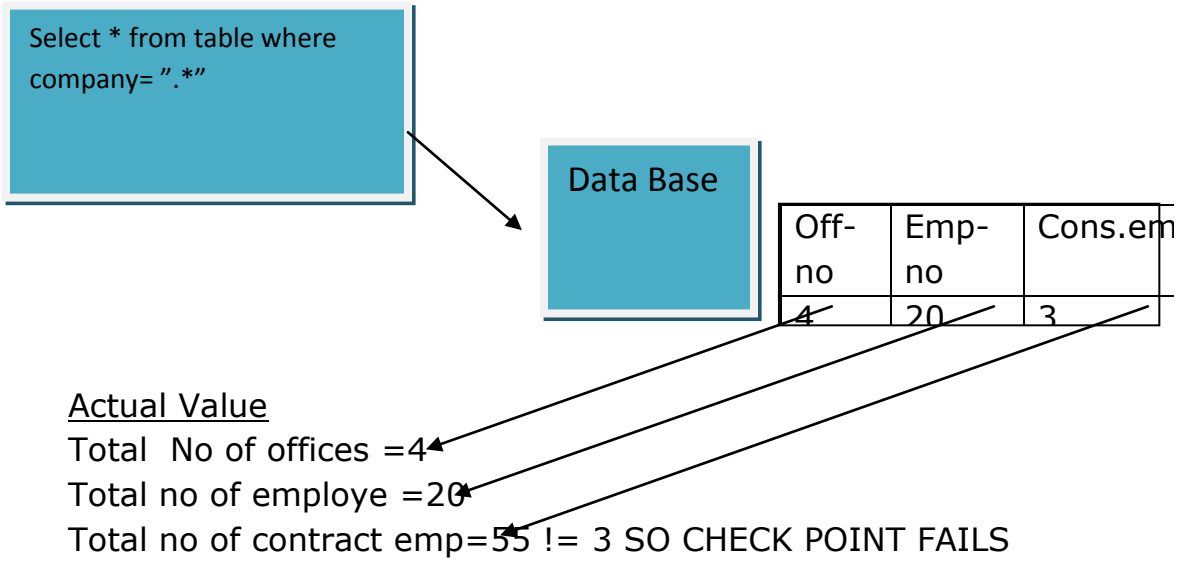
DataBase Checkpoint:

- ✓ A query is created during record time
- ✓ Query can be created by data base query wizard.
- ✓ During record time, query is created and send to data base and stored as expected.



Stored as Expected Value
Total No of offices =4
Total no of employe =20
Total no of contract emp=5

- During run time, same query is send to data base and data base value are fetched and compared with expected

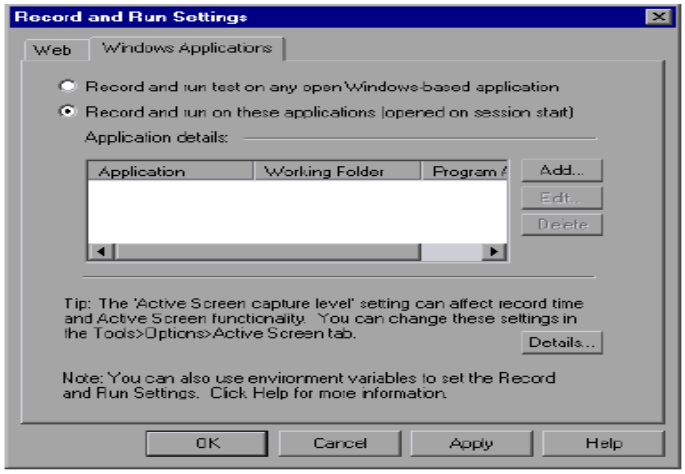


Actual Value
Total No of offices =4
Total no of employe =20
Total no of contract emp=55 != 3 SO CHECK POINT FAILS

EXAMPLE: Updating order by changing customer_name from paul to mamtha

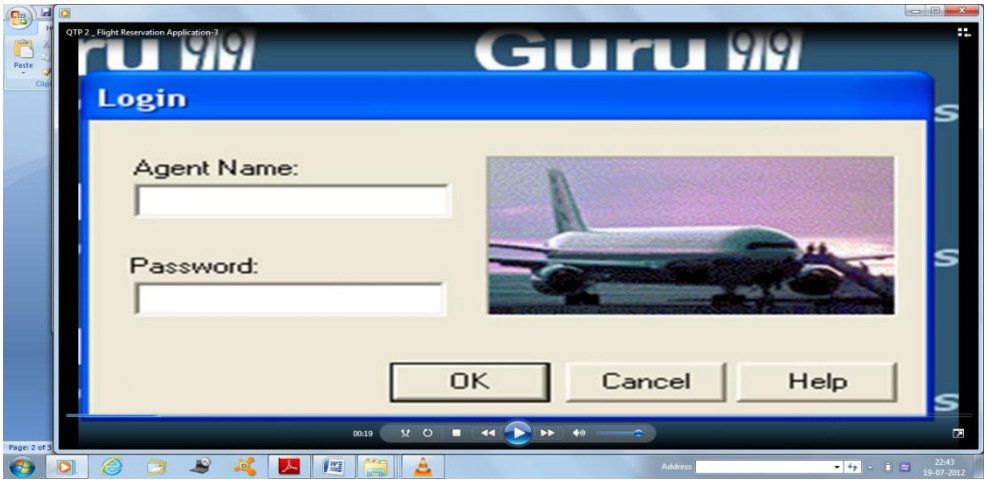
Steps:

1. Click on Record
2. You get Record and Run Setting dialog box appear.

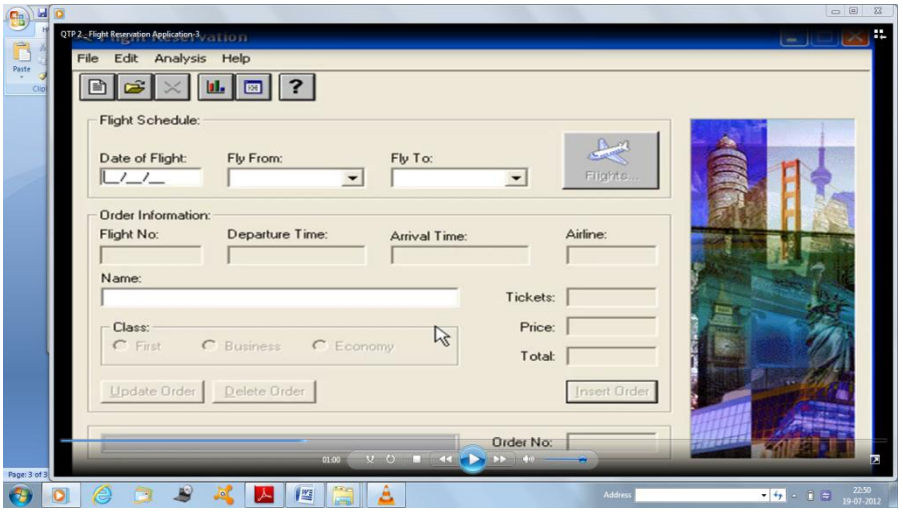


- ❖ select window application
- ❖ select 2nd option
- ❖ click ok

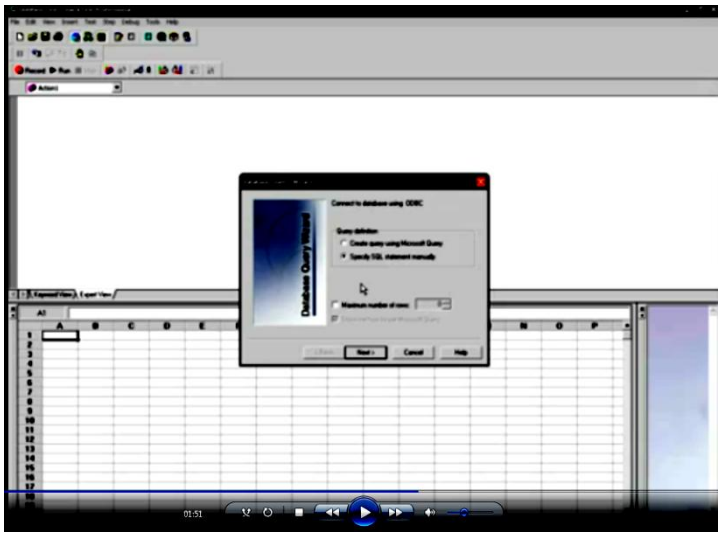
3. Login dialog box appears
   Agent Name: mercury
   Password: mercury
   Click on ok



4. Then Flight Reservation window opens,then insert new order, and minimize the dialog box.

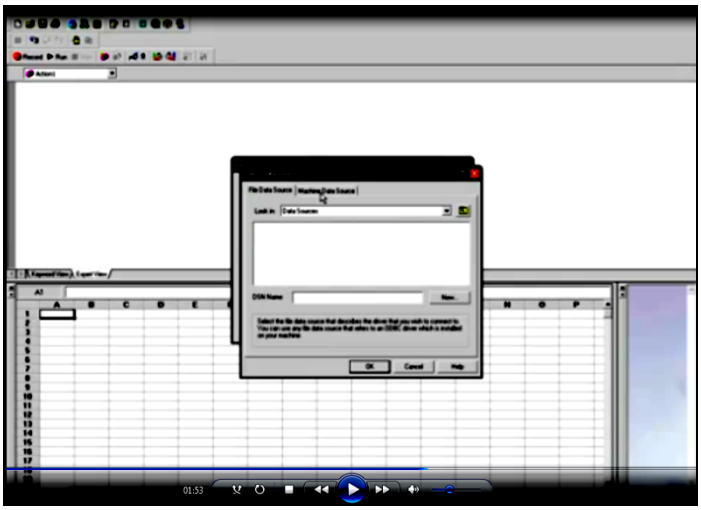


5. In HP UFT, select Design -> checkpoint -> database checkpoint
6. Data Base Query wizard dialog box opens as shown below



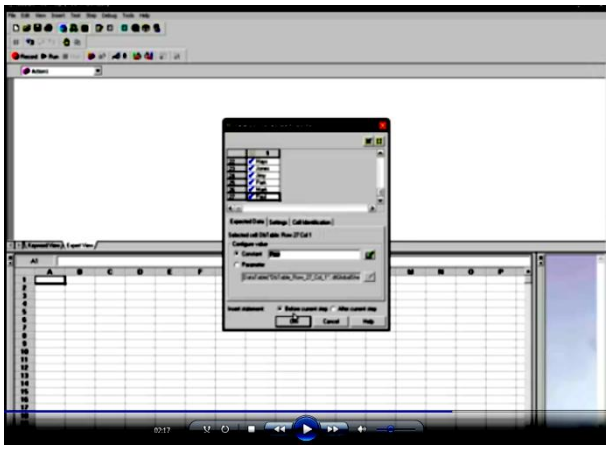
Choose specify SQL statement manually Click on Next

7. Data base query wizard dialog box opens

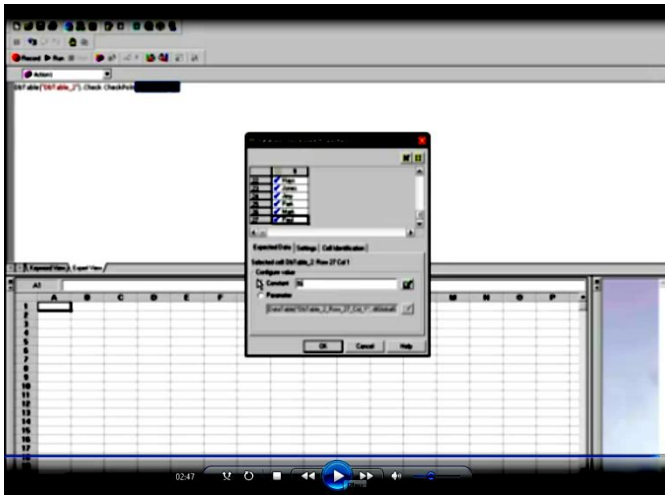


- Click on CREATE buttion
- Select machine data source
- Select QT_flight _32
- Click on OK

8. Specify the SQL Statement
   Select customer_name from orders
   Click on finish buttion

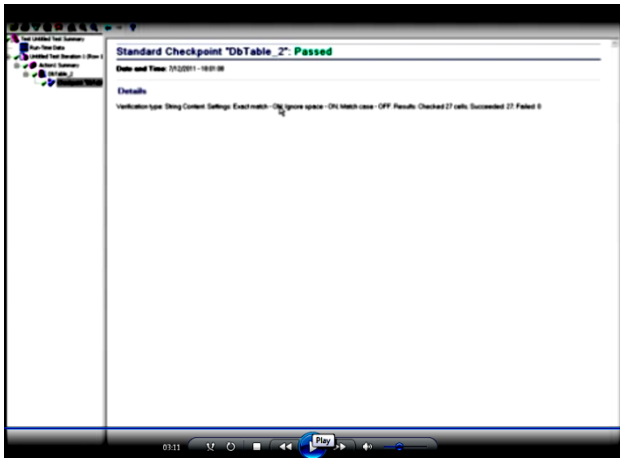


9. Data base checkpoint properties dialog box opens
   - Drag down and hight light on paul
   - On constant: paul highlight
   - Click ok
10. In HP UFT, you get the script
    DbTable("DbTable_2").checkpoint("DbTable_2")
    Highlight the Dbtable_2,and right click on it, select action properties and highlight on changed name



11. Open flight reservation
    On NAME: Mamatha, change the name as u desire
    Click on Update Order
12. Run the script

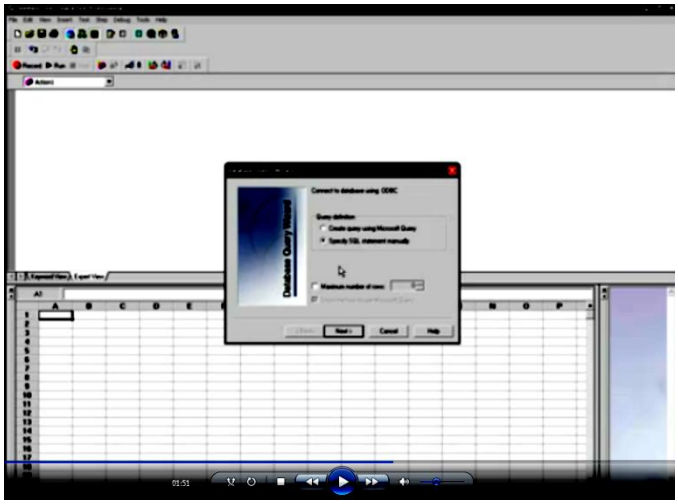
Result:



Dabase Checkpoint (fail):

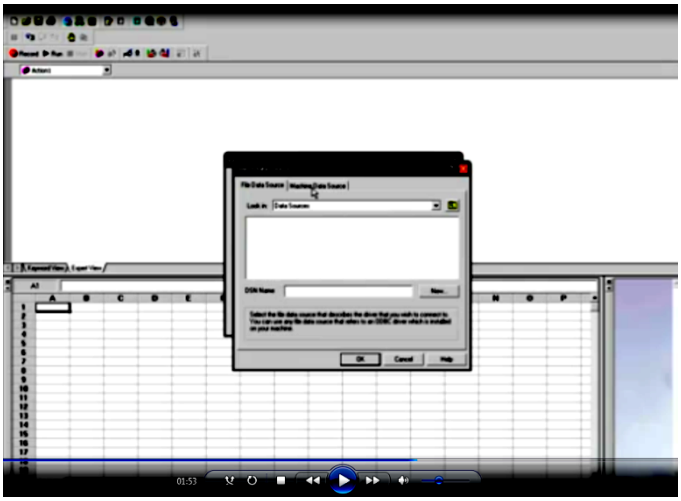
Steps:

1. In HP UFT, select
   Design -> checkpoint -> database checkpoint
2. Data Base Query wizard dialog box opens as shown below



Choose specify SQL statement manually, Click on Next

3. Data base query wizard dialog box opens



- Click on CREATE buttion
- Select machine data source
- Select QT_flight _32
- Click on OK

4. Specify the SQL Statement "Select * from orders", Click on finish buttion

5. Data base checkpoint properties dialog box opens

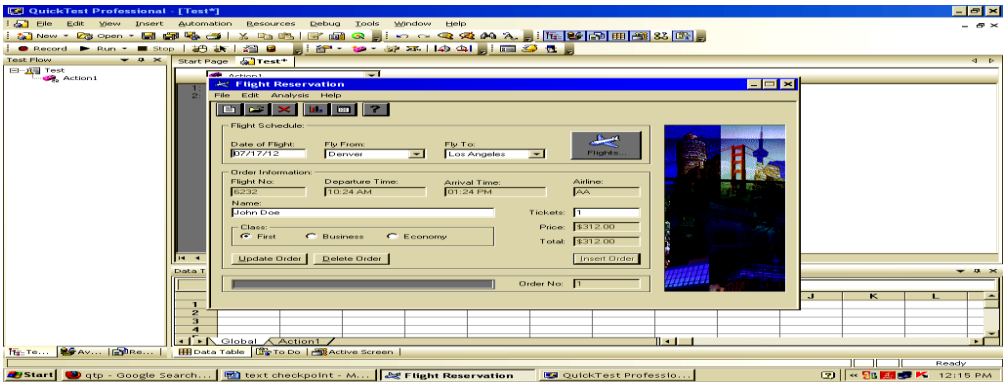
Change any of the fields in database, ex: change mamatha as mamta

6. In HP UFT, you get the script
   DbTable("DbTable_2").checkpoint("DbTable_2")
7. Run the script

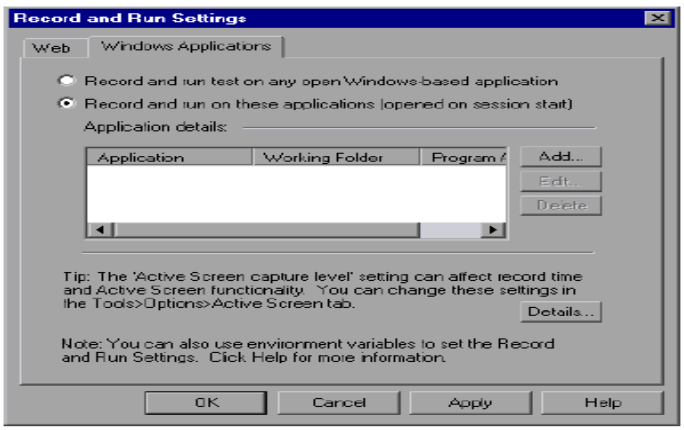
Image Checkpoint checks the value of an image in your application or Web page.

For example, you can check that a selected image's source file is correct.

1. Befor recording, open flightreservation and open an order as shown below.



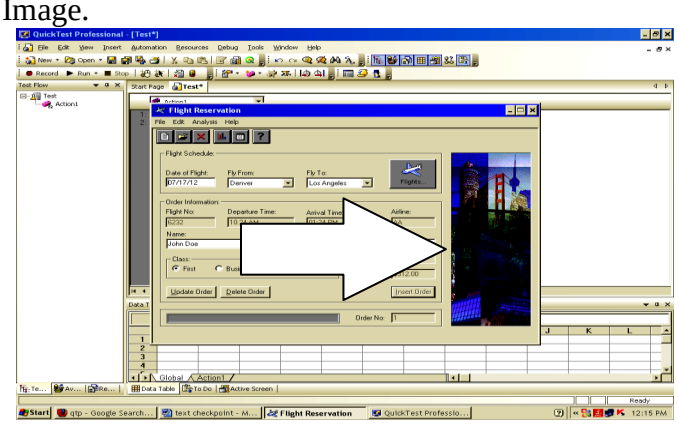
2. click on Record button
3. Record and Run Setting Dialog box open



- Select window applicaion

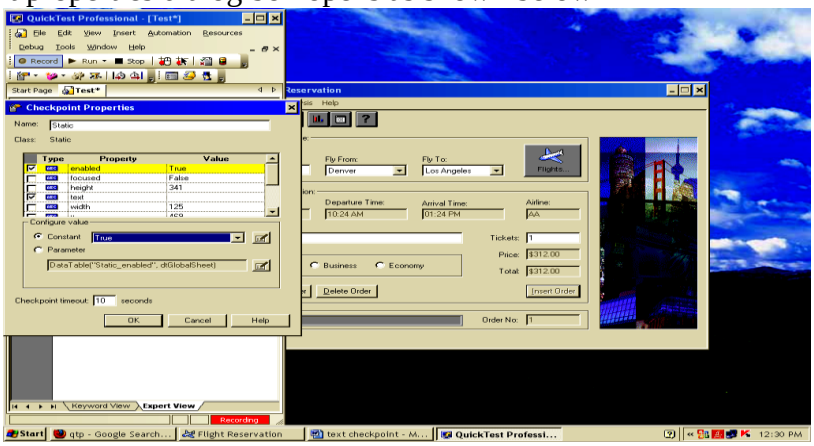
- 1st option
- Click ok

4. design -> checkpoint -> standard checkpoint. Cursor changed to hand icon, place where you want to test. Click on Image.



5.Object Selection-Text checkpoint properties dialog box opens. Click on ok

6. then checkpoint properties dialog box opens as shown below



- Hight light the enabled true value
- On Constant: false
- Click ok

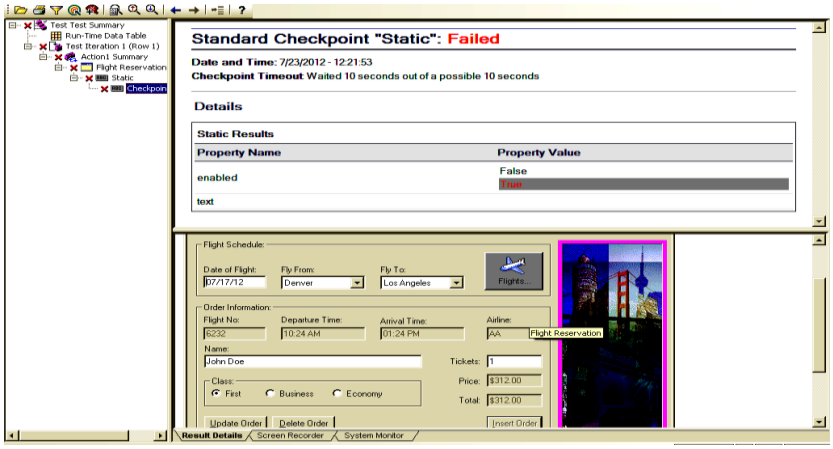
7. In HP UFT, you get the script like below

Window("Flight Reservation").Static("Static").Check CheckPoint("Static")

8. stop recording

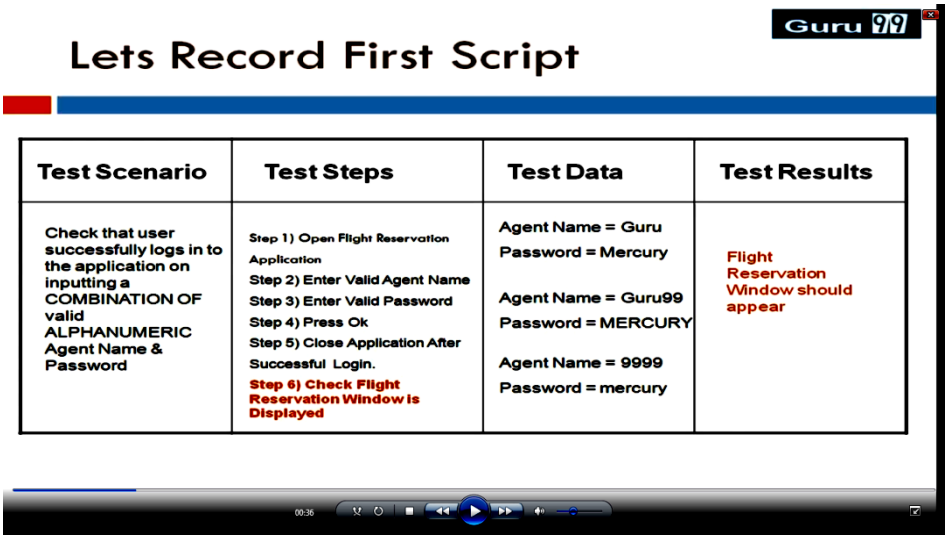
8. Run the script

Results:



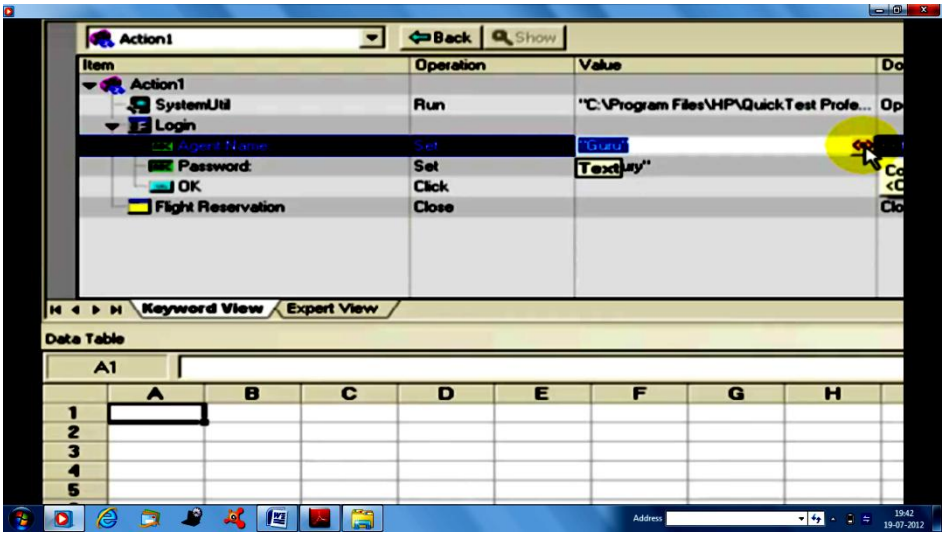
Standard Checkpoint:

- checks the property value of an object in your application or Web page.
- The standard checkpoint checks a variety of objects such as buttons, radio buttons, combo boxes, lists, and so forth.
- For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box.
  For example:



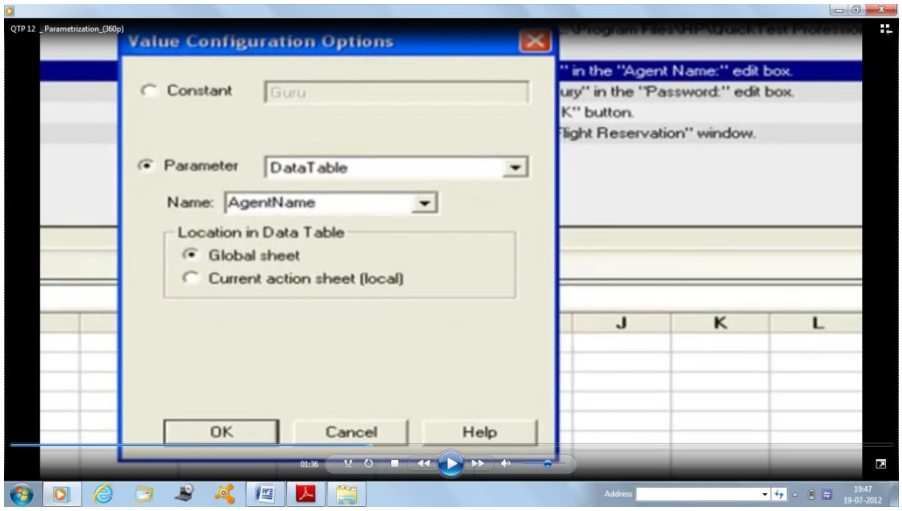
Steps:

13. 1. Click on Record buttion
14. Start-> programs->HP UFT -> sample application-> flight reservation
15. Enter Agent name:mercury
    Password: mercury
    Click –ok button
16. Flight Reservation Window open, then close the window
17. In HP UFT- stop Recording
18. Goto keyword view

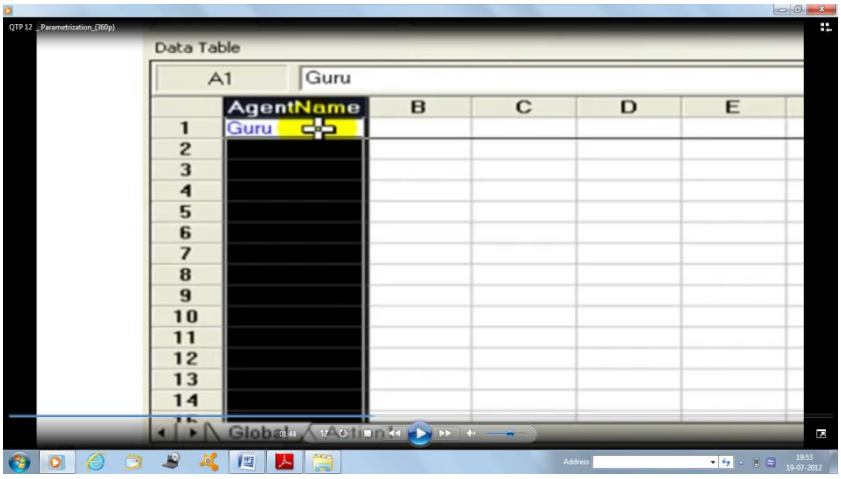


And parameterize the argument mercury. On clicking on Parameterization icon <<.>>

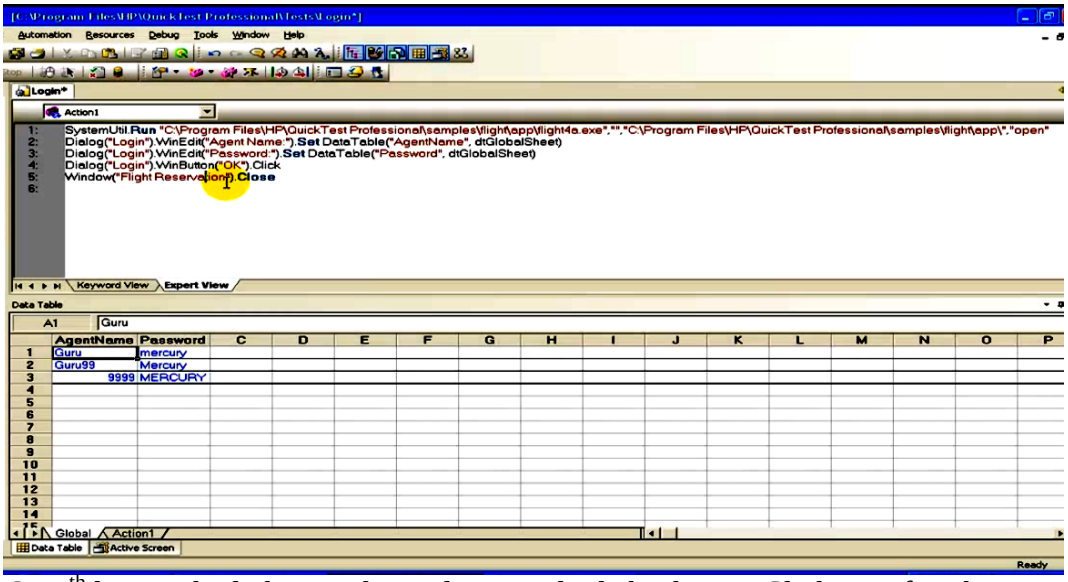
19. Then you get the Value configuration option dialog box appear.



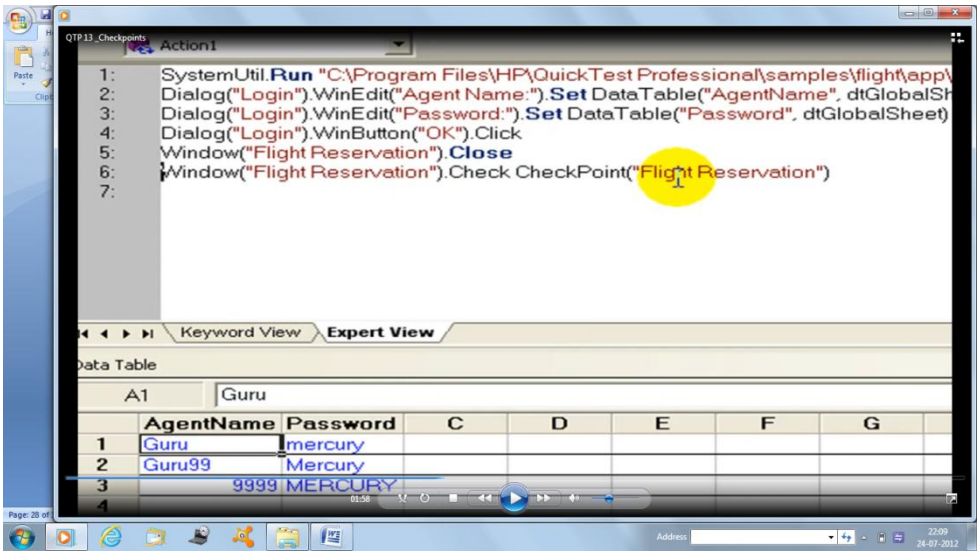
20. Click on parameter, Name: AgentName, Click ok
21. Then in Data table you get AgentNamecoloumn, like this



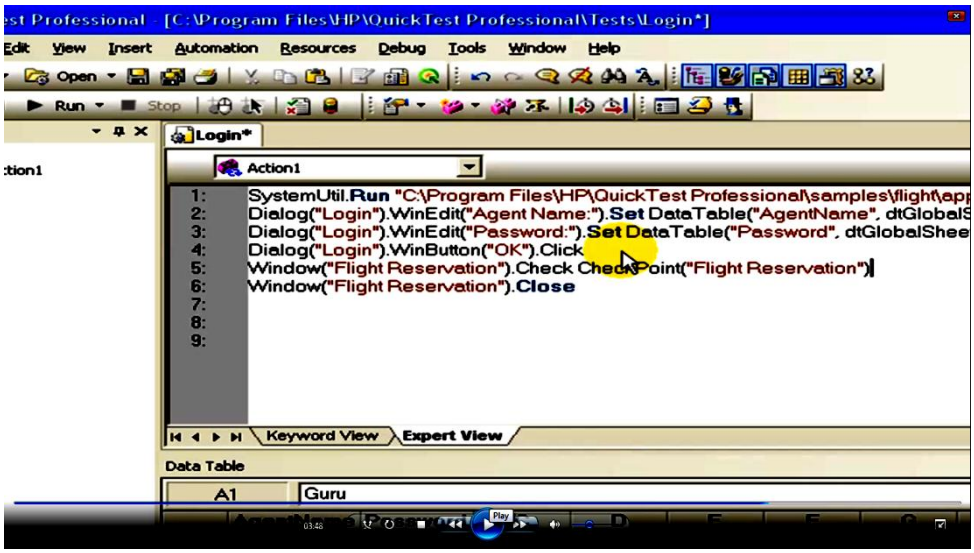
22. Like that you parameterize for password also.
23. After doing all you will see like below



24. On 5th line, rightclick on it then select standard checkpoint.Click on after the current step
25. In HP UFT, you get the script like below

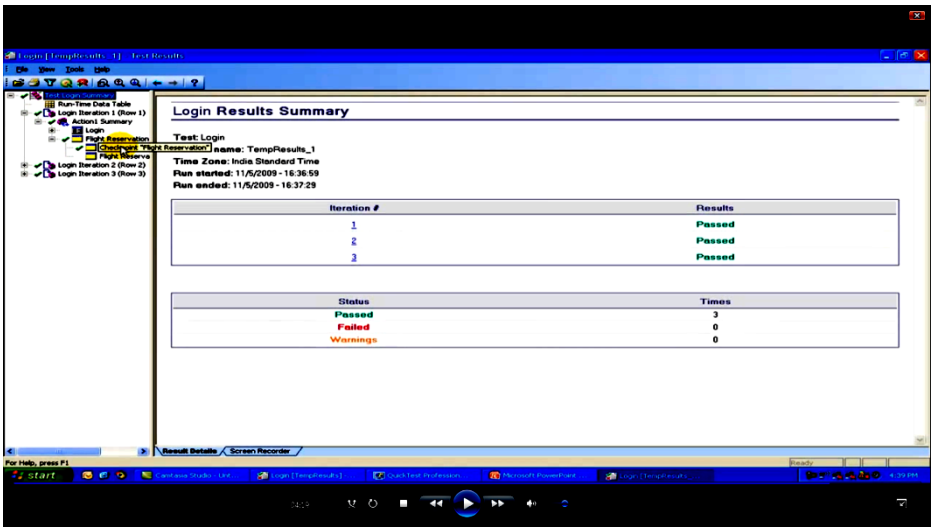


26. Run the script
27. You get the run time error
28. Bcause, when your checking properties of an object, then it should available to HP UFTto recognize
29. To run the script properly, exchange line 5 with 6 as below



30. Rerun the script
31. Analyze the result

Results:



XML checkpoint:

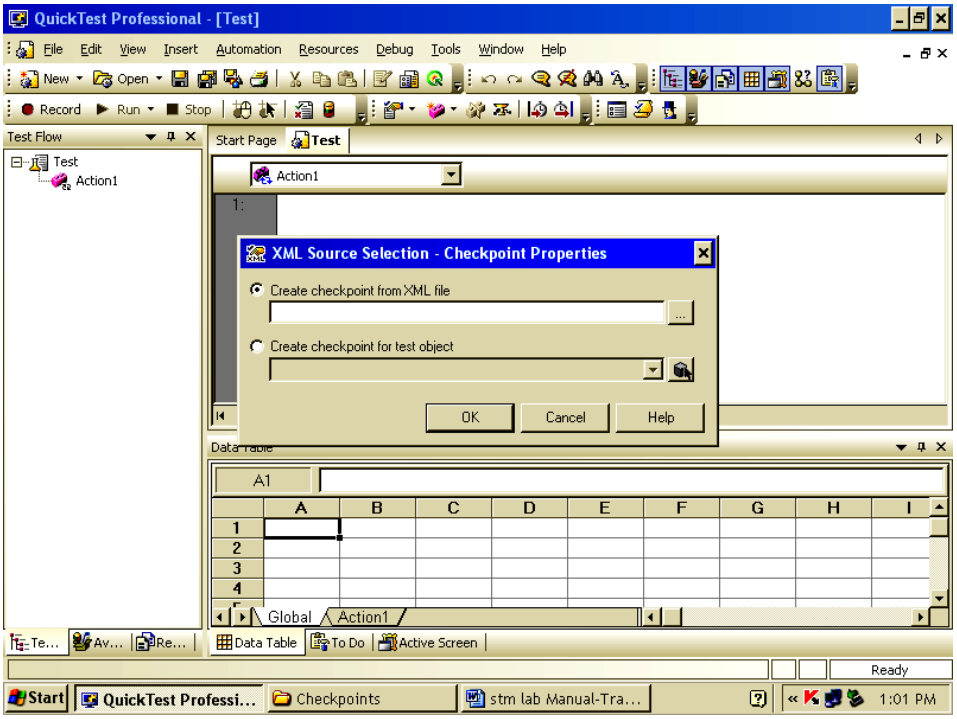
Steps:

1. In NotePad write any sample script .

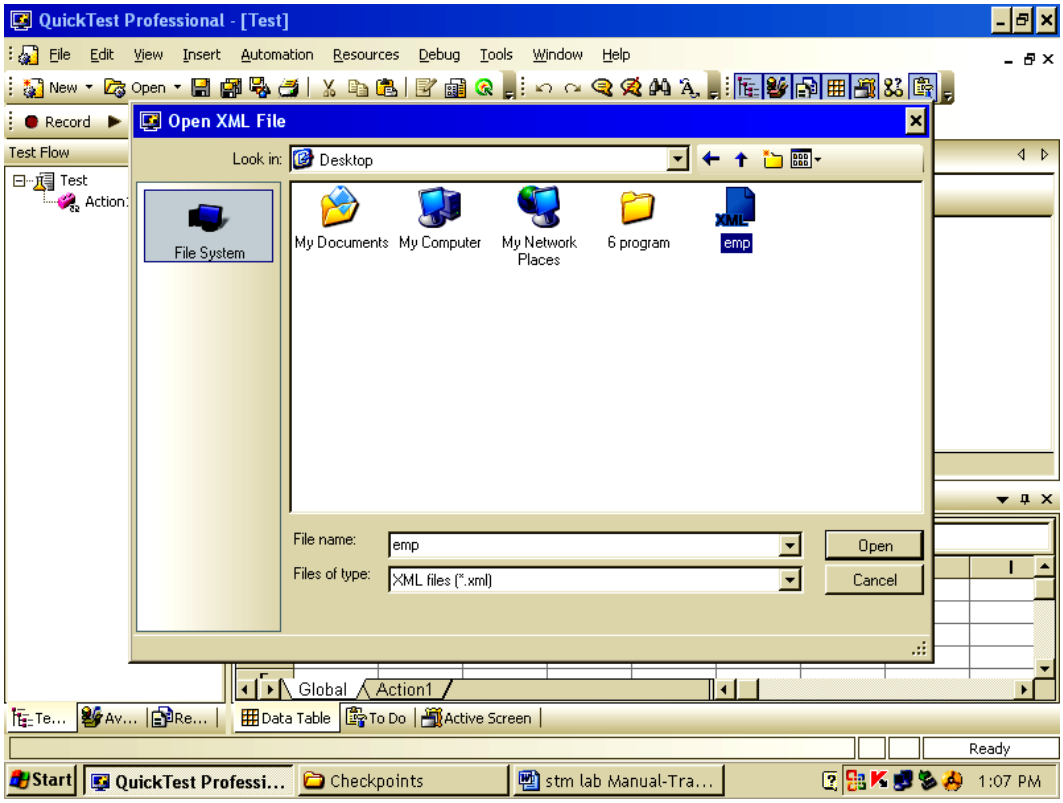
   Ex: Employees Biodata
   <EmployeesBiodata>
   <empid>1234</empid>
   <empname>mamatha</empname>
   <empdesignation>Asstprof</empdesignation>
   </employeesBiodata>
2. save as employee.xml In desktop
3. In HP UFT,

   Select design -> checkpoint -> XML Checkpoint

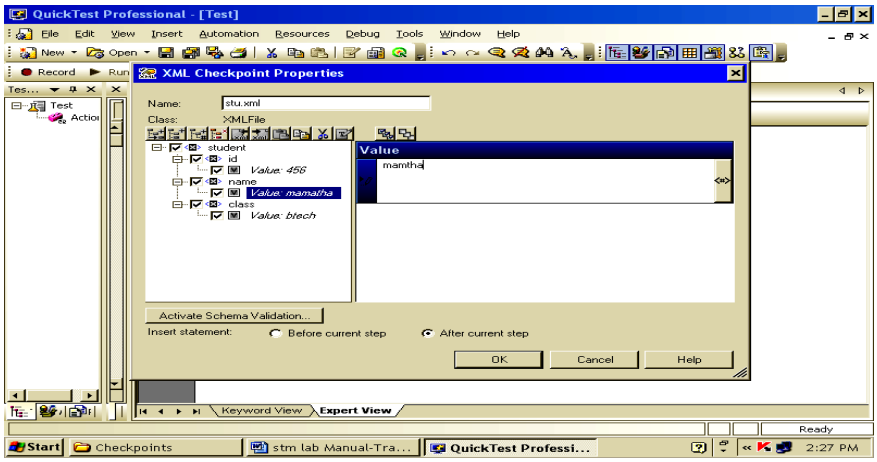
4. Then you get XML source selection dialog box opens



5. On create checkpoint from XML file, click on browse button, and select xml file



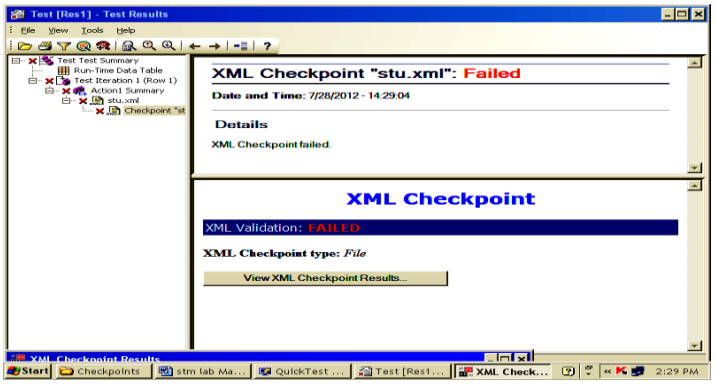
6. Left side expand the tag, do modification on any of the field, like shown below



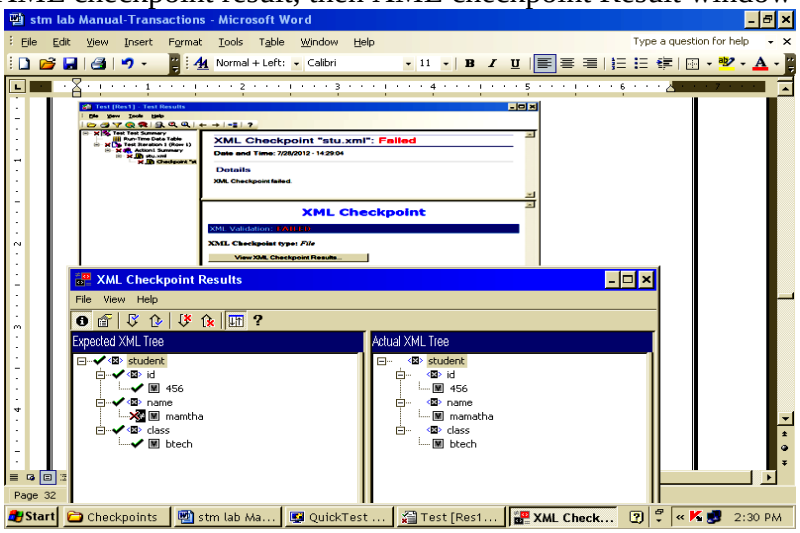
8. click ok

7. Run the script

Results:



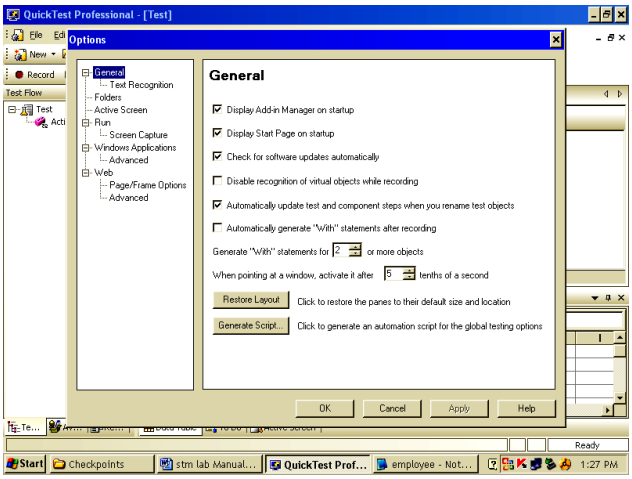
Click on View XML checkpoint result, then XML checkpoint Result window displays



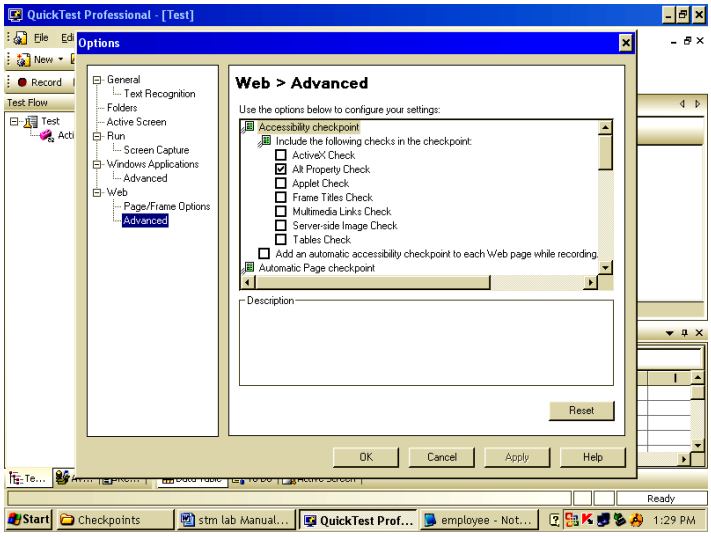
essibility Checkpoint:

Steps:

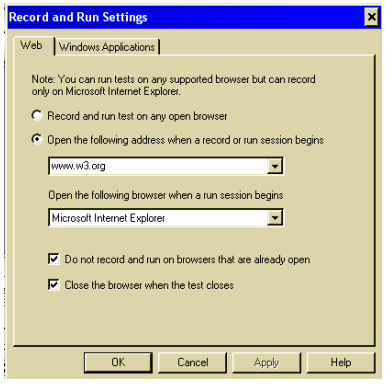
1. In HP UFT, go to Select tools -> options



2. On left side in web select Advanced



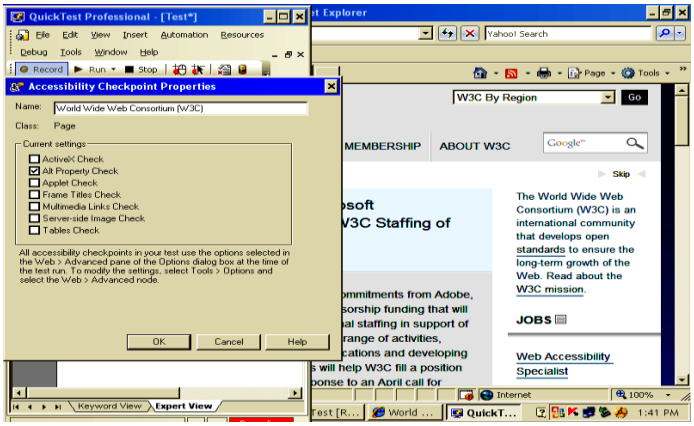
3. enable any property you need eg: Alt property
4. In HP UFT, click on Record button
5. Run and Record Setting Dialog box opens



- select 2nd option
- Give URL as www.W3.org

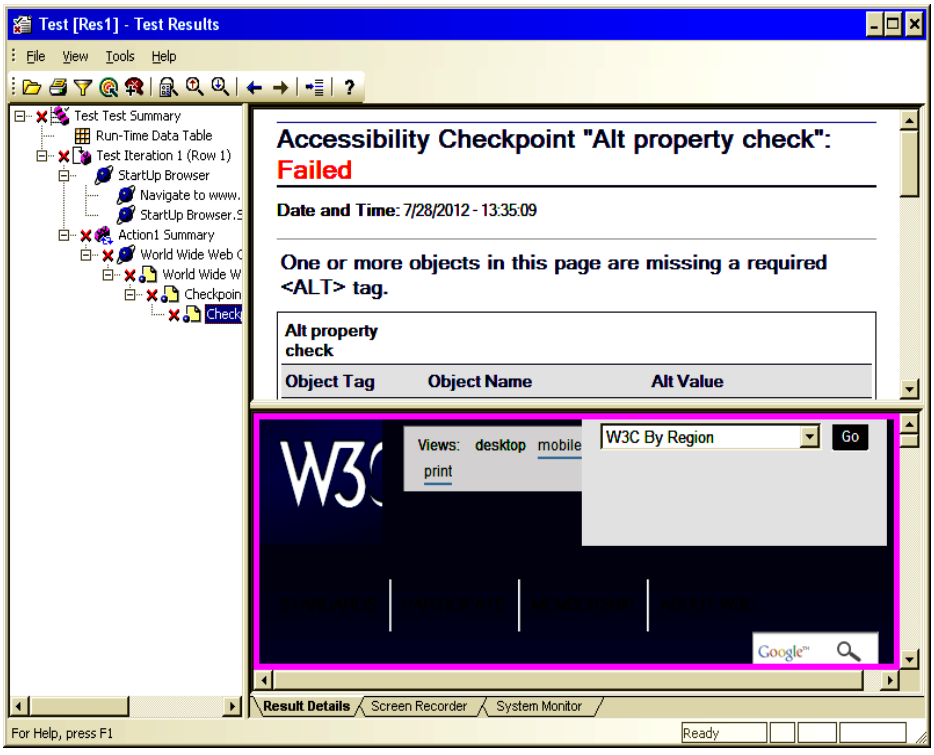
- Click on apply
- Click ok

6. Then W3C.org will launched
7. In HP UFT, design -> checkpoint -> Accessibility checkpoint
8. then cursor changed to hand icon, so click on w3.org page
9. Accessibility checkpoint properties dialog box opens, click ok on it as shown below



10. In HP UFT, you get the script.
11. Stop recording
12. Run the script

Results:



Pogram No.: 7.5

Transactions:

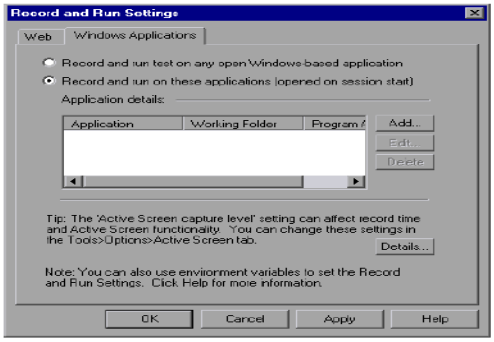
AIM: To Note time taken to login into the filght Reservation

THEORY:

- ✓ You can measure how long it takes to run a section of your test by defining transactions
- ✓ You define transactions within your test by enclosing the appropriate sections of the test with start and end transaction statements.
- ✓ Transactions can be inserted anywhere in the script
- ✓ There is no limit to the number of transactions that can be added to a test
- ✓ You can also insert a transaction within a transaction

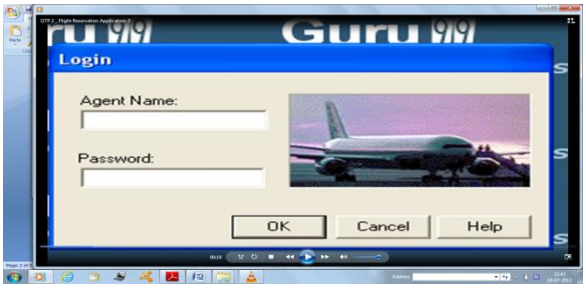
Steps:

10. Click on Record
11. You get Record and Run Setting dialog box appear.

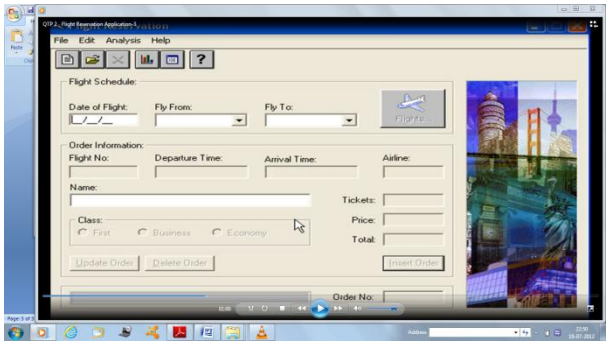


- ❖ select window application
- ❖ select 2nd option
- ❖ click ok

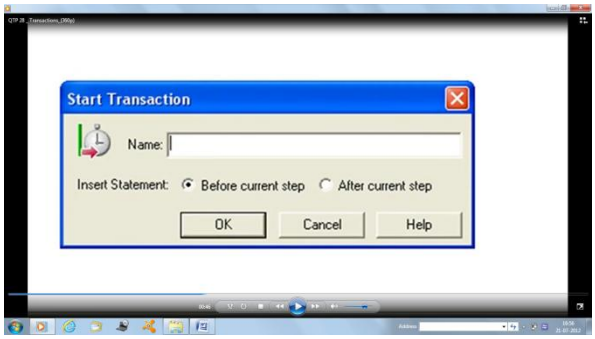
12. Login dialog box appears
    Agent Name: mercury
    Password: mercury
    Click on ok



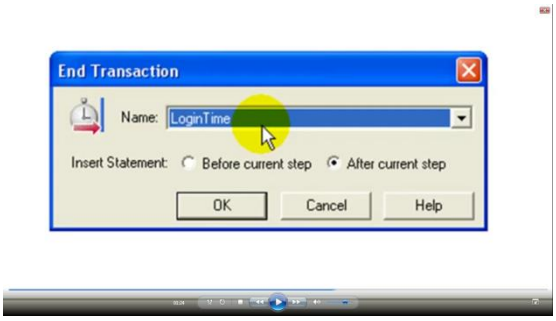
13. Then Flight Reservation window opens, then close the window



14. Stop recording
15. In HP UFT select the appropriate statement where you want to start transaction.
    In our example, place cursor in 1st line
16. And select Insert -> start transaction
17. Start transaction dialog box opens



18. Give suitable name Name: Login Time, Click ok
    Then start transaction statement added to your script
19. select a line where you want end your transaction. In our example, keep cursor in 7th line or after **window("Flight Reservation") .close** statement
20. and select insert -> end transaction
21. end transaction dialog box opens



Click ok

22. End transaction statement is added to your script like below
    Services.EndTransaction "LoginTime"
23. Run the script and Analyze the results

Results:

Object	Details
LoginTime	Transaction "LoginTime" ended with "Pass" status (Total Duration: 10.1719 sec).

Program:7.6

AIM: Synchronization Point

THEORY:

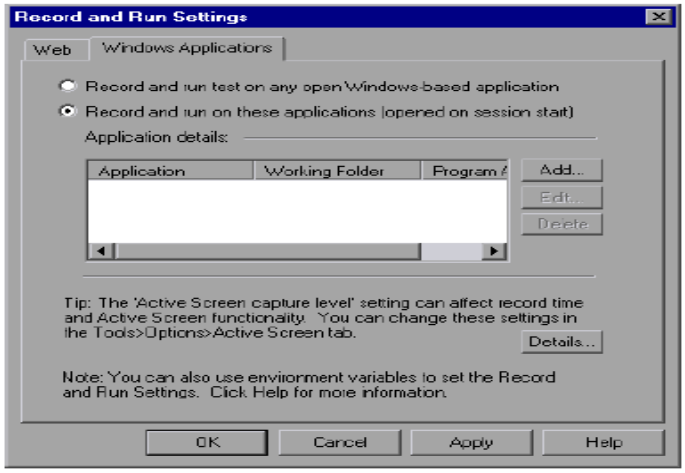
Synchronization Point:

- It is required to match execution speed of HP UFT with the loading/responding speed of application. Otherwise the test execution may fail at unexpected ways.
- If you do not want QuickTest to execute a step or checkpoint until particular object in your application appears, you should insert a synchronization point to instruct QuickTest to pause the test until the object appears (or until a specified timeout is exceeded).
- You can add synchronization point from menu (**Insert > Synchronization Point**) after start recording the test.
- And, we can use **Exist** or **Waitproperty** statement for providing synchronization

Wait() Property:

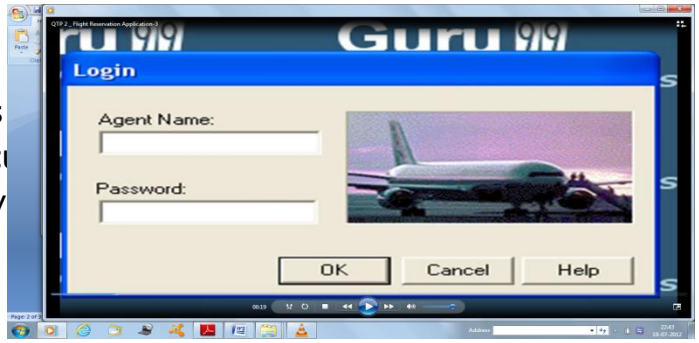
Steps:

1. Click on Record
2. You get Record and Run Setting dialog box appear.



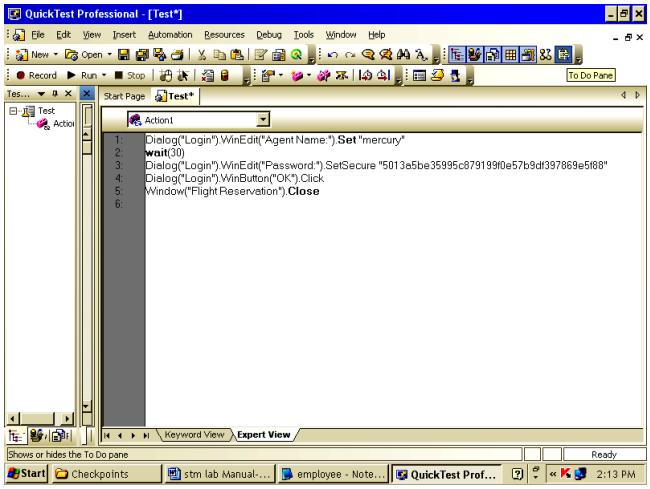
   - ❖ select window application
   - ❖ select 2nd option
   - ❖ click ok

3. Login dialog box appears
   Agent Name: mercu
   Password: mercury
   Click on ok

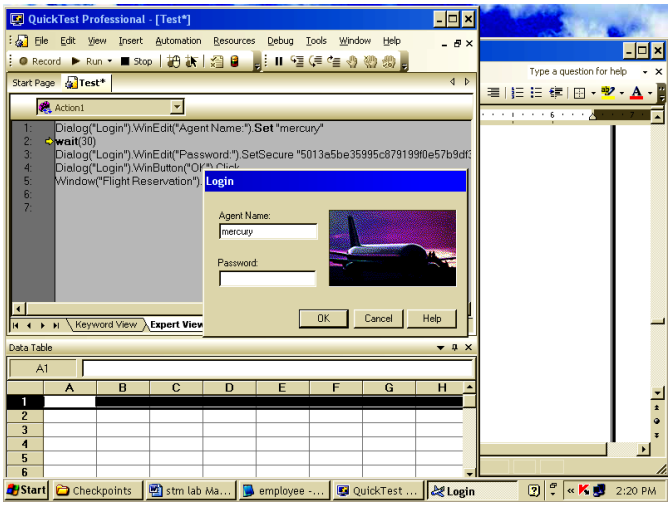


4. Then Flight Reservation window opens, then close the window
5. stop recording

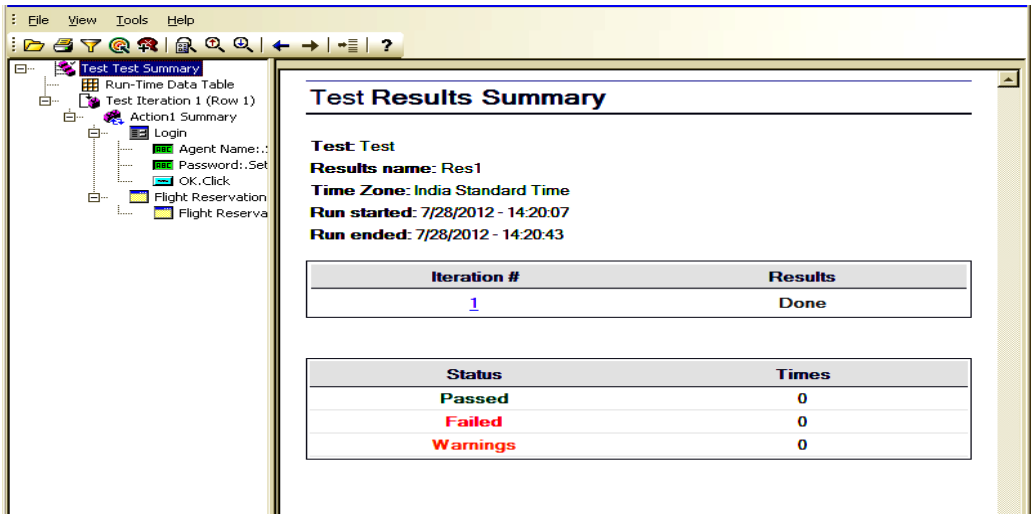
22. In HP UFT, in second line give as wait (30)



6. Analyze or observe the execution



Results:



Program: 8

AIM:Study of any Bug Tracking Tool (EX: By using Bugzilla, report a bug Details.)

THEORY:

What is Bugzilla?

- Bugzilla is a bug- or issue-tracking system.
- Bugzilla was originally written by Terry Weissman in a programming language called *TCL* and was later ported to PERL.
- Most commercial defect-tracking software vendors at the time charged enormous licensing fees, and Bugzilla quickly became a favorite of the open-source crowd (with its genesis in the open-source browser project, Mozilla).
- It is now the de-facto standard defect-tracking system against which all others are measured.
- Bugzilla boasts many advanced features. These include:
    - Powerful searching
    - User-configurable email notifications of bug changes
    - Full change history
    - Inter-bug dependency tracking and graphing
    - Excellent attachment management
    - Integrated, product-based, granular security schema
    - Fully security-audited, and runs under Perl's taint mode
    - A robust, stable RDBMS back-end
    - Web, XML, email and console interfaces
    - Completely customisable and/or localisable web user interface
    - Extensive configurability
    - Smooth upgrade pathway between versions

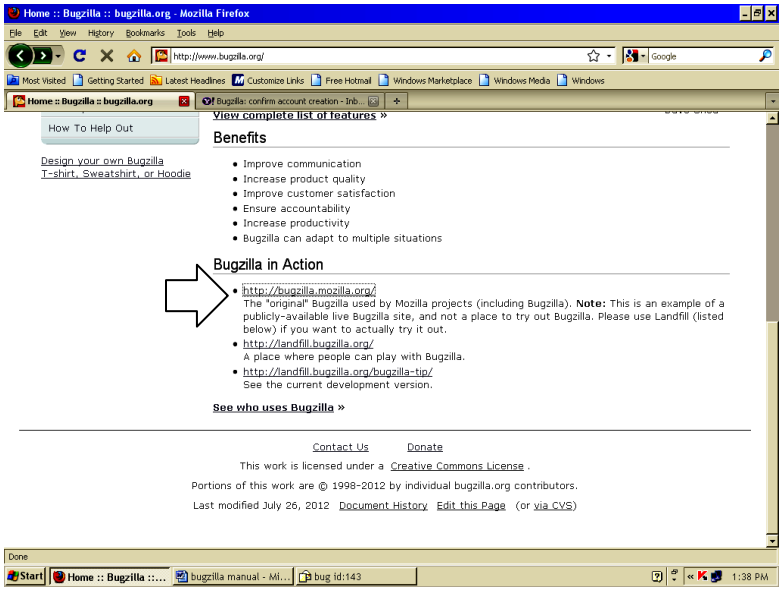
Why Should We Use Bugzilla?

- Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Redhat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS, Bonsai, or Perforce SCM, Bugzilla provides a powerful, easy-to-use solution to configuration management and replication problems.
- Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Ultimately, Bugzilla puts the power in your hands to improve your value to your employer or business while providing a usable framework for your natural attention to detail and knowledge store to flourish.

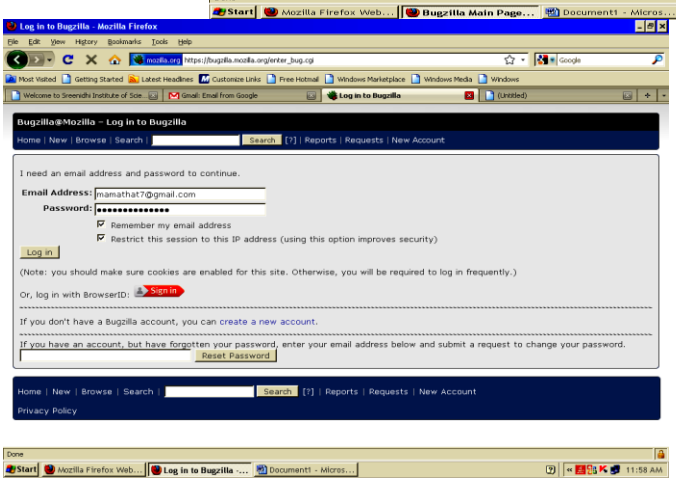
Steps:

1. Open a mozillafirfox and type www.bugzilla.org

2. On Bugzilla in Action Select 1st option as shown below

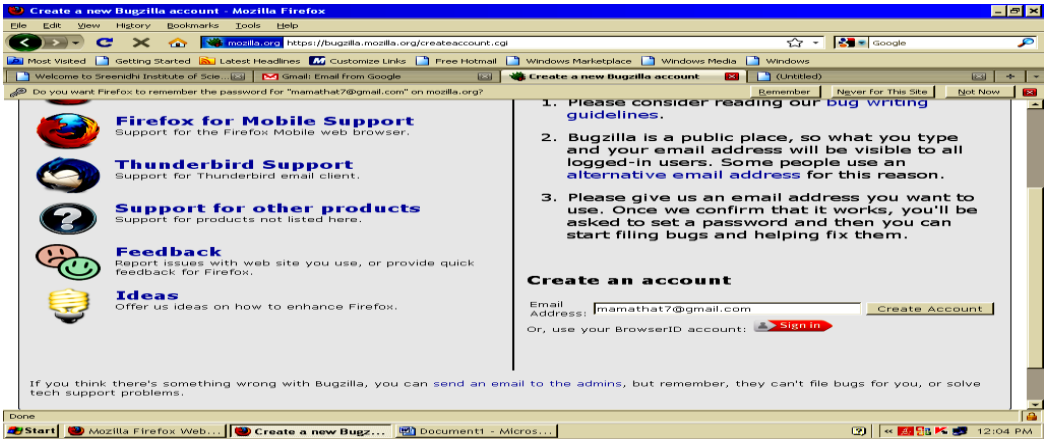


3. Then you get the screen as shown below

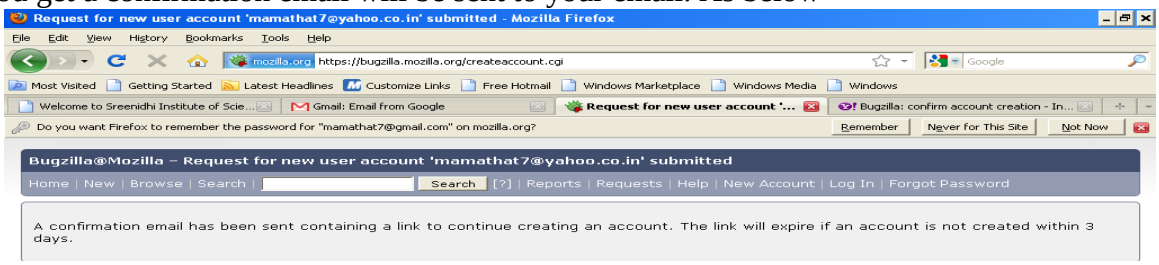


4. if you already have an account, then give Email address: Password:

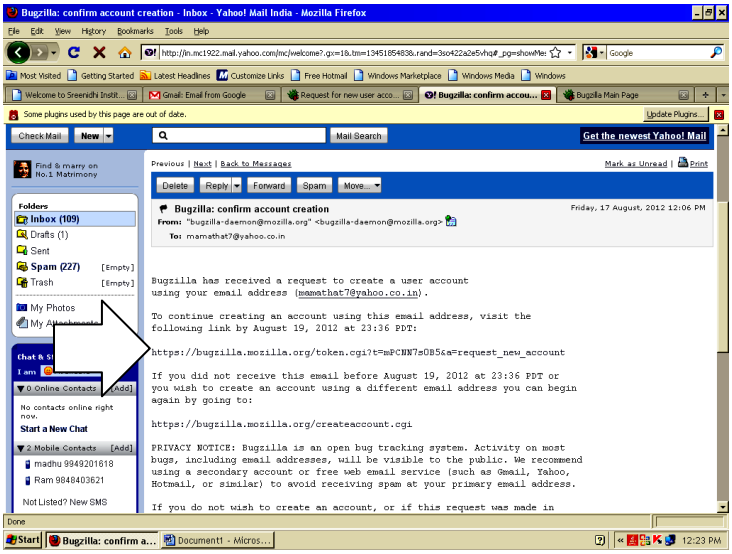
5. if you don't have an account then create a new account, by clicking on Create a new account
6. then You get the screen like this to create a new account



5. In email write your email id and click on Create Account
6.you get a confirmation email will be sent to your email. As below



7. open your email, account as below click on link



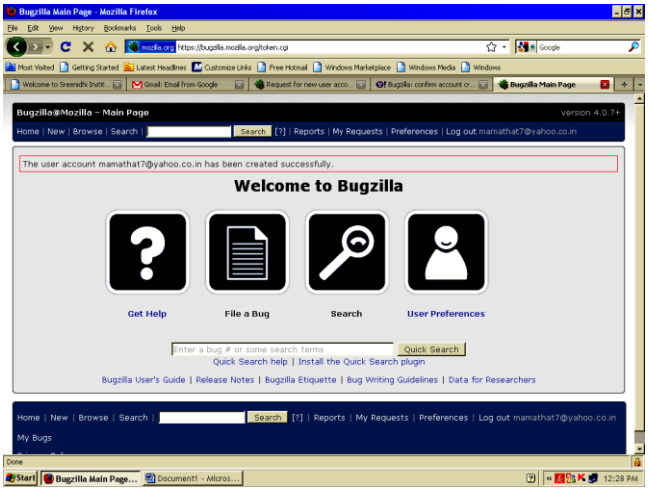
Click on the link to continue in bugzilla……

8. Then You get the screen like below
   - Write your real name:
   - Password :

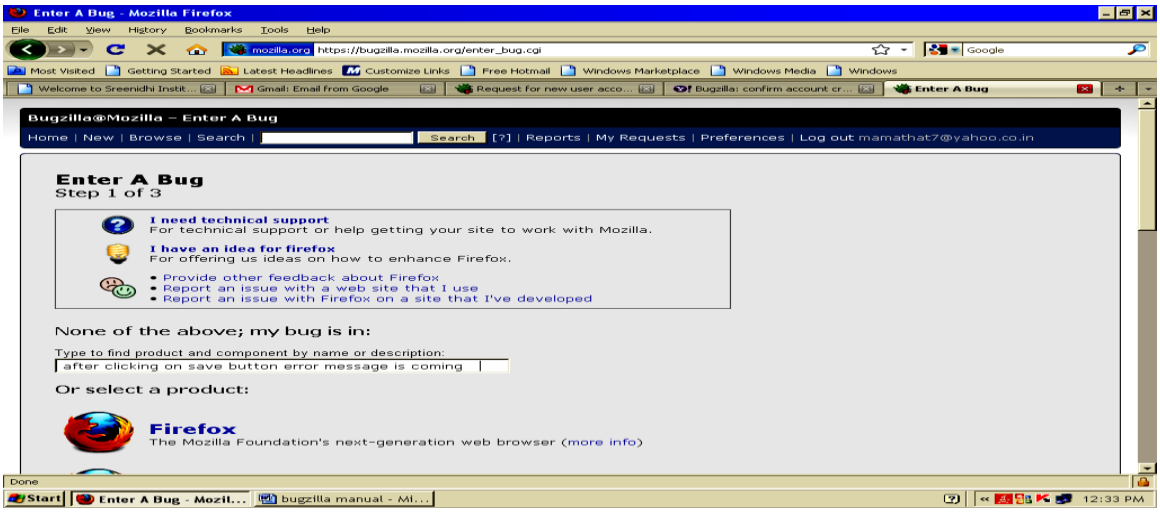
- Confirm password:

Click Create

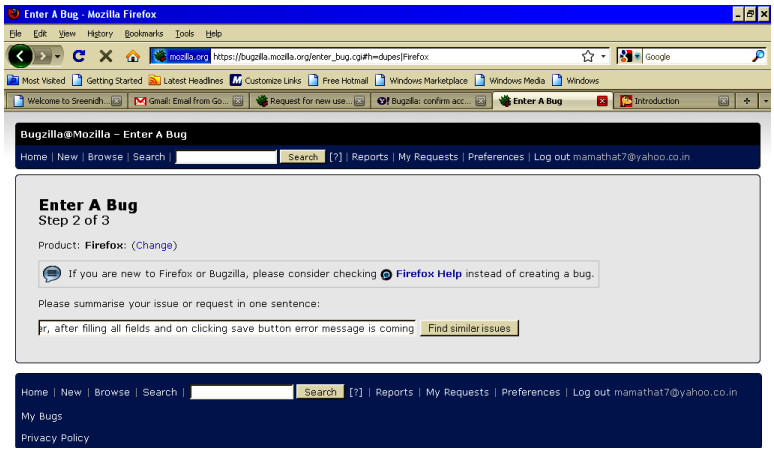
9. you get Bugzilla main page appears, indicating that your account is successfully created



10. click on File a bug option
11. Type to find product and component by name or description: in this option type your desired details.In login page: while creating new user, after clicking on save button error message is coming



12. summarize your issue in one sentence as below

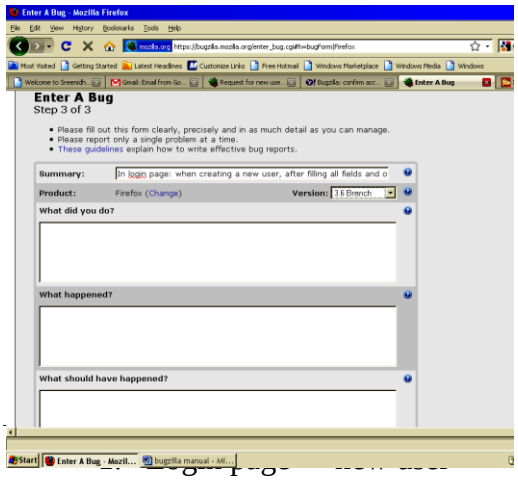


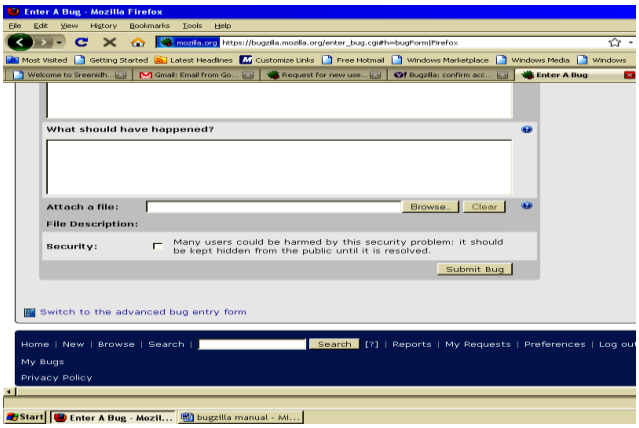
Click on find similar issues

13. Then bugzilla list out all the related bugs……. As shown below. If your issue is not in the list, click on My issue is not listed.

Click on my issue is not listed

14. you a get,





2. new user page appeared -> I filled all fields
3. I clicked on save button

What happended:

1.error message is coming, when I clicked on save button

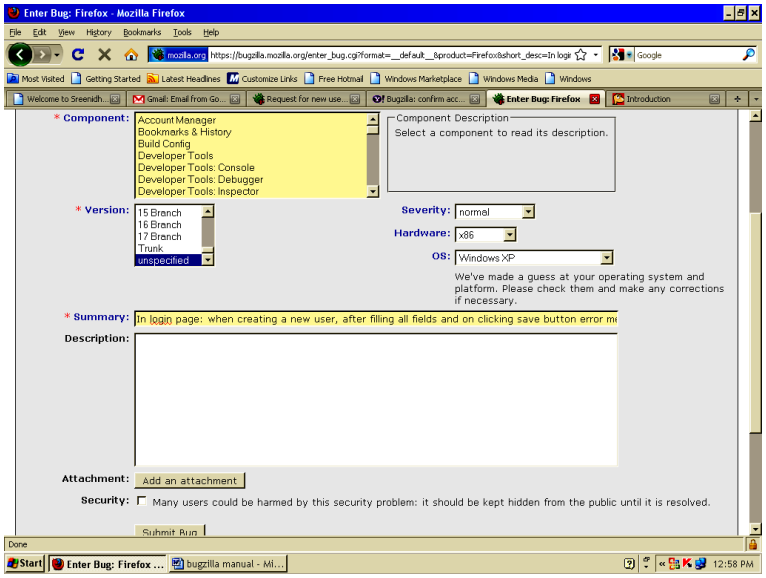
What should have happened

1. new user has been successfully created message should appear after clicking save button

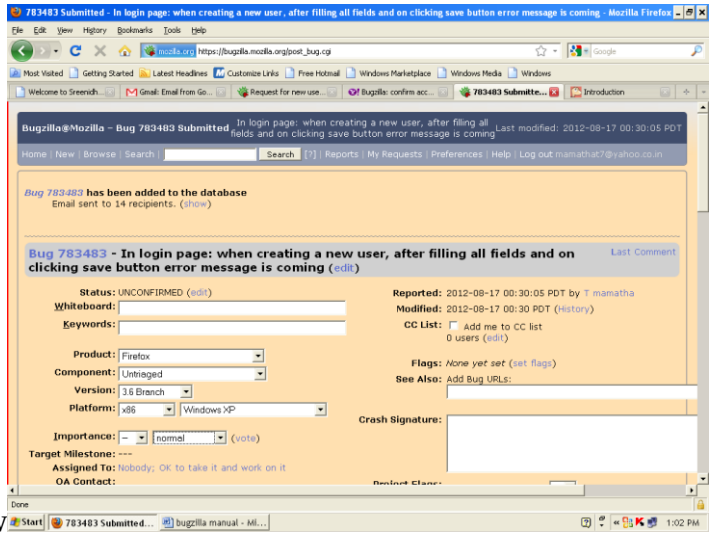
Attach file:

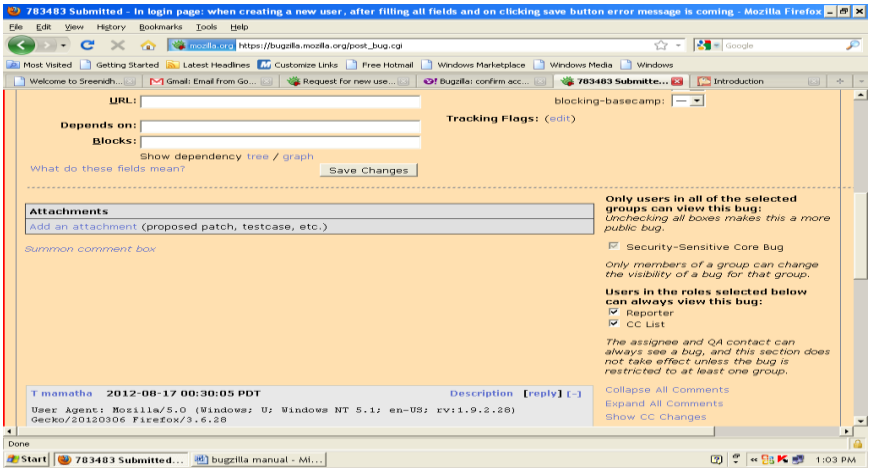
Security:

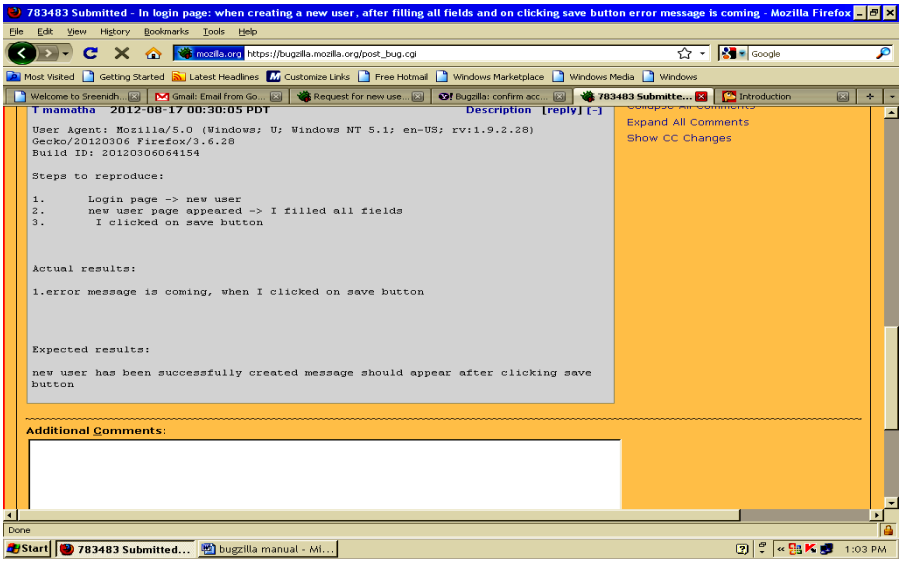
16. click on submit bug
17. on bottom, you have switch to the advanced bug entry form, if you want then you can fill on that form also



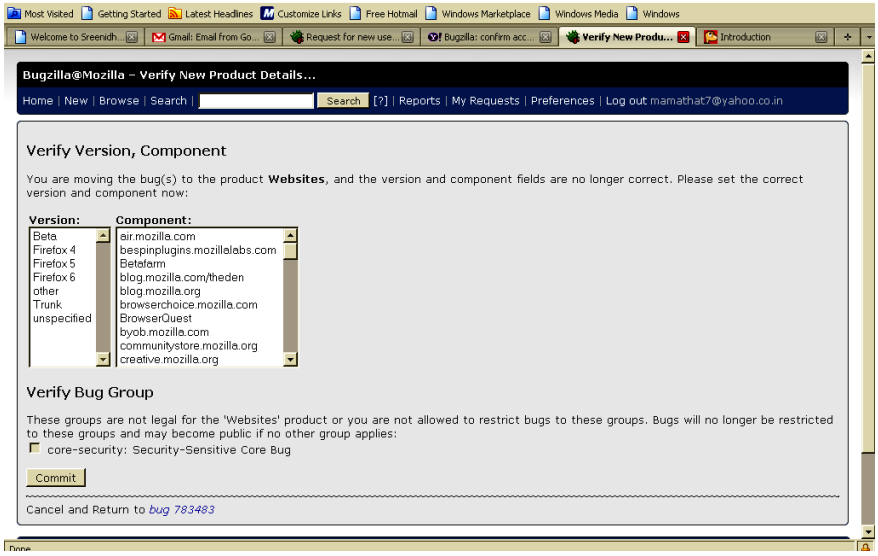
18. you get the screen as below





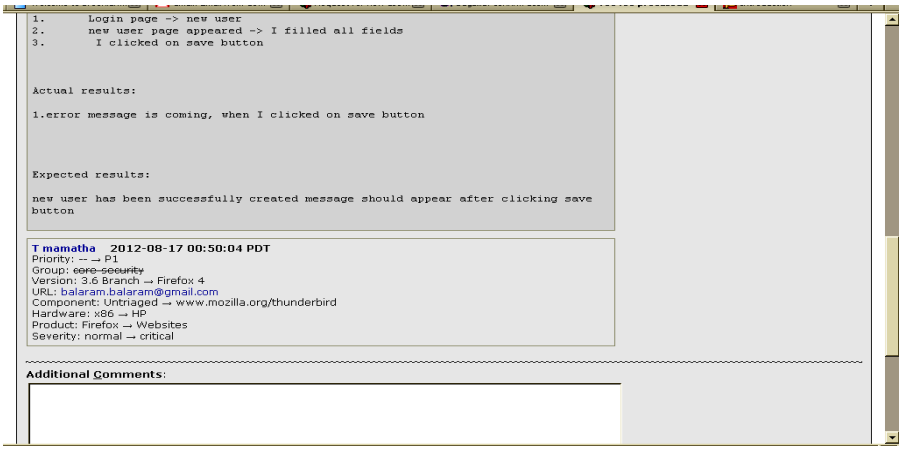


19. click on save changes

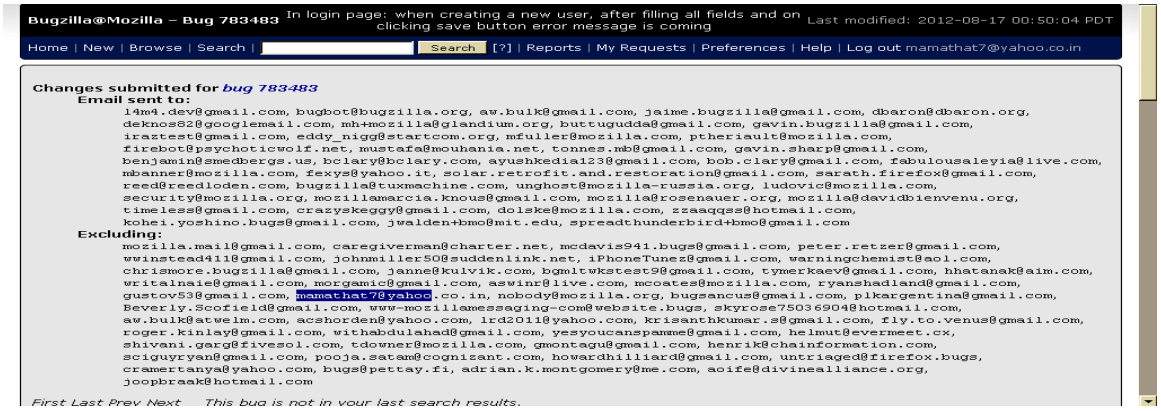


20. click on commit button
21. click on new button to enter more new bugs

Results: Bug id: 783483 is successfully sent to developer url: balaram.balaram@gmail.com
And bug information shown as below



Bug information sent to following people:

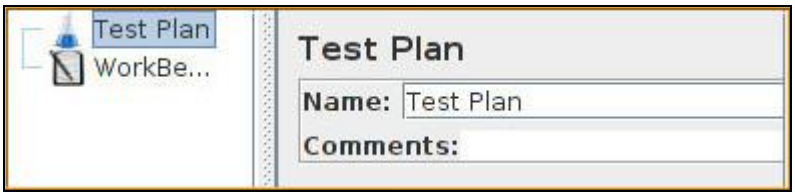


Program:9

AIM: Overview of Performance Testing Tools (Ex: Load runner)

Recording HTTP Requests in JMeter

We will study recording of HTTP requests in JMeter using **HTTP(S) Test Script Recorder** (or HTTP Proxy Server in older versions of JMeter). Let's begin with the very first step i.e. launching JMeter. Once, we launch JMeter, we can see the the two controls in the left pane -



- Test Plan - Test plan is the area where all the scripting is done and saved.
- Workbench - Workbench is the area which we use for temporary purpose, basically it is used for adding test elements that help in recording scripts in JMeter.

Recording in JMeter

Now let's see how we can record scripts in JMeter-

1. Add a Thread Group inside the Test Plan and name the Thread Group as per the action they are bound to perfrom e.g. 'LoginUsers'. Right Click on Test Plan -> Click on Add -> Threads (Users) -> Thread Group

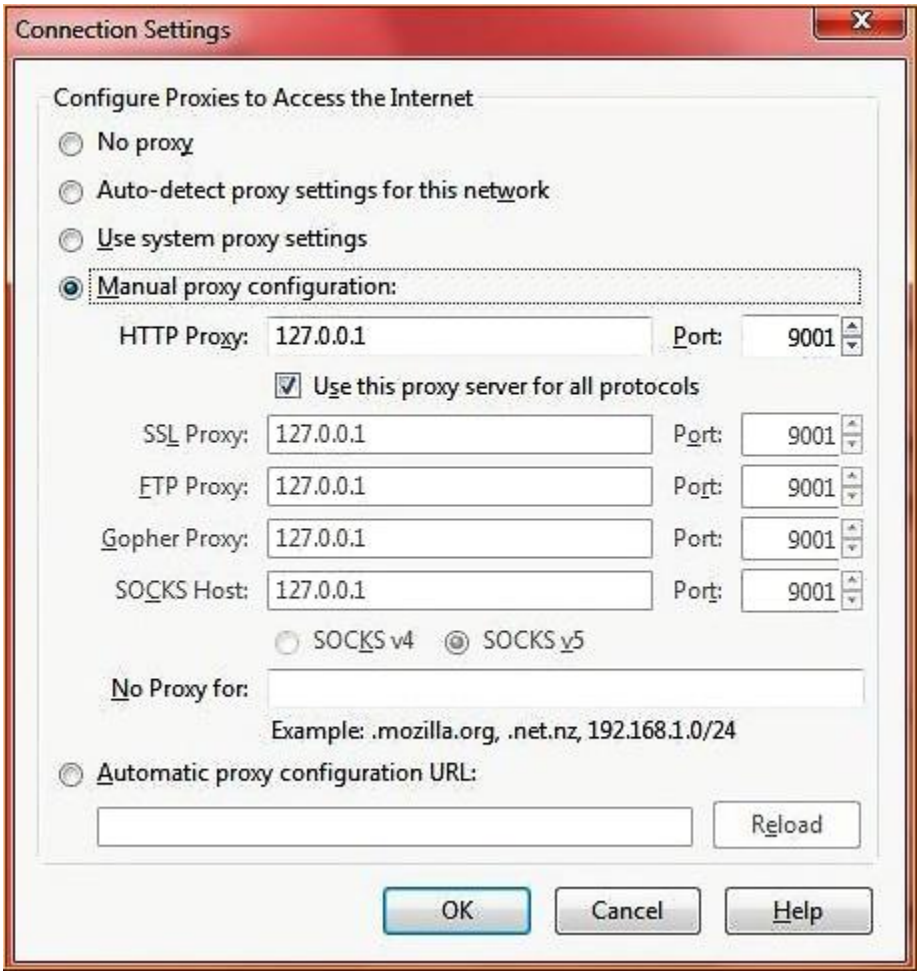
2. Add a Logic Controller (e.g. Transaction Controller) within the thread group. Right click thread group -> Click on Add -> Logic Controller -> Transaction Controller (make sure to click on generate parent sample checkbox). Add a transaction controller for each step of the user scenario of the thread group created e.g. transaction controller for "User launch the application", "User log in with valid credentials", "Click on the links of unread mails", "Log out and Exit the application".

3. Configure Browser for Proxy Settings

So, now we have got a skeleton where we can record and create scripts, next thing we will do is to record HTTP or HTTPS requests inside the transaction controllers. For this, we need to setup proxy in our browser. Although we can record calls with any browser but it is recommended to use Mozilla Firefox just because of its plug-in 'Firebug' as firebug is very important for validating whether all calls are successfully recorded or not.

Steps to set proxy in Mozilla are-

- Click on Tools-> Options
- Click on Advanced, under Advanced click on Network tab
- Click on setting, a connection Settings from would appear
- Click on manual proxy configuration radio button
- Now type 127.0.0.1 in the HTTP Proxy textbox and any available port in Port textbox like 9001

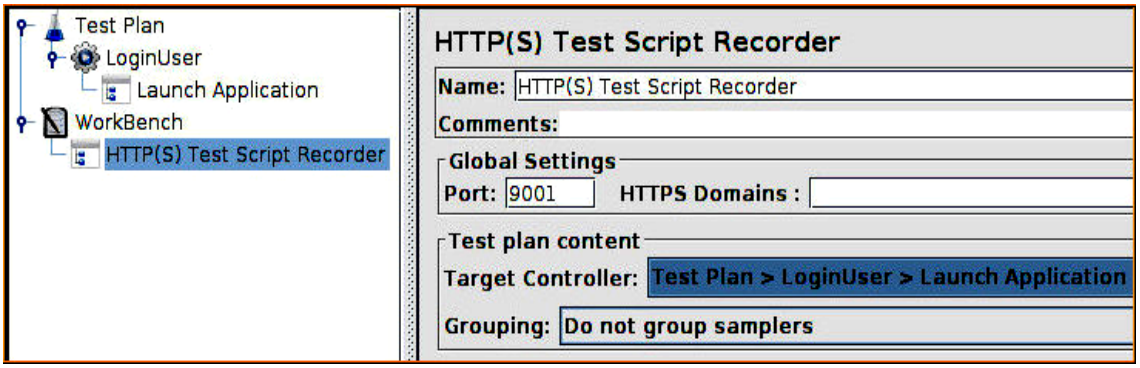


Let's see what we have done, the IP address we have entered in HTTP Proxy textbox is loopback address that is software loopback interface of the machine itself on which we are working and the port we have specified is the Port through which all the traffic will be routed, this port will be used in JMeter also.

4. **Configure JMeter to Record Scripts**

Now we will configure JMeter for recording HTTP requests in the transaction controllers -

- Right click on WorkBench.
- Click on Add-> Non Test Elements-> HTTP(S) Test Script Recorder(HTTP Proxy server in older JMeter versions).
- Enter the port value that we have entered in the Port field of our browser.
- Under Target Controller dropdown select the Transaction Controller in which we want to do the recording.
- Click on start button to start the recording.



Now whatever we do in our browser will get saved in the form of http requests in the transaction controller that we have chosen. Suppose, we chose "User launch application" as our target controller then we will click on start button on JMeter then we will go to our browser and launch the application. Again we will go to JMeter and check whether any requests are recorded in the "User launch application" transaction controller or not. If yes then we will click on stop button, chose next transaction controller from the target controller dropdown (User enter valid credentials and click on Login button), click on start button, go to browser and enter credentials and click on login button. Go to JMeter and click on stop button. Perform these steps for every transaction controller.

This completes the recording part, next thing is to add Listeners to our Test Plan for interpretation of test results. For this right click on Test Plan-> Add-> Listeners. You will get a list of all the Listeners available, as of now use Aggregate graph and View Result Tree Listeners. Now we can run the script by pressing Ctrl+R keys or click on Play icon. On top right you will see

an icon indicating that the test is running. Once complete, check the results and graphs in the Listeners. Some of the transaction controllers may fail (check in view result tree- failed requests come in red) because lots of scripting is still required in the test plan.

So, this was all about record and playback in JMeter. In the next we will study "Parameterization and Correlation" that are heart and soul of scripting.

Program:10

AIM: Study of Selenium IDE(open source testing tool)
EX: To test web-based applications that could be run in an Internet browser i.e. to record how an application is being used and then play back those recordings followed by asserts.
THEORY:

- Selenium is a FUNCTIONAL testing tool.
- Selenium is set of different software tools each with a different approach to supporting test automation.
- The entire suite of tools results in a rich set of testing functions specifically to fulfill the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior.
- One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

Introduction:

- Selenium IDE (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plug-in and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.
- It is open source software realesed under the apache 2.0 license.
  a) Selenium's Tool Suite: Selenium is composed of multiple software tools. Each has a specific role.
  b) Selenium IDE :(Selenium Recorder)
- Selenium IDE is a complete Integrated Development Environment for Selenium tests which is a extension for Firefox and allows recording, editing and debugging tests. It is prototyping tool.

Features:

- Record and playback
- Intelligent field selection will use IDs, names as needed
- Auto complete for all common Selenium commands
- Debug and set breakpoints
- Save tests as HTML,Ruby scripts or other scripts

- Support for Selenium user-extensions.js file
- Option to automatically assert the title of every page

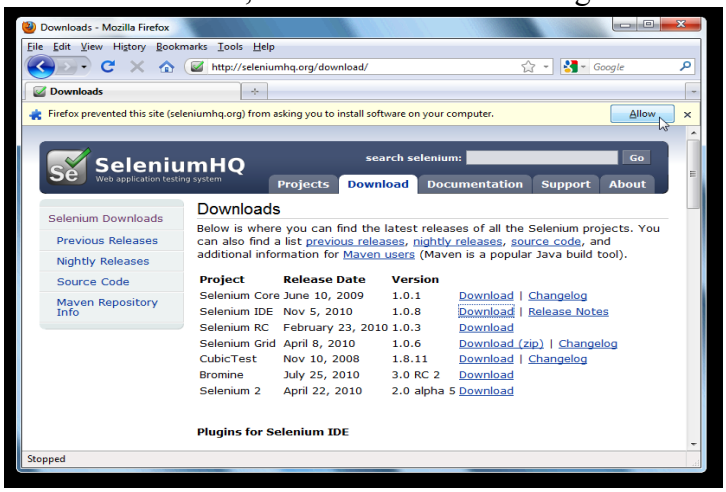
Selenium Remote Control: (Selenium 1): It is a server, written in java , that accepts commands for the browser via HTTP. RC makes it possible to write automated tests for a web application in any programming language, which allows for better integration of Selenium in existing unit test frameworks.

Selenium 2 :(Selenium Webdriver): Selenium 2 is the newest addition in the Selenium toolkit. In includes a more cohesive and object oriented API. It supports backwards compatibility with Selenium 1's and Selenium RC interface.

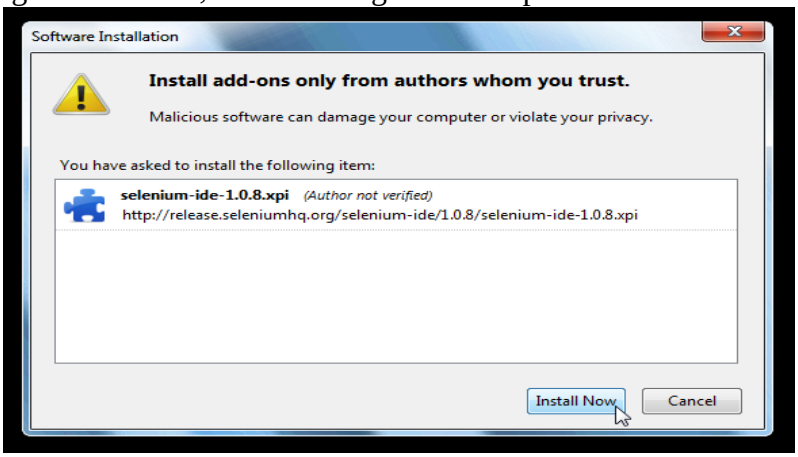
Selenium-Grid: Selenium Grid allows you to run your tests in parallel, that is, different tests can be run at the same time on different remote machines.

Steps to Install:

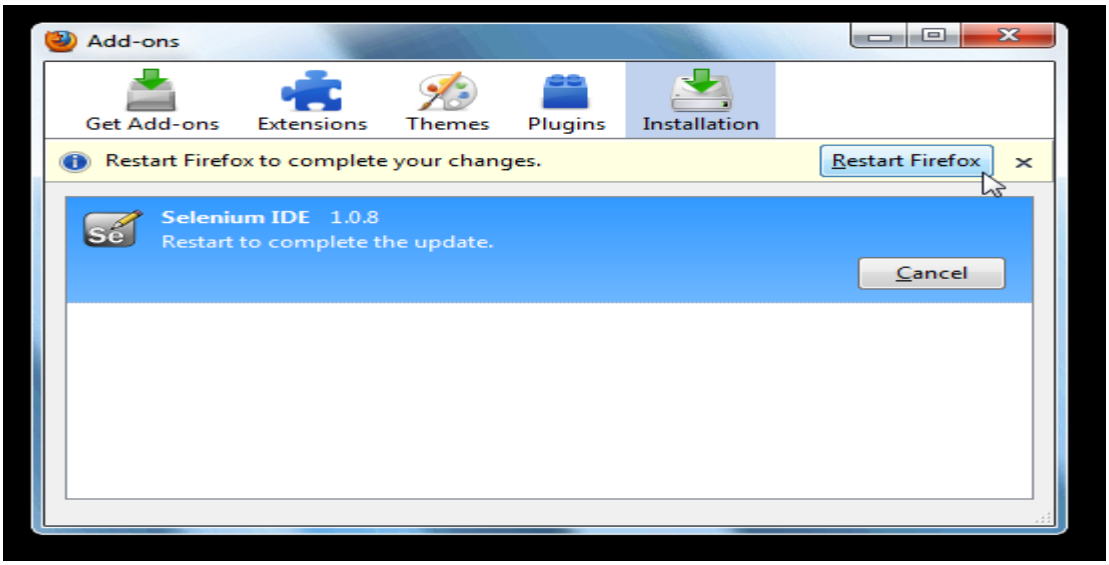
Using Firefox, first, download the IDE from the SeleniumHQ downloads page. Firefox will protect you from installing add-ons from unfamiliar locations, so you will need to click 'Allow' to proceed with the installation, as shown in the following screenshot.



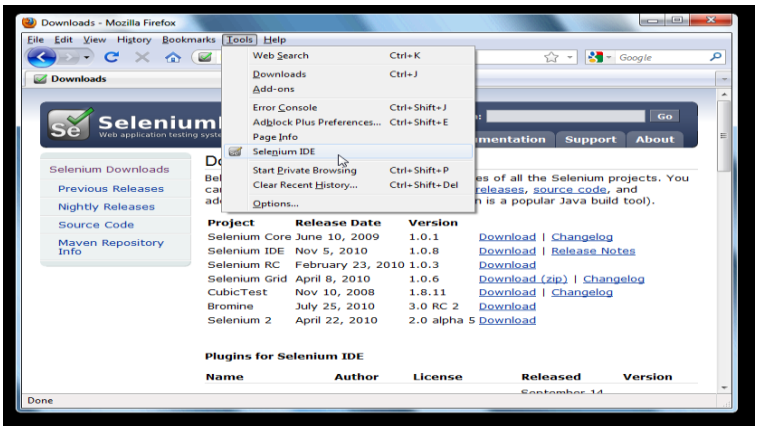
When downloading from Firefox, the following window opens



Select Install Now. The Firefox Add-ons window pops up, first showing a progress bar, and when the download is complete, displays the following.



Restart Firefox. After Firefox reboots you will find the Selenium-IDE listed under the Firefox Tools menu.



Selenium IDE Features:

b) Toolbar:The toolbar contains buttons for controlling the execution of your test cases, including a step feature for debugging your test cases

Speed Control: controls how fast your test case runs.

Run All: Runs the entire test suite when a test suite with multiple test cases is loaded.

Run: Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same effect.

Pause/Resume: Allows stopping and re-starting of a running test case.

Step: Allows you to "step" through a test case by running it one command at a time. Use for debugging test cases.

Test Runner Mode: Allows you to run the test case in a browser loaded with the Selenium-Core TestRunner. The TestRunner is not commonly used now and is likely to be deprecated. This button is for evaluating test cases for backwards compatibility with the TestRunner. Most users will probably not need this button.

Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu.

Record: Records the user's browser action

c) Test Case Pane:

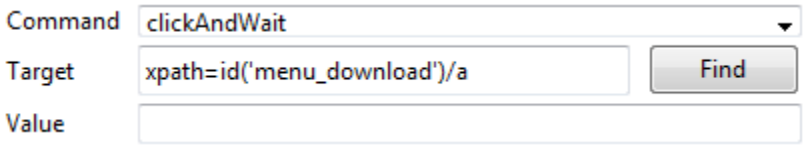
The script is displayed in the test case pane.

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

It has two tabs, one for displaying the command and their parameters in a readable "table" format.

The Command, Target, and Value entry fields display the currently selected command along with its parameters. These are entry fields where the modifications to the currently selected command can be done.

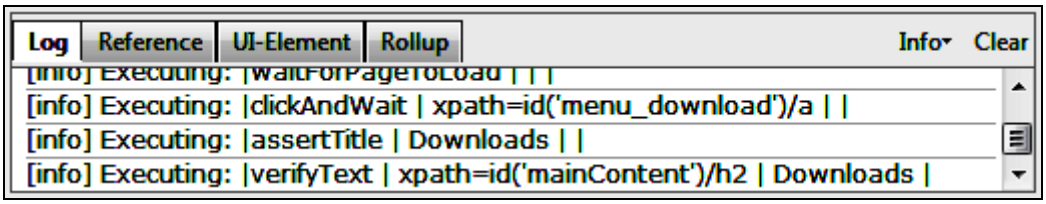
The first parameter specified for a command in the Reference tab of the bottom pane always goes in the Target field. If a second parameter is specified by the Reference tab, it always goes in the Value field.



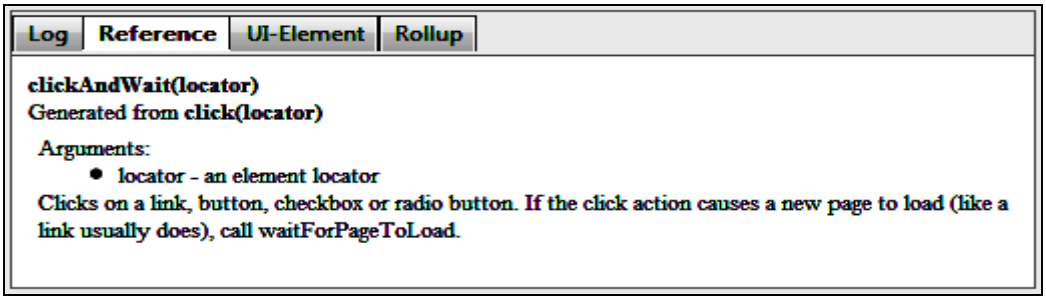
d) Log/Reference/UI-Element/Rollup Pane:

The bottom pane is used for four different functions–Log, Reference, UI-Element, and Rollup–depending on which tab is selected.

Log: When running the test case, error messages and information messages showing the progress are displayed in this pane automatically. These messages are often useful for test case debugging.



Reference: The Reference tab is the default selection for entering or modifying Selenese commands and parameters in Table mode.



e) Selenium Commands – "Selenese"

Selenium provides a rich set of commands for fully testing your web-app in virtually any way you can imagine. The command set is often called *selenese*. These commands essentially create a testing language.

A *command* is what tells Selenium what to do. Selenium commands come in three "flavors": Actions, Accessors and Assertions.

- Actions are commands that generally manipulate the state of the application. Many Actions can be called with the "AndWait" suffix, e.g. "clickAndWait". This suffix tells Selenium that the action will cause the browser to make a call to the server, and that Selenium should wait for a new page to load.
- Accessors examine the state of the application and store the results in variables, e.g. "storeTitle". They are also used to automatically generate Assertions.
- Assertions are like Accessors, but they verify that the state of the application conforms to what is expected. Examples include "make sure the page title is X" and "verify that this checkbox is checked".

Commonly Used Selenium Commands

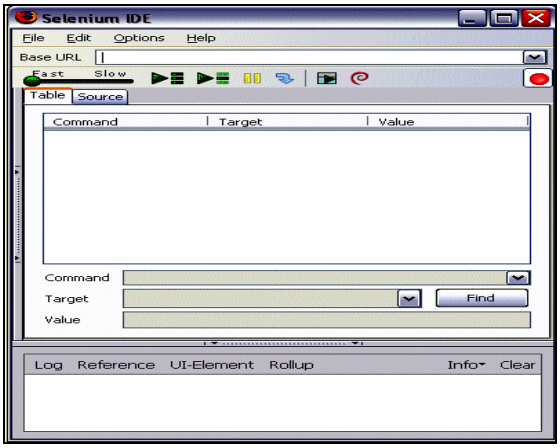
1) Open: opens a page using a URL.
2) Click/clickAndWait: performs a click operation, and optionally waits for a new page to load.
3) VerifyTitle/assert Title: verifies an expected page title.
4) VerifyTextPresent: verifies expected text is somewhere on the page.
5) VerifyElementPresent: verifies an expected UI element, as defined by its HTML tag, is present on the page.
6) VerifyText: verifies expected text and its corresponding HTML tag are present on the page.
7) VerifyTable: verifies a table's expected contents.

8) WaitForPageToLoad: pauses execution until an expected new page loads. Called automatically when clickAndWait is used.

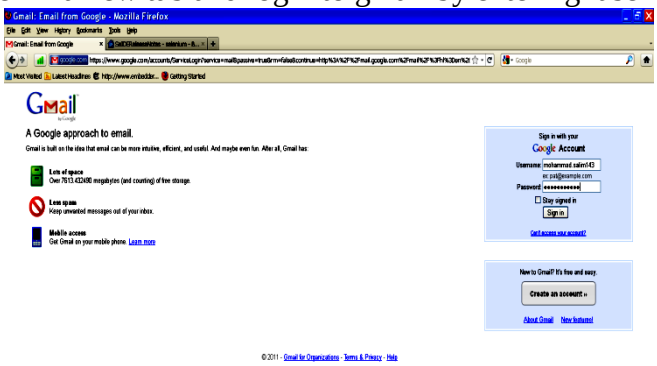
Program: 8.1

Step 1: Open the selenium IDE from the Mozilla browser

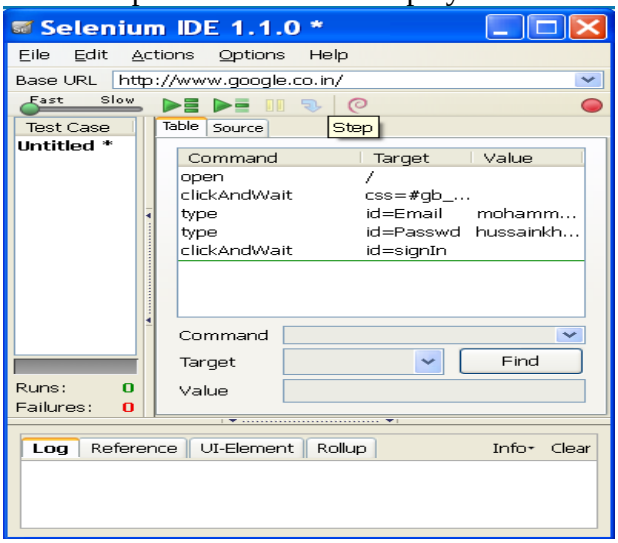
Step 2: Hit the Record Button in selenium IDE.



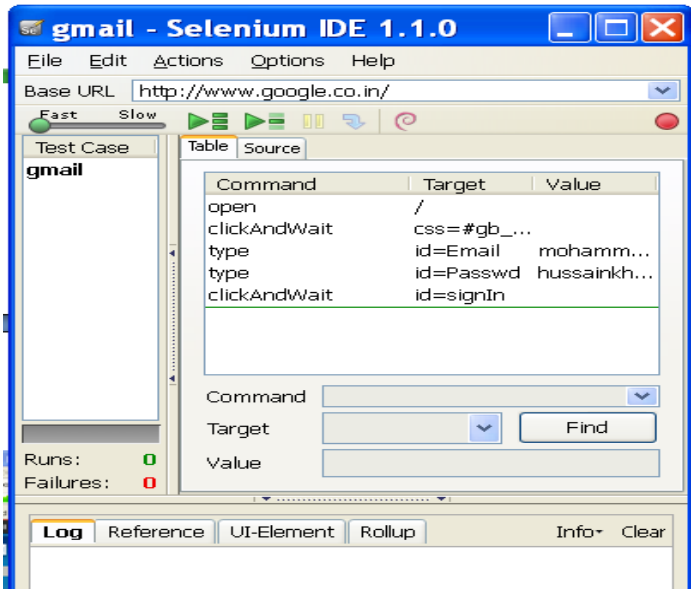
Step 3:Open the browser in a new tab and login to gmail by entering user name and password.



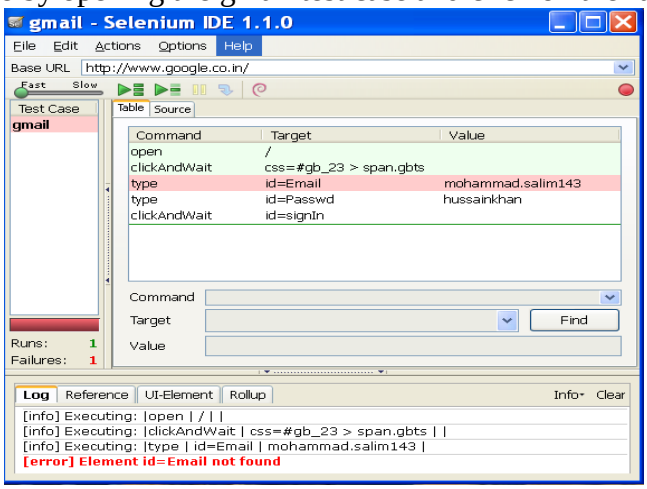
All the action performed will be displayed on the screen



- Step 4: now save the test case as "gmail"



Step 5: Run the test case by opening the gmail test case and click on the run all button.



Step 6: Finally, all operations are performed in browser.

Results:

Selenium is highly flexible. We can add functionality to both Selenium test scripts and Selenium's framework to customize your test automation. Since Selenium is Open Source, the source code can always be downloaded and modified.

Programs

1.Implement RSA algorithm

(a) Generate Public key and Private key pair

(b) Generate Ciphertext for the Plaintext

(c) Obtain the Plaintext from the Ciphertext

ALGORITHM:

PROGRAM:

(RSA)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{
clrscr();
printf("\nENTER FIRST PRIME NUMBER\n");
scanf("%d",&p);
flag=prime(p);
if(flag==0)
{
printf("\nWRONG INPUT\n");
getch();
}
printf("\nENTER ANOTHER PRIME NUMBER\n");
scanf("%d",&q);
flag=prime(q);
if(flag==0||p==q)
{
printf("\nWRONG INPUT\n");
getch();
}
printf("\nENTER MESSAGE\n");
fflush(stdin);
scanf("%s",msg);
for(i=0;msg[i]!=NULL;i++)
```

```
m[i]=msg[i];
n=p*q;
t=(p-1)*(q-1);

ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
for(i=0;i<j-1;i++)
printf("\n%ld\t%ld",e[i],d[i]);
encrypt();
decrypt();
getch();
}
int prime(long int pr)
{
int i;
j=sqrt(pr);
for(i=2;i<=j;i++)
{
if(pr%i==0)
return 0;
}
return 1;
}
void ce()
{
int k;
k=0;
for(i=2;i<t;i++)
{
if(t%i==0)
continue;
flag=prime(i);
if(flag==1&&i!=p&&i!=q)
{
e[k]=i;
flag=cd(e[k]);
if(flag>0)
{
d[k]=flag;
k++;
}
if(k==99)
break;
} } }
long int cd(long int x)
```

```
{
long int k=1;
while(1)
{
k=k+t;
if(k%x==0)
return(k/x);
} }
void encrypt() {
long int pt,ct,key=e[0],k,len; i=0;
len=strlen(msg);
while(i!=len) { t=m[i];
pt=pt-96;
k=1;
for(j=0;j<key;j++)
{ k=k*pt;
k=k%n;
}
temp[i]=k;
ct=k+96;
en[i]=ct;
i++;
}
en[i]=-1;
printf("\nTHE ENCRYPTED MESSAGE IS\n");
for(i=0;en[i]!=-1;i++)
printf("%c",en[i]);
}
void decrypt()
{
long int pt,ct,key=d[0],k;
i=0;
while(en[i]!=-1)
{
ct=temp[i];
k=1;
for(j=0;j<key;j++)
{
k=k*ct;
k=k%n;
}
pt=k+96;
m[i]=pt;
i++;
}
```

```
m[i]=-1;
printf("\nTHE DECRYPTED MESSAGE IS\n");
for(i=0;m[i]!=-1;i++)
printf("%c",m[i]);
}
```

2. Implement DES

(a) Generate Cipher text for the given Plaintext

(b) Retrieve the Plaintext from the given Ciphertext

```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES {
byte[] skey = new byte[1000];
String skeyString;
static byte[] raw;
String inputMessage,encryptedData,decryptedMessage;

public DES() {
try {
generateSymmetricKey();

inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");
byte[] ibyte = inputMessage.getBytes();
byte[] ebyte=encrypt(raw, ibyte);
String encryptedData = new String(ebyte);
System.out.println("Encrypted message "+encryptedData);
JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);

byte[] dbyte= decrypt(raw,ebyte);
String decryptedMessage = new String(dbyte);
System.out.println("Decrypted message "+decryptedMessage);

JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);
}
```

```
catch(Exception e) {
System.out.println(e);
}

}
void generateSymmetricKey() {
try {
Random r = new Random();
int num = r.nextInt(10000);
String knum = String.valueOf(num);
byte[] knumb = knum.getBytes();
skey=getRawKey(knumb);
skeyString = new String(skey);

System.out.println("DES Symmetric key = "+skeyString);
}
catch(Exception e) {
System.out.println(e);
}
}
private static byte[] getRawKey(byte[] seed) throws Exception {
KeyGenerator kgen = KeyGenerator.getInstance("DES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(56, sr);
SecretKey skey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}
```

```
private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}
```

```
public static void
main(String args[]) {
DES des = new DES();
}
}
```

3. Implement Diffie Hell man Algorithm and generate Secret Key

```
// Retrieve the prime, base, and private value for generating the key pair.
// If the values are encoded as in
// Generating a Parameter Set for the Diffie-Hellman Key Agreement Algorithm,
// the following code will extract the values.
String[] values = valuesInStr.split(",");
BigInteger p = new BigInteger(values[0]);
BigInteger g = new BigInteger(values[1]);
int l = Integer.parseInt(values[2]);

try {
// Use the values to generate a key pair
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DH");
DHParameterSpec dhSpec = new DHParameterSpec(p, g, l);
keyGen.initialize(dhSpec);
KeyPair keypair = keyGen.generateKeyPair();

// Get the generated public and private keys
PrivateKey privateKey = keypair.getPrivate();
PublicKey publicKey = keypair.getPublic();

// Send the public key bytes to the other party...
byte[] publicKeyBytes = publicKey.getEncoded();

// Retrieve the public key bytes of the other party
publicKeyBytes = ...;

// Convert the public key bytes into a PublicKey object
X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(publicKeyBytes);
KeyFactory keyFact = KeyFactory.getInstance("DH");
publicKey = keyFact.generatePublic(x509KeySpec);

// Prepare to generate the secret key with the private key and public key of the other party
KeyAgreement ka = KeyAgreement.getInstance("DH");
ka.init(privateKey);
ka.doPhase(publicKey, true);
// Specify the type of key to generate;
```

```
// see Listing All Available Symmetric Key Generators
String algorithm = "DES";

// Generate the secret key
SecretKey secretKey = ka.generateSecret(algorithm);

// Use the secret key to encrypt/decrypt data;
// see Encrypting a String with DES
} catch (java.security.InvalidKeyException e) {
} catch (java.security.spec.InvalidKeySpecException e) {
} catch (java.security.InvalidAlgorithmParameterException e) {
} catch (java.security.NoSuchAlgorithmException e) {
}
```

4. Implement Hash Algorithm

```
import java.util.Scanner;
class LinearProbingHashTable
{
private int currentSize, maxSize;
private String[] keys;
private String[] vals;
 /** Constructor **/
public LinearProbingHashTable(int capacity)
{
currentSize = 0;
maxSize = capacity;
keys = new String[maxSize];
vals = new String[maxSize];
}

/** Function to clear hash table **/
public void makeEmpty()
```

```
{
currentSize = 0;
keys = new String[maxSize];
vals = new String[maxSize];
}
 /** Function to get size of hash table **/
public int getSize()
{
return currentSize;
}
 /** Function to check if hash table is full **/
public boolean isFull()
{
return currentSize == maxSize;
}
 /** Function to check if hash table is empty **/
public boolean isEmpty()
{
return getSize() == 0;
}
 /** Fucntion to check if hash table contains a key **/
public boolean contains(String key)
{
return get(key) !=  null;
}
 /** Functiont to get hash code of a given key **/
private int hash(String key)
{
return key.hashCode() % maxSize;
```

```
}
 /** Function to insert key-value pair **/
public void insert(String key, String val)
{
int tmp = hash(key);
int i = tmp;
do
{
if (keys[i] == null)
{
keys[i] = key;
vals[i] = val;
currentSize++;
return;
}
if (keys[i].equals(key))
{
vals[i] = val;
return;
}
i = (i + 1) % maxSize;
} while (i != tmp);
}
 /** Function to get value for a given key **/
public String get(String key)
{
int i = hash(key);
while (keys[i] != null)
{
```

```
if (keys[i].equals(key))
return vals[i];
i = (i + 1) % maxSize;
}
return null;
}
 /** Function to remove key and its value **/
public void remove(String key)
{
if (!contains(key))
return;
 /** find position key and delete **/
int i = hash(key);
while (!key.equals(keys[i]))
i = (i + 1) % maxSize;
keys[i] = vals[i] = null;
 /** rehash all keys **/
for (i = (i + 1) % maxSize; keys[i] != null; i = (i + 1) % maxSize)
{
String tmp1 = keys[i], tmp2 = vals[i];
keys[i] = vals[i] = null;
currentSize--;
insert(tmp1, tmp2);
}
currentSize--;
}
/** Function to print HashTable **/
public void printHashTable()
{
```

```
System.out.println("\nHash Table: ");
for (int i = 0; i < maxSize; i++)
if (keys[i] != null)
System.out.println(keys[i] +""+ vals[i]);
System.out.println();
}
}
/** Class LinearProbingHashTableTest **/
public class LinearProbingHashTableTest
{
public static void main(String[] args)
{
Scanner scan = new Scanner(System.in);
System.out.println("Hash Table Test\n\n");
System.out.println("Enter size");
/** maxSizeake object of LinearProbingHashTable **/
LinearProbingHashTable lpht = new LinearProbingHashTable(scan.nextInt() );
 char ch;
/**  Perform LinearProbingHashTable operations  **/
do
{
System.out.println("\nHash Table Operations\n");
System.out.println("1. insert ");
System.out.println("2. remove");
System.out.println("3. get");
System.out.println("4. clear");
System.out.println("5. size");
 int choice = scan.nextInt();
switch (choice)
```

```
{
case 1 :
System.out.println("Enter key and value");
lpht.insert(scan.next(), scan.next() );
break;
case 2 :
System.out.println("Enter key");
lpht.remove( scan.next() );
break;
case 3 :
System.out.println("Enter key");
System.out.println("Value = "+ lpht.get( scan.next() ));
break;
case 4 :
lpht.makeEmpty();
System.out.println("Hash Table Cleared\n");
break;
case 5 :
System.out.println("Size = "+ lpht.getSize() );
break;
default :
System.out.println("Wrong Entry \n ");
break;
}
/** Display hash table **/
lpht.printHashTable();
 System.out.println("\nDo you want to continue (Type y or n) \n");
ch = scan.next().charAt(0);
} while (ch == 'Y'|| ch == 'y');
```

}

}

5. Implement Digital Signature

Signing the Message

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.Signature;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JOptionPane;

public class Message {
private List<byte[]> list;

//The constructor of Message class builds the list that will be written to the file.
//The list consists of the message and the signature.
public Message(String data, String keyFile) throws InvalidKeyException, Exception {
list = new ArrayList<byte[]>();
list.add(data.getBytes());
list.add(sign(data, keyFile));
}

//The method that signs the data using the private key that is stored in keyFile path
public byte[] sign(String data, String keyFile) throws InvalidKeyException, Exception{
Signature rsa = Signature.getInstance("SHA1withRSA");
rsa.initSign(getPrivate(keyFile));
```

```
rsa.update(data.getBytes());
return rsa.sign();
}

//Method to retrieve the Private Key from a file
public PrivateKey getPrivate(String filename) throws Exception {
byte[] keyBytes = Files.readAllBytes(new File(filename).toPath());
PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);
KeyFactory kf = KeyFactory.getInstance("RSA");
return kf.generatePrivate(spec);
}

//Method to write the List of byte[] to a file
private void writeToFile(String filename) throws FileNotFoundException, IOException {
File f = new File(filename);
f.getParentFile().mkdirs();
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename));
out.writeObject(list);
out.close();
System.out.println("Your file is ready.");
}

public static void main(String[] args) throws InvalidKeyException, IOException, Exception{
String data = JOptionPane.showInputDialog("Type your message here");

new Message(data, "MyKeys/privateKey").writeToFile("MyData/SignedData.txt");
}
}
```

Verifying the Signature

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
```

```
import java.security.PrivateKey;
import java.security.Signature;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JOptionPane;

public class Message {
private List<byte[]> list;

//The constructor of Message class builds the list that will be written to the file.
//The list consists of the message and the signature.
public Message(String data, String keyFile) throws InvalidKeyException, Exception {
list = new ArrayList<byte[]>();
list.add(data.getBytes());
list.add(sign(data, keyFile));
}
//The method that signs the data using the private key that is stored in keyFile path
public byte[] sign(String data, String keyFile) throws InvalidKeyException, Exception{
Signature rsa = Signature.getInstance("SHA1withRSA");
rsa.initSign(getPrivate(keyFile));
rsa.update(data.getBytes());
return rsa.sign();
}
//Method to retrieve the Private Key from a file
public PrivateKey getPrivate(String filename) throws Exception {
byte[] keyBytes = Files.readAllBytes(new File(filename).toPath());
PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);
KeyFactory kf = KeyFactory.getInstance("RSA");
return kf.generatePrivate(spec);
}
//Method to write the List of byte[] to a file
```

```
private void writeToFile(String filename) throws FileNotFoundException, IOException {
File f = new File(filename);
f.getParentFile().mkdirs();
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename));
out.writeObject(list);
out.close();
System.out.println("Your file is ready.");
}
public static void main(String[] args) throws InvalidKeyException, IOException, Exception{
String data = JOptionPane.showInputDialog("Type your message here");
new Message(data, "MyKeys/privateKey").writeToFile("MyData/SignedData.txt");
}
}
```

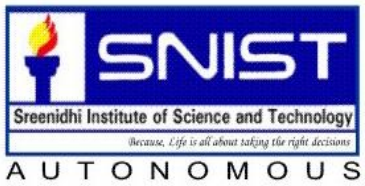
6. Implement Digital Envelope

```
import java.security.KeyPairGenerator;
import java.security.KeyPair;
import java.security.PublicKey;
import java.security.PrivateKey;
import java.security.Signature;
import java.io.FileInputStream;
public class SignatureTest {
private static byte[] sign(String datafile, PrivateKey prvKey,
String sigAlg) throws Exception {
Signature sig = Signature.getInstance(sigAlg);
sig.initSign(prvKey);
FileInputStream fis = new FileInputStream(datafile);
byte[] dataBytes = new byte[1024];
int nread = fis.read(dataBytes);
while (nread > 0) {
sig.update(dataBytes, 0, nread);
nread = fis.read(dataBytes);
};
```

```
return sig.sign();
}
private static boolean verify(String datafile, PublicKey pubKey,
String sigAlg, byte[] sigbytes) throws Exception {
Signature sig = Signature.getInstance(sigAlg);
sig.initVerify(pubKey);
FileInputStream fis = new FileInputStream(datafile);
byte[] dataBytes = new byte[1024];
int nread = fis.read(dataBytes);
while (nread > 0) {
sig.update(dataBytes, 0, nread);
nread = fis.read(dataBytes);
};
return sig.verify(sigbytes);
}
public static void main(String[] unused) throws Exception {
// Generate a key-pair
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
kpg.initialize(512); // 512 is the keysize.
KeyPair kp = kpg.generateKeyPair();
PublicKey pubk = kp.getPublic();
PrivateKey prvk = kp.getPrivate();
String datafile = "SignatureTest.java";
byte[] sigbytes = sign(datafile, prvk, "SHAwithDSA");
System.out.println("Signature(in hex):: " +
Util.byteArray2Hex(sigbytes));

boolean result = verify(datafile, pubk, "SHAwithDSA", sigbytes);
System.out.println("Signature Verification Result = " + result);
} }
```

SREENIDHI INSTITUTE OF SCIENCE & TECHNOLOGY

(An Autonomous Institution approved by UGC and affiliated to JNTUH))
(Accredited by NAAC with 'A" Grade, Accredited by NBA of AICTE and
Recipient of World Bank under TEQIP-I and II)
Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301



Lab Manual

FOR

DATA WAREHOUSING AND DATA MINING

FOR

B.Tech - IV year - I Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JUNE 2018

Vision of the Department

To emerge as a leading department in Technical Education and Research in Computer Science and Engineering with focus to produce professionally competent and socially sensitive engineers capable of working in global environment.

Mission of the Department

I. To prepare Computer Science and Engineering graduates to be a life long learner with competence in basic science & engineering and professional core, multidisciplinary areas , with continuous update of the syllabus, so that they can succeed in industry as an individual and as a team or to pursue higher studies or to become an entrepreneur.

II. To enable the graduates to use modern tools, design and create novelty based products required for the society and communicate effectively with professional ethics.

III. To continuously engage in research and projects development with financial management to promote scientific temper in the graduates and attain sustainability

Programme Educational Objectives

I. Graduates will have a strong foundation in fundamentals of mathematics, Physics, Chemistry, Computer Science and basic engineering knowledge with abilities for analysis of the problem and to design, development of solutions and to arrive at an optimal solution using modern tools which help them to be employable.

II. Ability to work in a team/ lead a team which needs effective communication skills and knowledge of project management, finance and entrepreneurial abilities.

III. Graduates should have abilities to conduct investigation of complex problems and attitude for lifelong learning skills which will enable them to pursue advanced studies, Research and Development.

IV. The graduates must be aware of the engineering professional ethics, the impact of engineering profession on the society and the need for environmental protection and sustainable development

The Programme Outcomes (POs) of the B.Tech (CSE) programme, which every graduate must attain, are listed below:

a) An ability to apply knowledge of basic sciences, mathematics and engineering in the area of Computer Science.

b) An ability to design, implement and evaluate a software or software / hardware system to meet the desired needs within realistic constraints such as space and time.

c) An ability to use the techniques, skills, and modern engineering tools such as software testing tools, data warehousing and mining tools, necessary for practice as a CSE professional.

d) An ability to analyze and solve open-ended problems using mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices and to arrive at an optimal solution.

e) To understand principles of engineering, entrepreneurship with emphasis on women, and financial management through relevant management courses to demonstrate knowledge in the conceptualization and realizing group projects, mini & main projects.

f) An ability to function effectively as individual and as a member or leader in diverse team in achieving multidisciplinary tasks.

g) Learn to communicate effectively on complex engineering activities through report writing, experimental work, assignments, seminars, group projects, mini & main projects.

h) To recognize the need for and have the preparation and ability to be a life-long learner through the courses such as seminars & projects.

i) An ability to identify, formulate and analyze engineering problems.

j) An ability to conduct investigation of complex problems in multidisciplinary areas.

k) An understanding of professional ethics and responsibilities.

l) An engineer should be aware of social, safety, cultural and information security issues and also responsibilities relevant to professional practice and skills.

m) An ability to understand the impact of environmental protection and sustainable development.

a	b	c	d	e	f	g	h	i	j	k	l
	x	x		x							

Syllabus for B. Tech. IV Year I semester
Computer Science and Engineering
Data Warehousing and Data Mining Lab

Code: 5F779

L	T	P/D	C
-	-	4	2

Data Warehousing and Data Mining Lab:

1. Build a Data Warehouse to perform filter transformation for the employee database.
2. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.
3. Using Aggregator transformation display the average salary of employees in each departments.
4. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.
5. Compare the GRI and Apriori usage (Prepare a sample data set in Spread Sheet)
6. Determine the Drugs importance w.r.t. Age, Cholestrol and BP using C 5.0
7. Predict the accuracy of the test data set using Neural Net model using a Case Study of Botanical data set.
8. Compare the C 5.0 and Neural Net using the sample data.
9. Using BASKETS1n dataset select the data as given below

   a) Customer age < 35 and count the customers who buy dairy and VEG products

   b) Find the AVG income of customers who buy atleast 5 products

   c) Derive the field whose homeown is 'YES' and Age > 30 and sort data w.r.t. income in Ascending order, and output only the item fields.

Course Outcomes:

1. Understand the workflow of Informatica (ETL: Extract, Transform and Load) tool,
2. Perform the record transformations like Expression, Filter, etc and attribute transformations like Aggregator, Joiner, etc.
3. Using data mining tool: Clementine, perform functions- GRI, Apriori, C 5.0, NeuralNet etc. for analyzing the data and analyze and compare the patterns obtained by changing the thresholds and parametric values.

Exercises/ Case Studies:

WEEK 1 : Exploring and understanding Informatica tool

WEEK 2 :

1. Build a Data Warehouse to perform filter transformation for the employee database.

WEEK 3:

1. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.

1 Using Aggregator transformation display the average salary of employees in each departments.
2 Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.

WEEK 4:

1. Using Aggregator transformation display the average salary of employees in each departments.
2. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table

WEEK 5: Exploring understanding Clementine Tool

WEEK 6:

1. Compare the GRI and Apriori usage (Prepare a sample data set in Spread Sheet)
2. Determine the Drugs importance w.r.t. Age, Cholestrol and BP using C 5.0

WEEK 7:

1. Predict the accuracy of the test data set using Neural Net model using a Case Study of Botanical data set.
2. Compare the C 5.0 and Neural Net using the sample data.

WEEK 8:

1. Using BASKETS1n dataset select the data as given below

   a) Customer age < 35 and count the customers who buy dairy and VEG products

   b) Find the AVG income of customers who buy atleast 5 products

   c) Derive the field whose homeown is 'YES' and Age > 30 and sort data w.r.t. income in Ascending order, and output only the item fields.

Course Outcomes and Relevant Program Outcomes for Data warehousing And Data Mining Lab

Course Outcomes	Program Outcomes
1. Explain concepts of data mining its features and Functionalities **Explain Data Preprocessing involving** Data Cleaning, Data Integration and Transformation ,Concept Hierarchy Generation. Data Mining Primitives, Data Mining Query Languages and Architectures.	**c,i,j**
2. Explain the fundamentals of Data Warehousing and issues of Mining with respect to architectures , technologies such as OLAP , Data Cube computation .	
3. Discuss Data Generalization and Summarization, Analysis of Attribute Relevance and Mining Class Comparisons.	
4. Apply the algorithms for mining Association rules in large databases.	
5. Discuss and apply the models of classification and use those models for prediction of the new samples.	
6. Apply various clustering techniques available for numerous applications. identify the optimal clustering technique for a particular application	

INTRODUCTION TO DATA MINING

Data Mining is a general term, which describes a number of techniques used to identify pieces of information or decision-making knowledge in data. A common misconception is that it involves passing huge amounts of data through intelligent techonologies that find patterns and give solutions to business problems.

Data Mining is an interactive and iterative process.

Many of the techniques used in Data Mining are referred to as “machine learning” or “modeling”. Historical data are used to generate models, which can be applied at a later date to areas such as prediction, forecasting, estimation and decision support.

The data mining process model recommended for use with Clementine is the Cross-Industry Standard Process for Data Mining (CRISP-DM). The first version of the CRISP-DM process model is now available. It is included with Clementine and can be downloaded from www.crisp-dm.org.

WEEK 1 : Exploring and understanding Informatica tool

PREREQUISITES OF DATA WAREHOUSING USING INFORMATICA POWER CENTER EXPRESS (PERSONAL EDITION) TOOL

I. Installing the Tool:

Make sure that you have the latest version of Java installed on your system and the PATH variable set in the System Environment Variables sub-section of Advanced System Settings of your computer. Then install Informatica PowerCenter Express (Personal Edition) version 9.6.1 as per the instructions given by the manual. Remember the username and password that you entered for the server while installation as these settings will be required for the tool to work properly.

II. Setting up the Data Source Name (DSN):

A **Data Source Name** (DSN) is the logical **name** that is used by Open Database Connectivity (ODBC) to refer to the drive and other information that is required to access **data**. The **name** is use by Internet Information Services for a connection to an ODBC **data source**, such as a Microsoft SQL Server database or in this case **Oracle Database 10g Express Edition.**

The steps to configure a Data Source are as follows:

1. Type in **ODBC Data Sources** into the Search Bar and run it with Administrator privileges.
2.

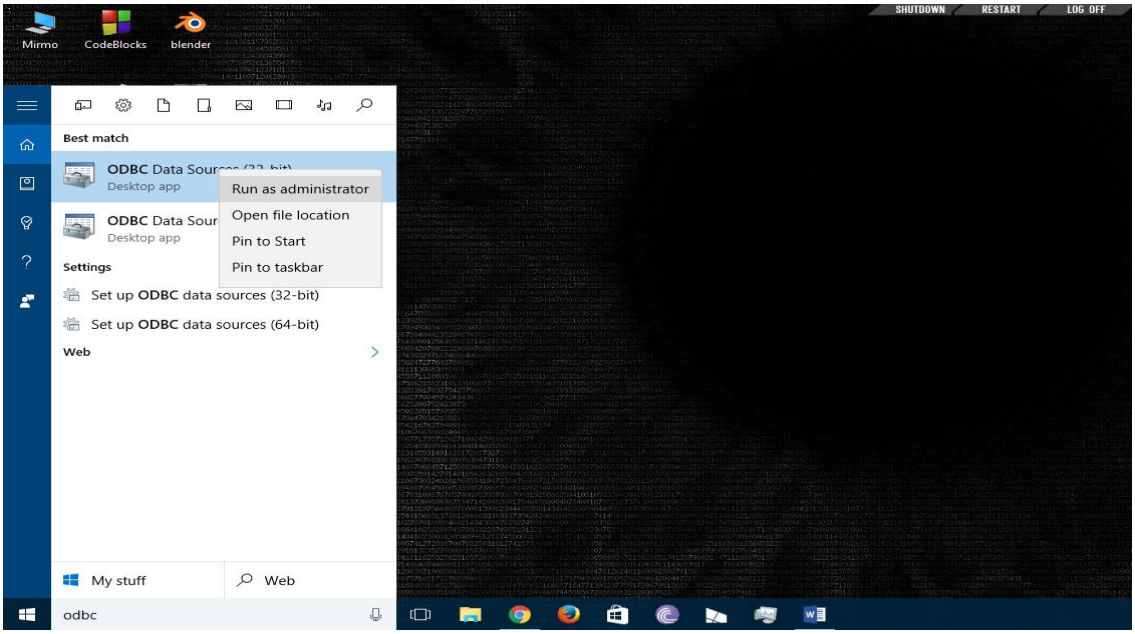


Fig 2.1: Step 2.1

3. Click on **Add** button to add a DSN and select **Oracle in XE** from the list that appears and then select **Finish**.

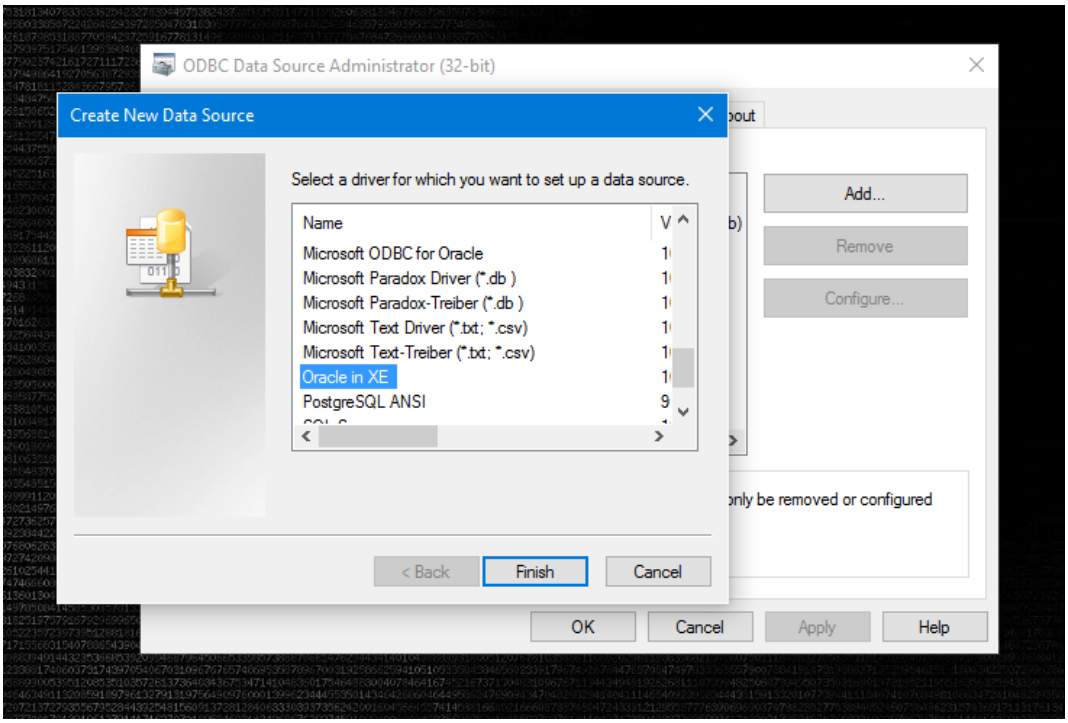


Fig 2.2: Step 2.2

4. Give your data source a name and then click on OK.

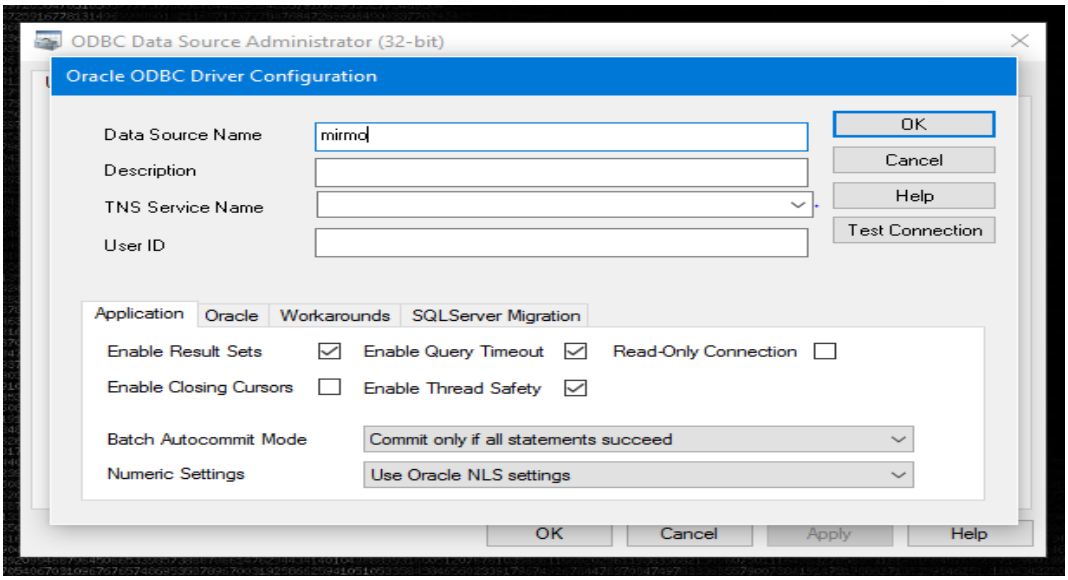


Fig 2.3: Step 2.3

Now you have created a new DSN for use in the tool.

III. Starting Informatica PowerCenter Express

Step 1:After booting the system up, go to Start Menu and run **Start Informatica Services** with Administrator privileges.

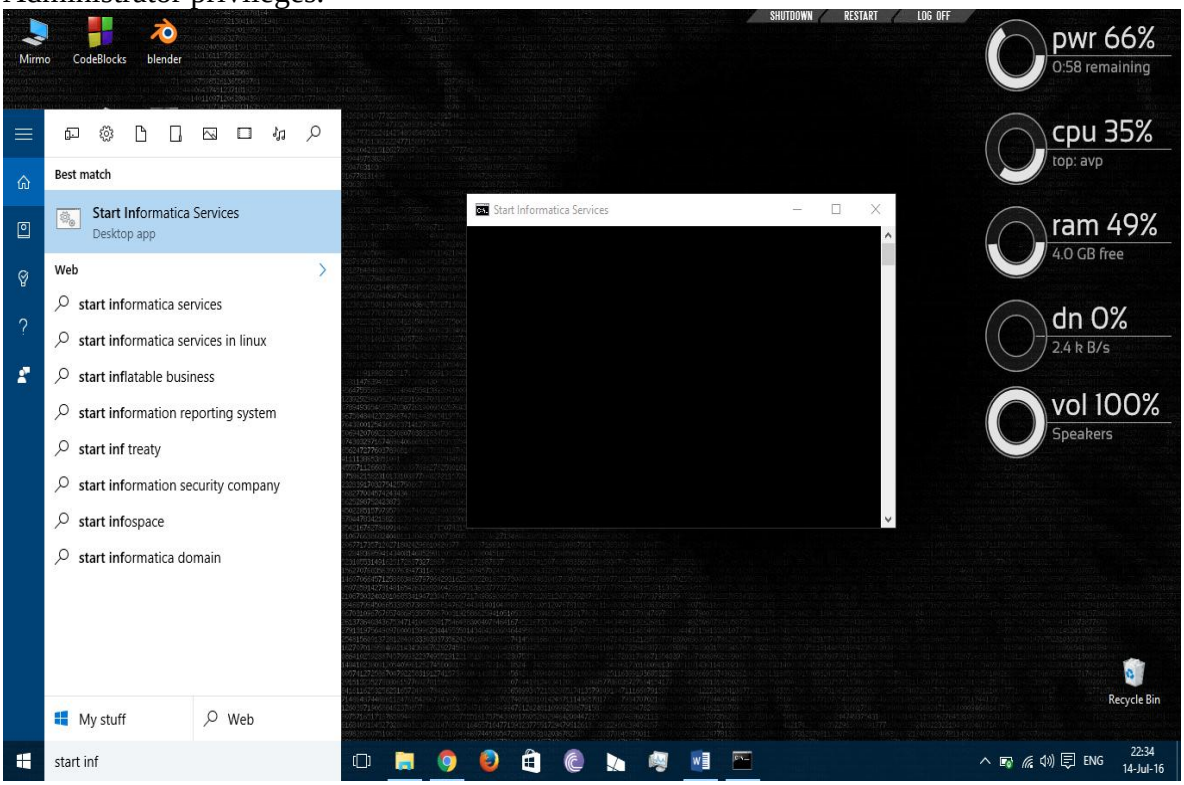


Fig. 3.1: Step 3.1

Step 2:Once the batch file runs its course, navigate to **localhost:7009** in your favourite browser. This is the login page for the Administrator Server. Login with the credentials that were supplied during the install process.

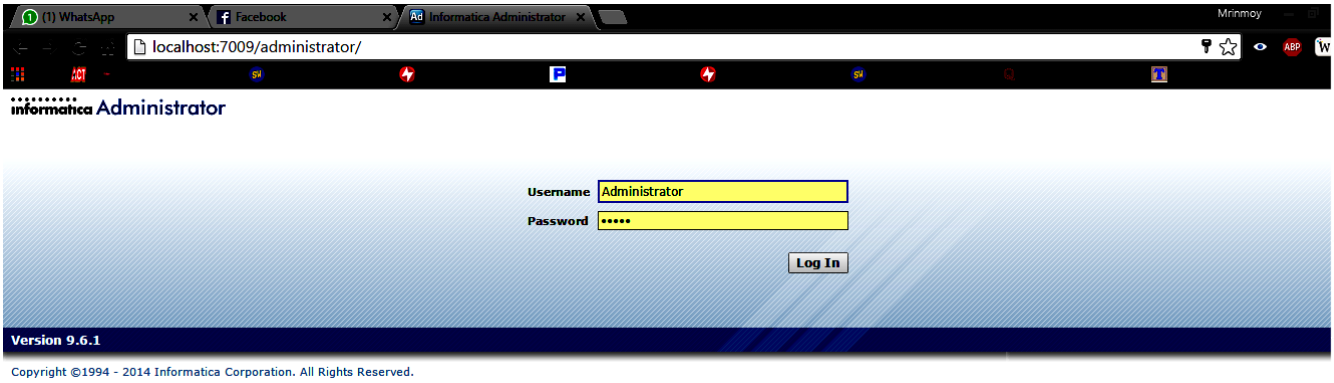


Fig 3.2: Step 3.2

*Step 3:*Once the server page starts up, we are ready to **Launch Informatica Developer** from the Search Bar.

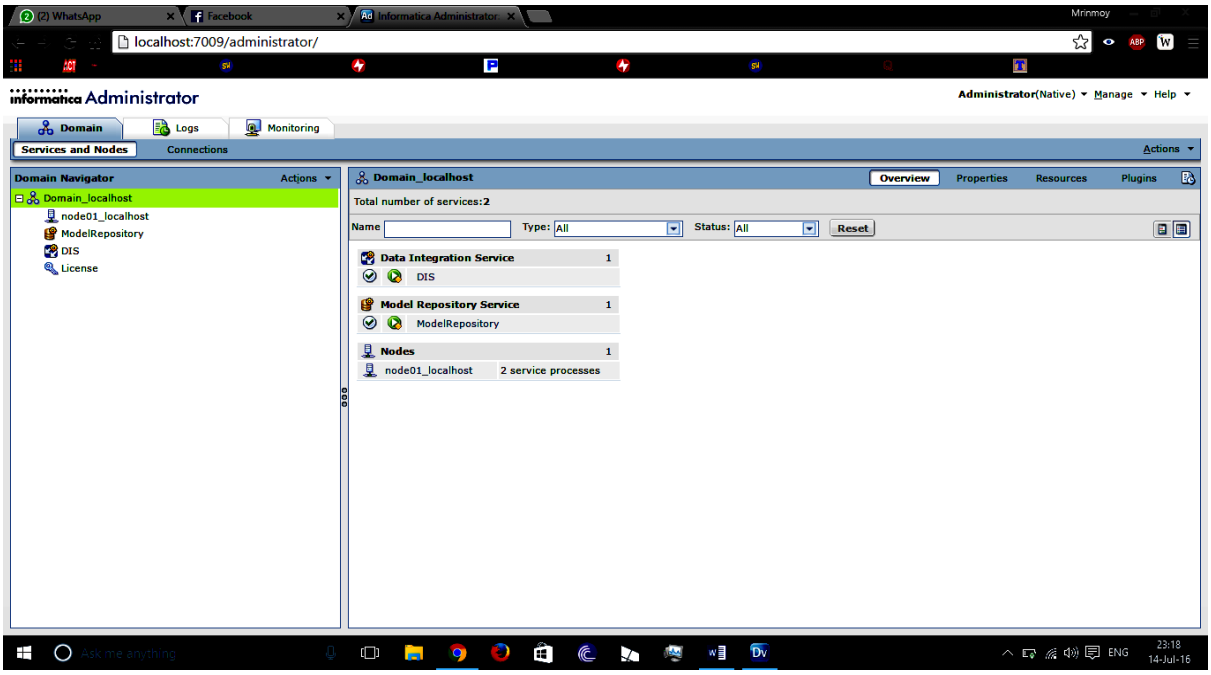


Fig 3.3: Step 3.3

*Step 4:*Once Informatica Developer loads up, connect to Model Repository using the credentials as specified in the server and you are good to go.

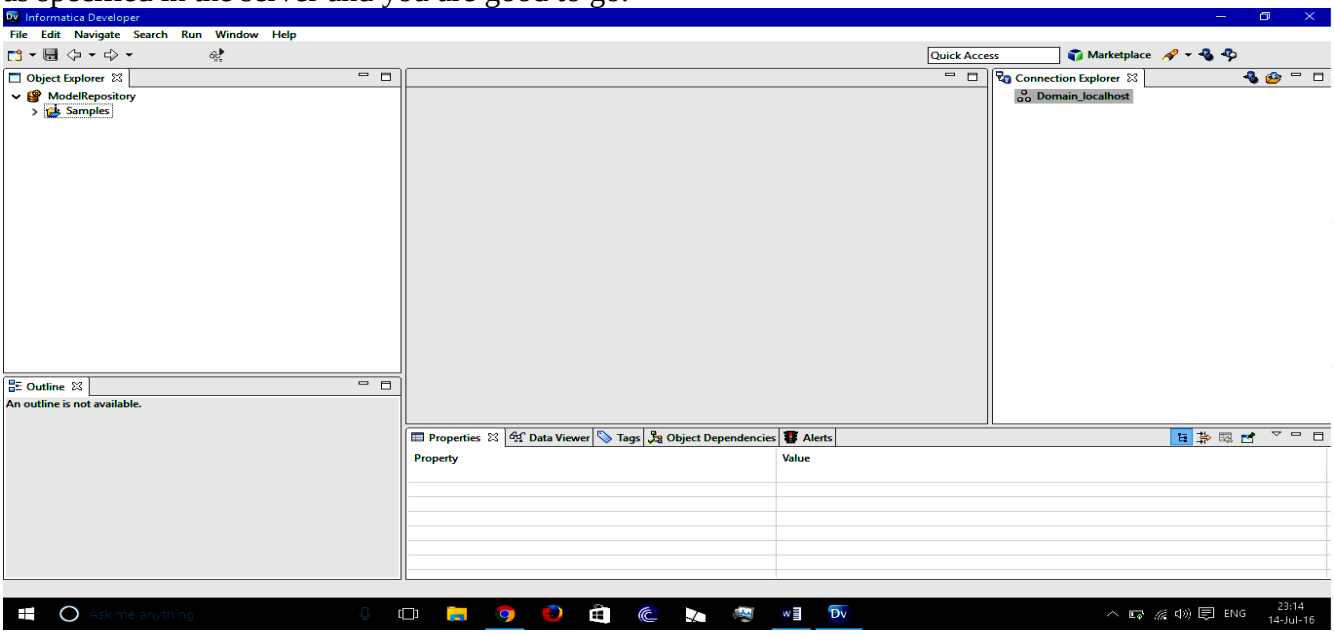


Fig 3.4: Step 3.4

IV. Connecting to Data Source

*Step 1:*Click on the **Create Connection** Icon in the **Connection Explorer.** In the dialog window that pops up, set a unique name and set the type as **ODBC**

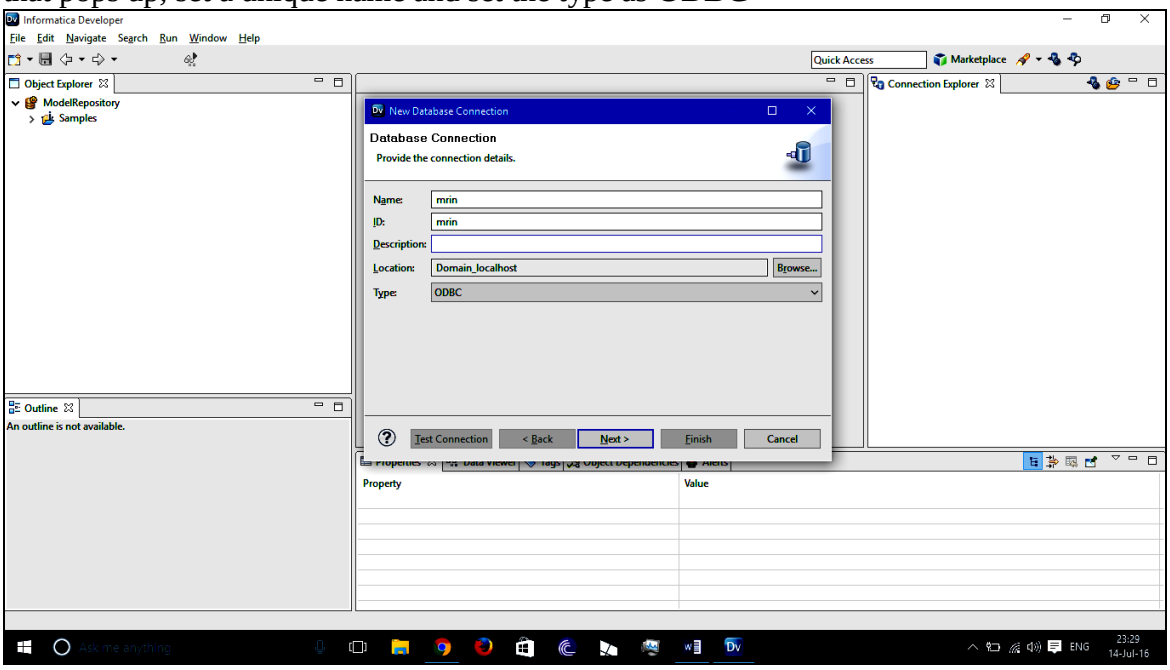


Fig 4.1: Step 4.1

Step 2: In the following dialog box that appears, set the **Username** and **Password** of your database. Set the **Connection String** to the name given in the DSN. This will connect the Tool using ODBC Drivers to your Oracle Database. Click on **Test Connection** to verify connection status.

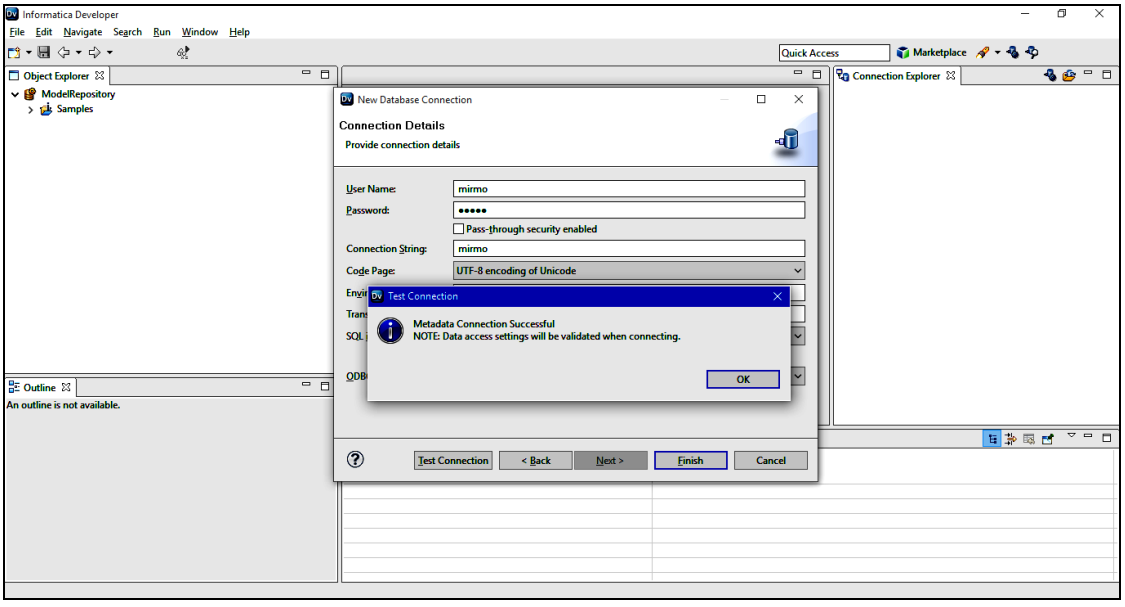


Fig 4.2: Step 4.2

Step 3: If the Connection Test is successful, click on **Finish**. You can now start using the resources that are available in the database.
Note: Unless there is a necessity to connect to another database, the same connection can be reused. Hence these steps are required to be performed only once.

Informatica Tool

Week-2:

1. Build a Data Warehouse to perform filter transformation for the employee database.
2. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.

Filter transformation

• Active Transformation

• Used to filter the data or reduce the amount of data.

• Single condition can be applied at a time.

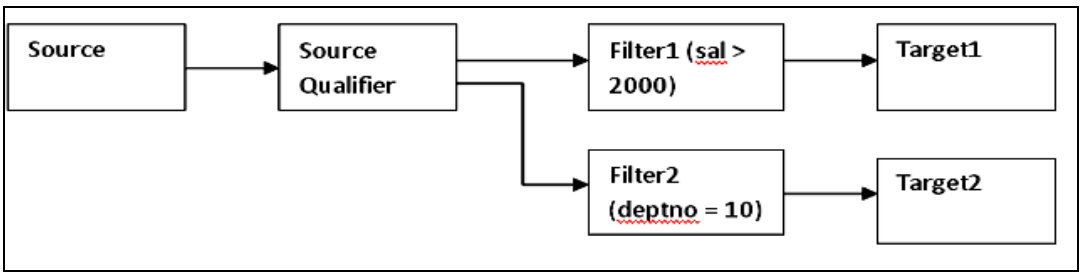
• Filter transformation accepts only two types of data - Boolean (True or False)

• Filter accepts condition satisfied data.

• The false data is rejected or dropped to the bad files. **(Drawback)**

Scenario 1: More than on targets.

We can't combine multiple conditions in one filter



Scenario 2: Only one target.

Here we can combine multiple conditions in one filter using logical operators (AND, OR)

Steps to use Filter Transformation in Informatica

Define the Source (EMP)

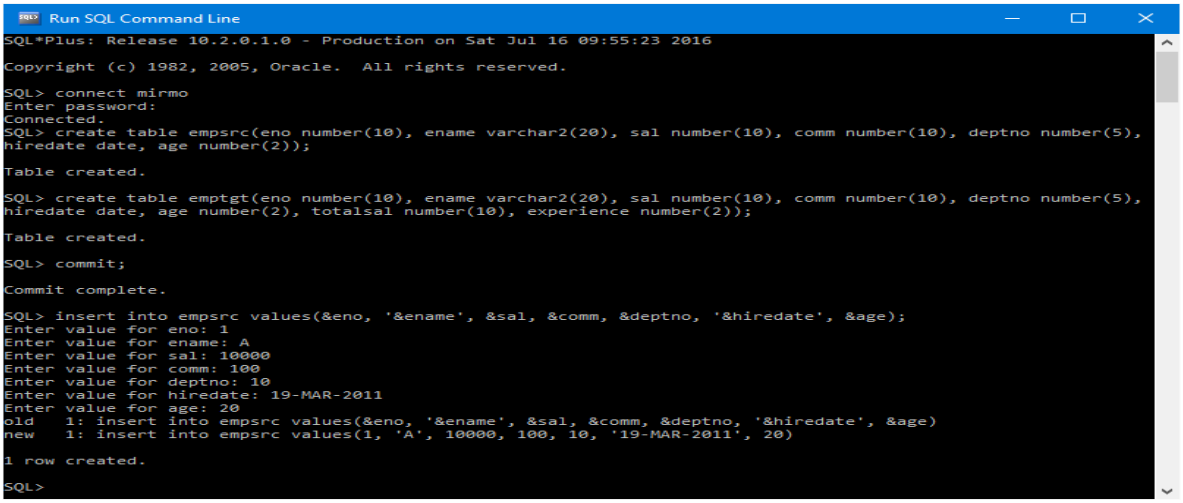
Define the Target (DIM_EMP)

Aim: To apply Filter Transformation to a Data Set and filter out tuples matching a certain criterion.

Theory: The filter transformation takes one table as an input and writes into another table all the values that match a certain criterion. The tuples which do not meet the criteria are left as they are.

Procedure:

Step 1: Create two tables. The first table is the source table for all the employee information and populate it with values and the second table is an empty table which will be the target table for the filter transformation. Commit when done.



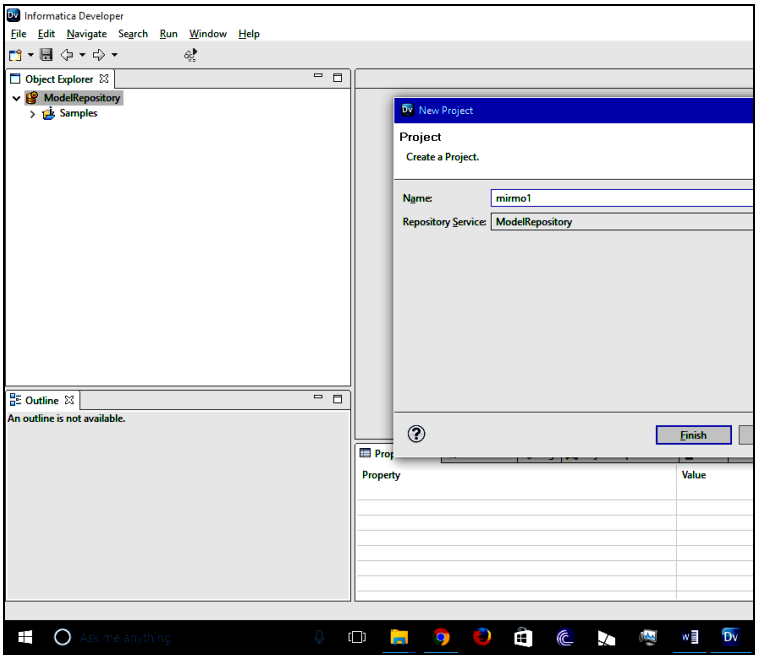
Step 1

Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.

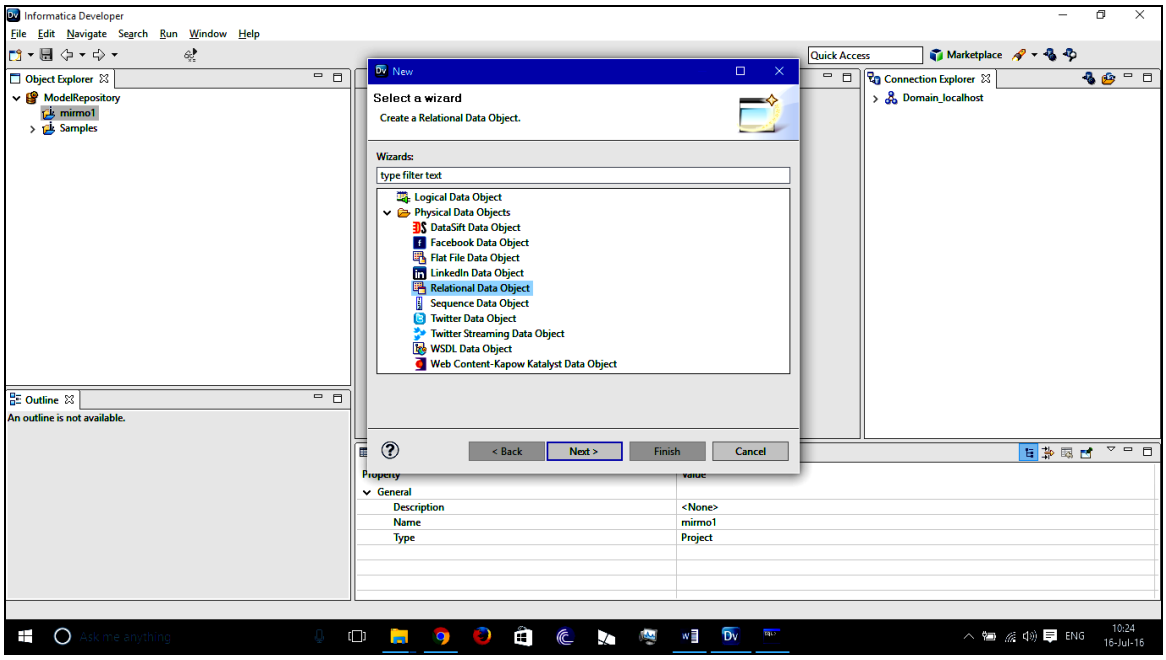
Step 3: Double click on Model Repository under **Object Explorer** to connect it to the Server and also on the DSN under Domain_localhost tab of the **Connection Explorer** to establish connection to the Database.

Step 4: Right click on **Model Repository**. Select **New ➔Project**. Enter the project name (mirmo1) and then click on **Finish**. A new project is created.

Step 5: Right click on mirmo1, select **New ➔ Data Object.** From the dialog box that pops up, select Relational Data Object because we are working with Relational Data (Oracle DB). Then click **Next.**

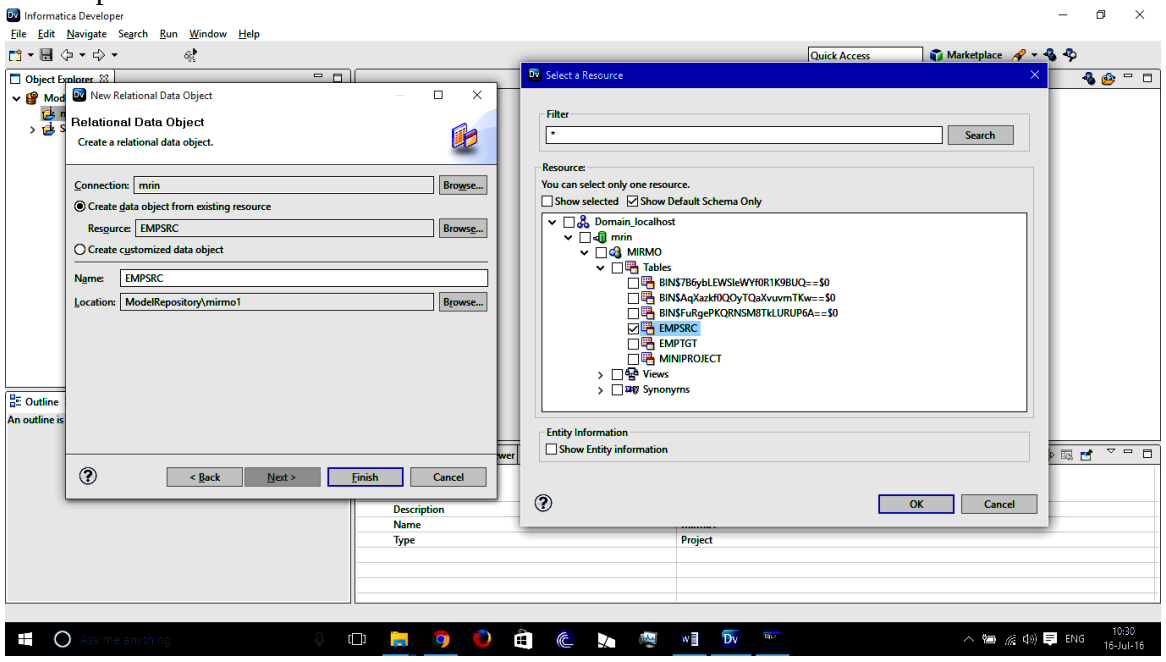


Step 4



Step 5

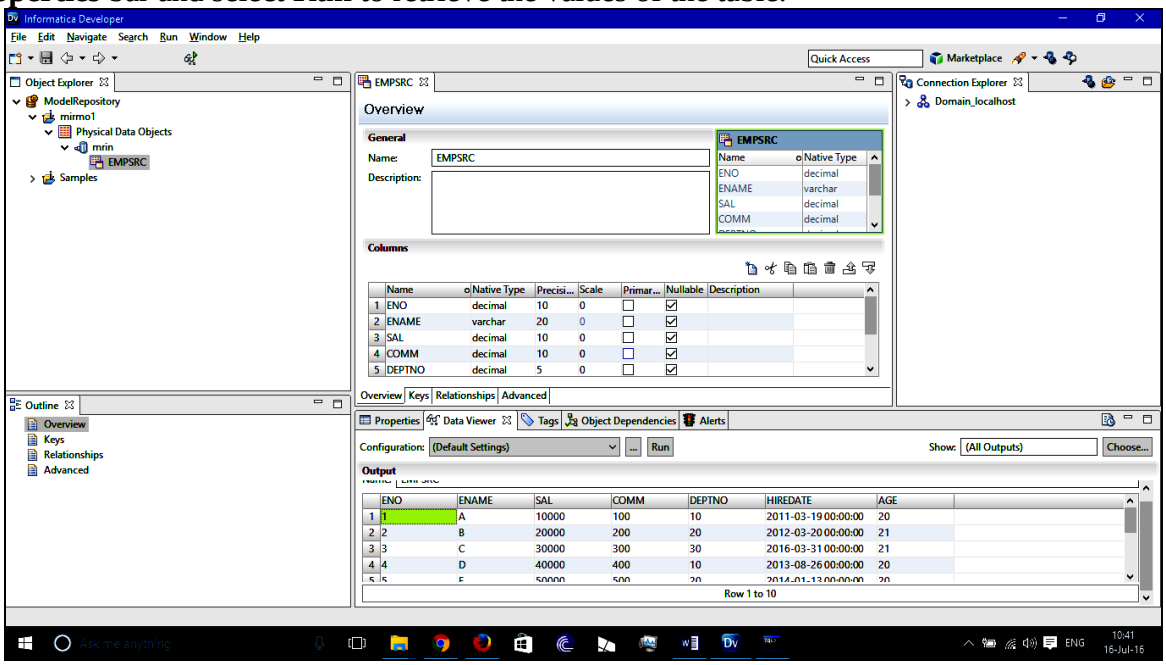
Step 6: In the following dialog box that appears, set the **Connection** to mrin (created earlier). Make sure that **Create data object from existing source** is selected and browse to the table you want to import. Click on **OK** and then click on **Finish.**



Step 6

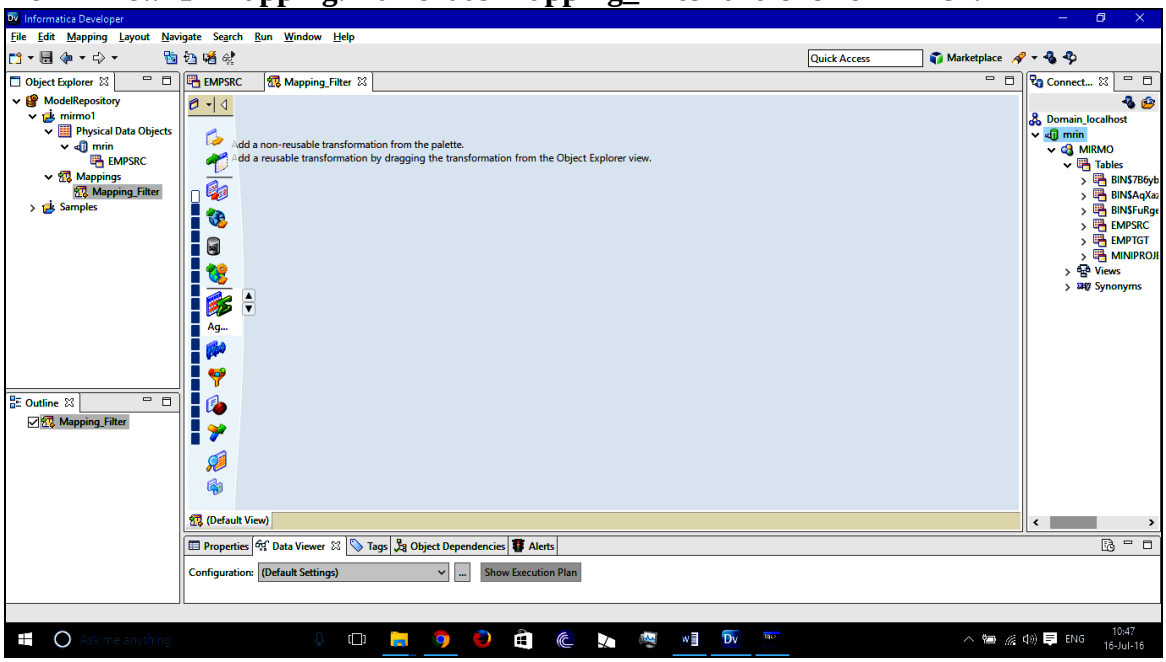
Step 7: If you want to import any more tables, right click on **Physical Data Objects➔New ➔ Physical Data Object** under mirmo1 and go through step 5 and 6 to add tables.

Step 8: Select EMPSRC table in the Object Explorer. Click on the **Data Viewer** tab on the **Properties** bar and select **Run** to retrieve the values of the table.



Step 8

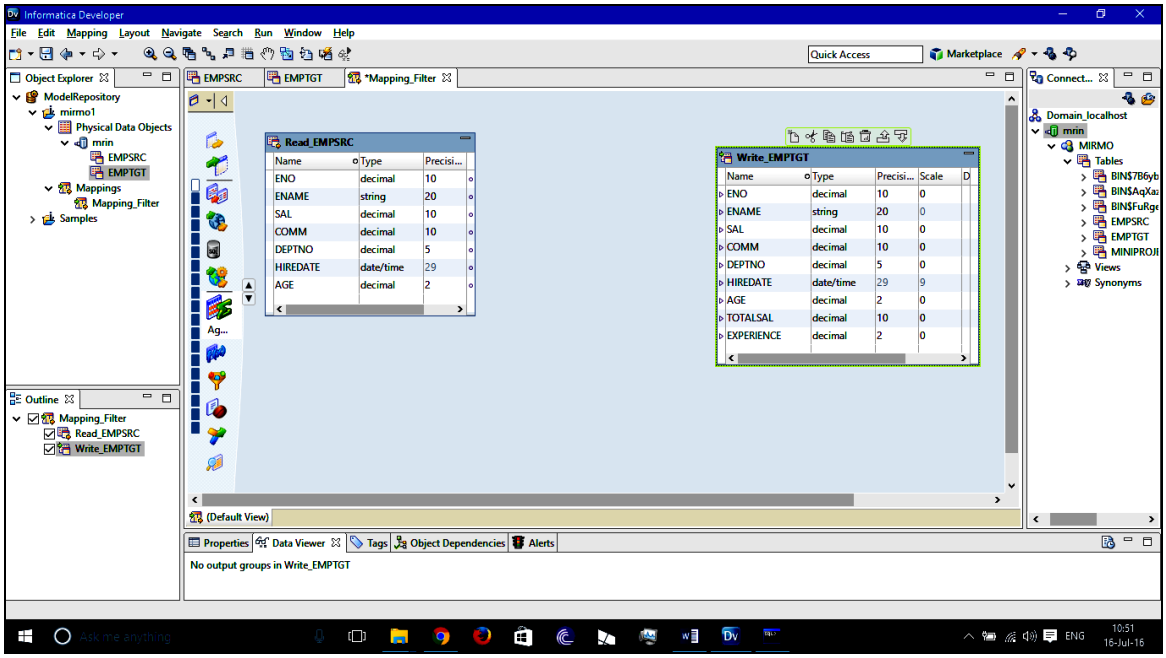
Step 9: This step is to create a new mapping for performing transformations. Right Click on mirmo1 ➔**New** ➔ **Mapping.** Name it as **Mapping_Filter** and click on **Finish.**



Step 9

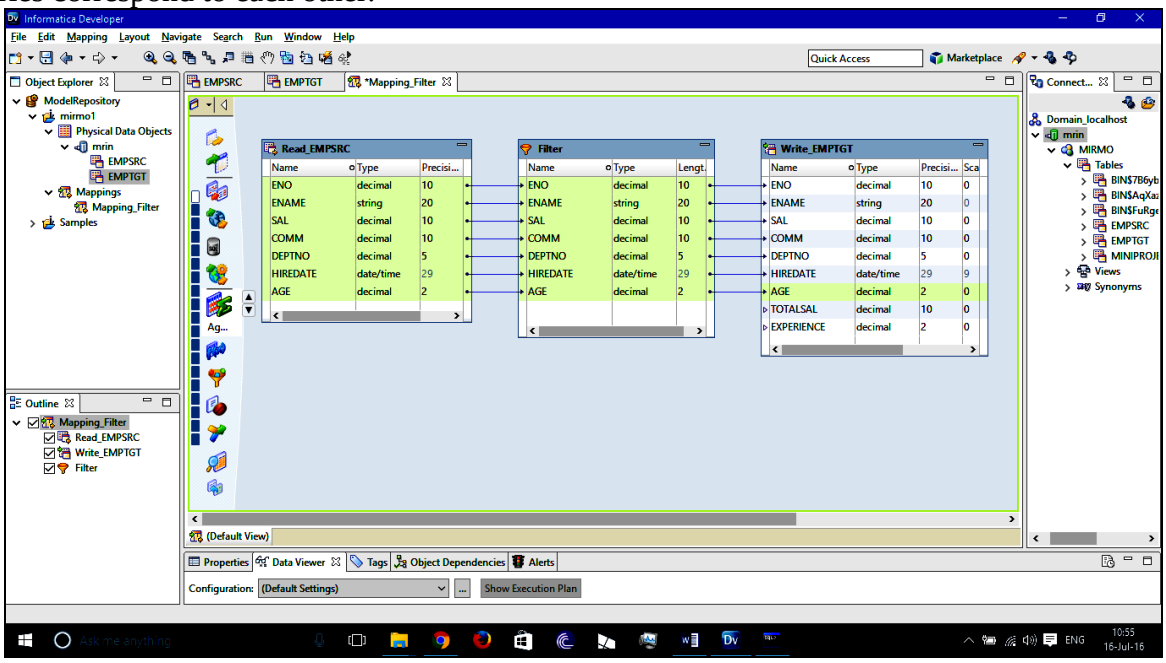
Step 10: Drag EMPSRC table from Object Explorer into the Mapping_Filter tab. Set **Physical Data Object Access** to **Read** as we use this table only to take inputs. Then drag EMPTGT table

and set Physical **Data Object Access** to **Write** as we use this table only to store the output of the filter transformation.



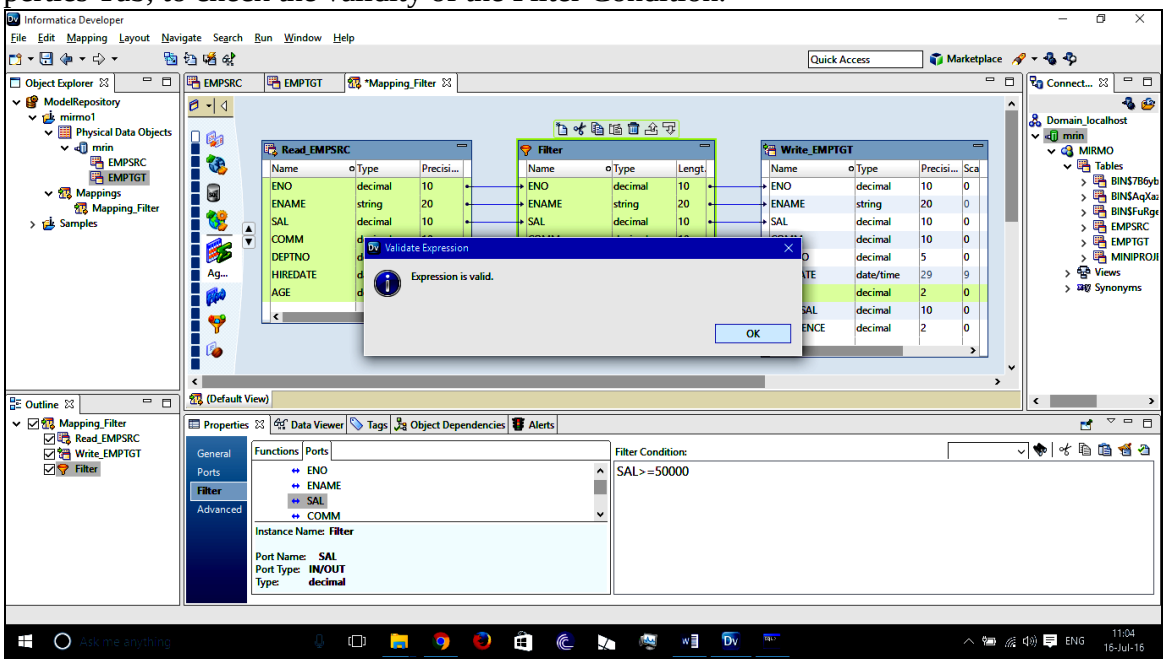
Step 10

Step 11: Select **Filter** transformation from the Palette and drag it onto the Mapping Tab. Select all entries from **Read_EMPSRC** and drop it onto the Filter. Now select all entries of **Filter** and drop it onto **Write_EMPTGT.** This creates a connection between the three. Make sure the entries correspond to each other.



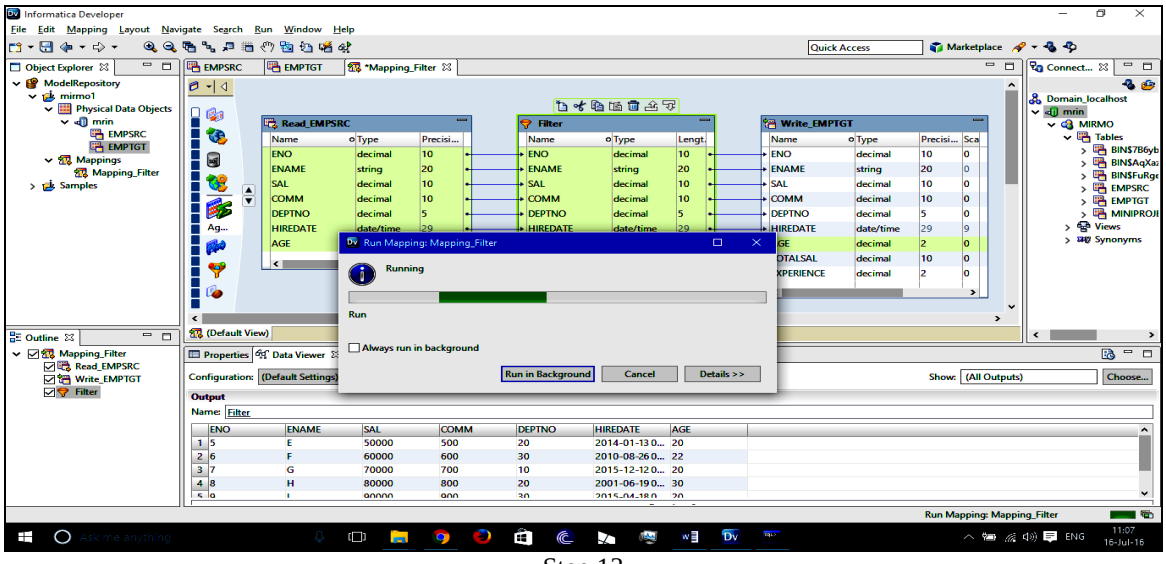
Step 11

Step 12: Make sure that **Filter** is selected. Then navigate to **Properties ➔ Filter** and specify the condition in the **Filter Condition** box. Click on **Validate Expression** at the right corner of the Properties Tab, to check the validity of the Filter Condition.



Step 12

Step 13: Click on the **Data Viewer** Tab of the Properties Explorer and select Run. If the Transformation is successful, then all entries matching the condition are displayed. Then click on **Run** in the menu bar and select **Run Mapping.**

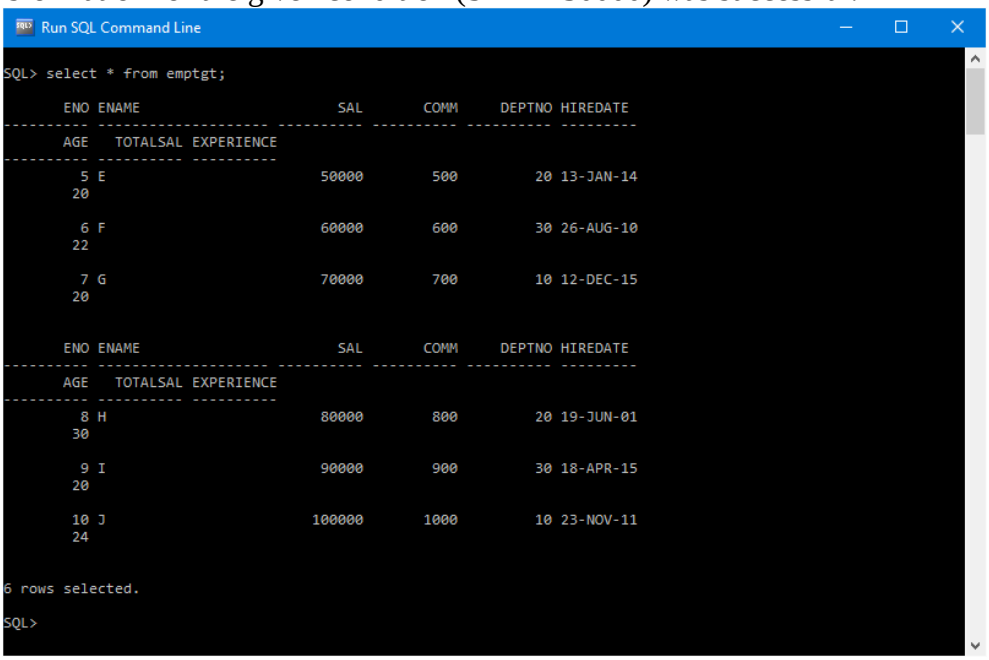


Step 13

Step 14: Verify the output of the Filter Transformation in SQL Command Prompt.

Output:

Filter Transformation for the given condition (SAL>=50000) was successful.



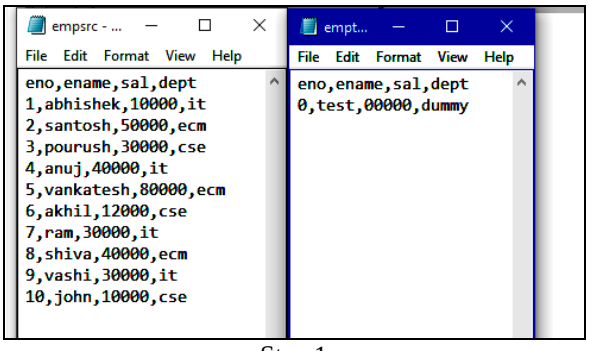
Output

Aim: To apply Filter Transformation to a Data Set and filter out tuples matching a certain criterion.

Theory: The filter transformation takes one table as an input and writes into another table all the values that match a certain criterion. The tuples which do not meet the criteria are left as they are.

Procedure:

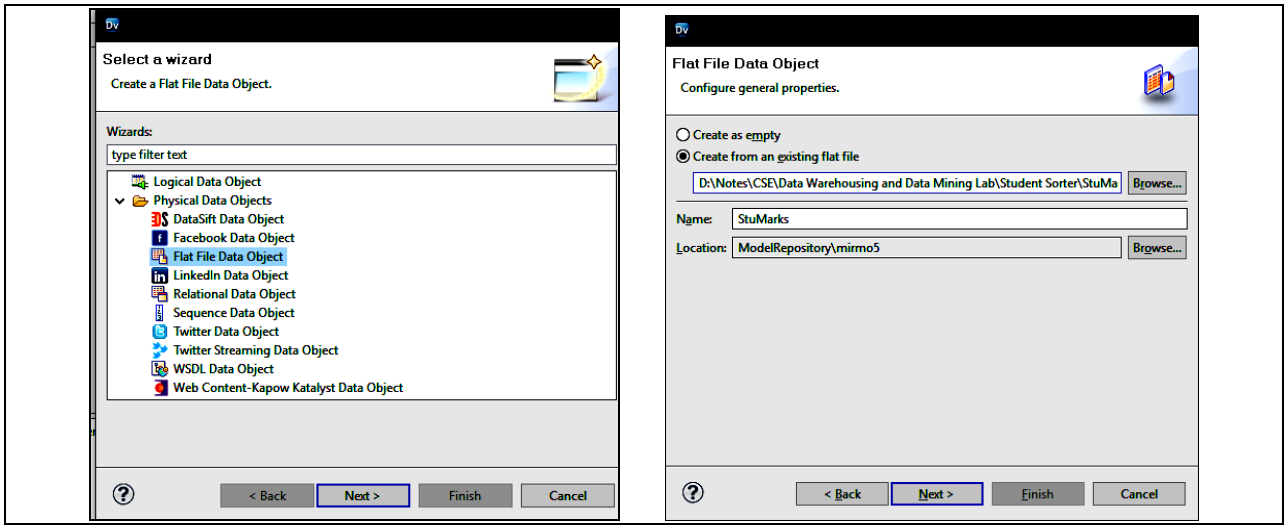
Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ',' as a delimiter.



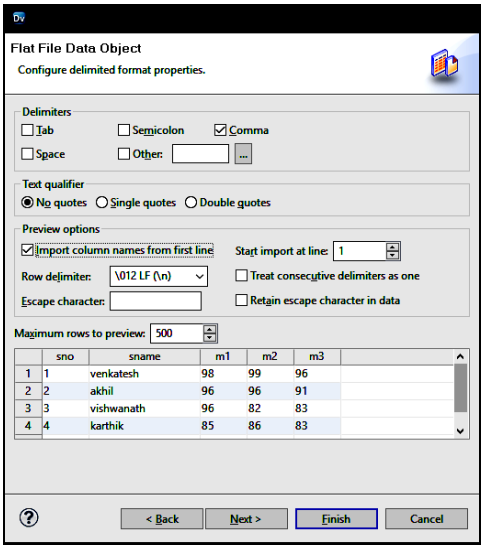
Step 1

Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Import your flat files (both source and target files) into **mirmo1** project. Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.

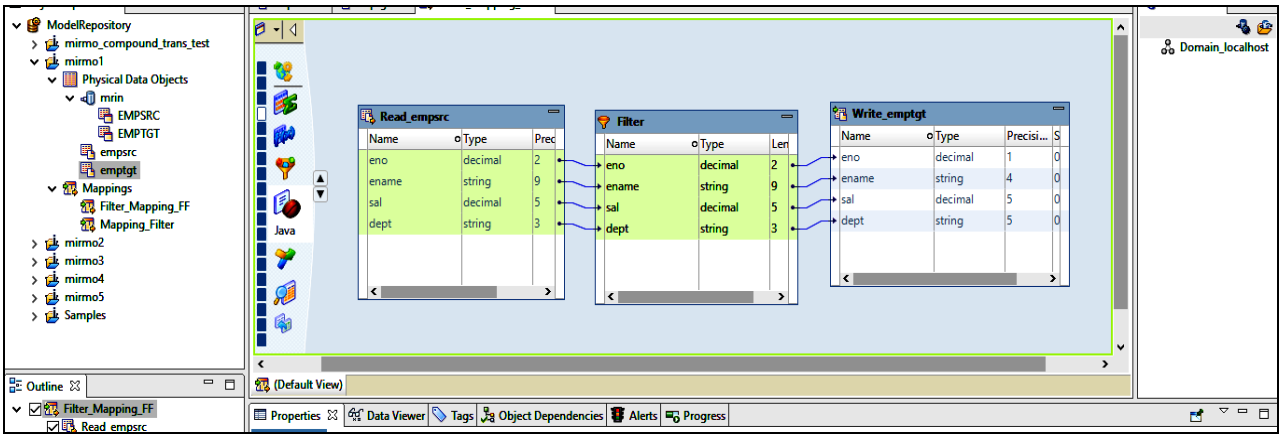


Step 3



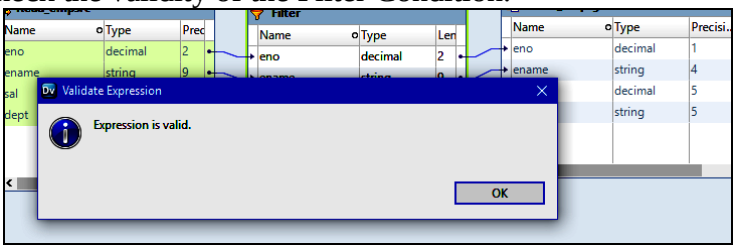
Step 3

Step 4: Once the files have been imported, create a new mapping and name it **Filter_Mapping**.
Step 5: Import the source file with read mode and target file as write mode and drag and drop the Filter Transformation from the Palette into the workspace. Give the necessary connections.



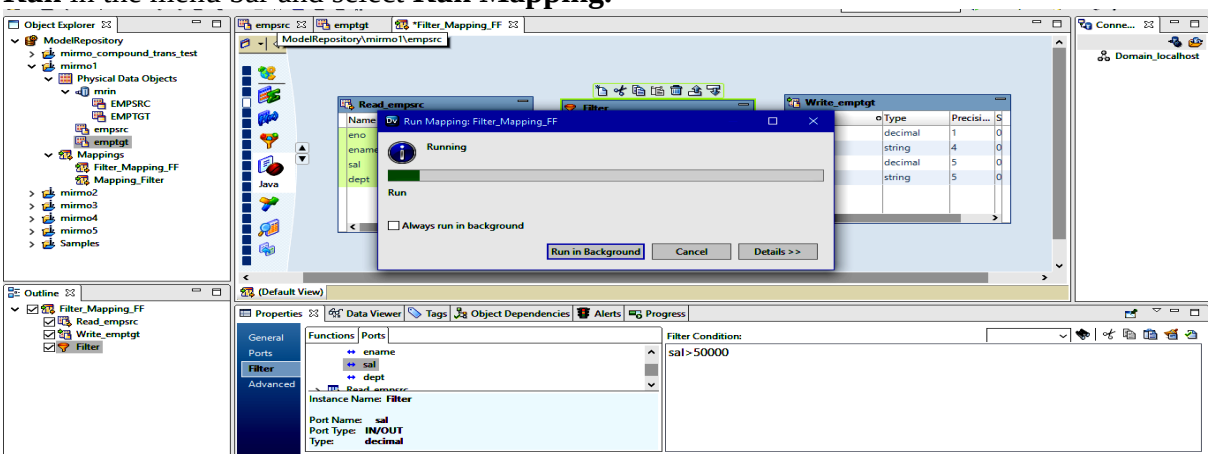
Step 5

Step 6: Make sure that **Filter** is selected. Then navigate to **Properties ➔ Filter** and specify the condition in the **Filter Condition** box. Click on **Validate Expression** at the right corner of the Properties Tab, to check the validity of the Filter Condition.



Step 6

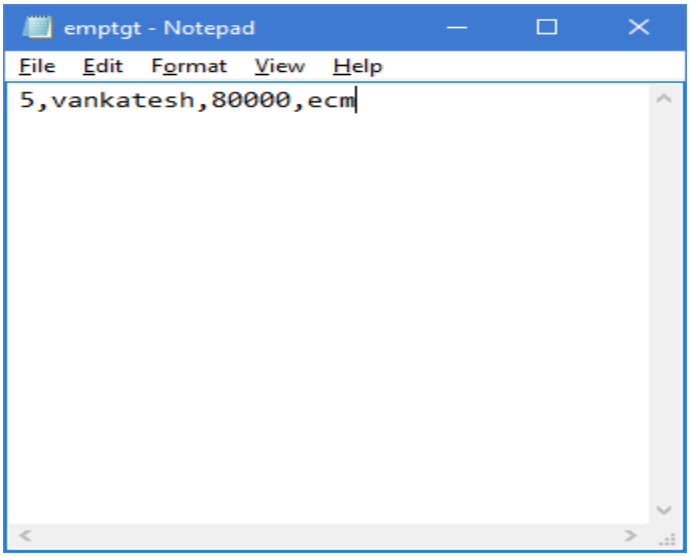
Step 7: Click on the **Data Viewer** Tab of the Properties Explorer and select Run. If the Transformation is successful, then all entries matching the condition are displayed. Then click on **Run** in the menu bar and select **Run Mapping.**



Step 7

Step 8: Make sure that the connections are intact. Now Run the Mapping. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Output

Filter Transformation for the given condition (SAL>50000) was successful.

EXPRESSION TRANSFORMATION:

Aim: To apply an Expression Transformation on a Data Set to evaluate two columns into an output column in the target table.

Theory: Expression transformation is a connected, passive transformation used to calculate values on a single row. Examples of calculations are concatenating the first and last name, adjusting the employee salaries, converting strings to date etc. Expression transformation can also be used to test conditional statements before passing the data to other transformations.

Procedure:

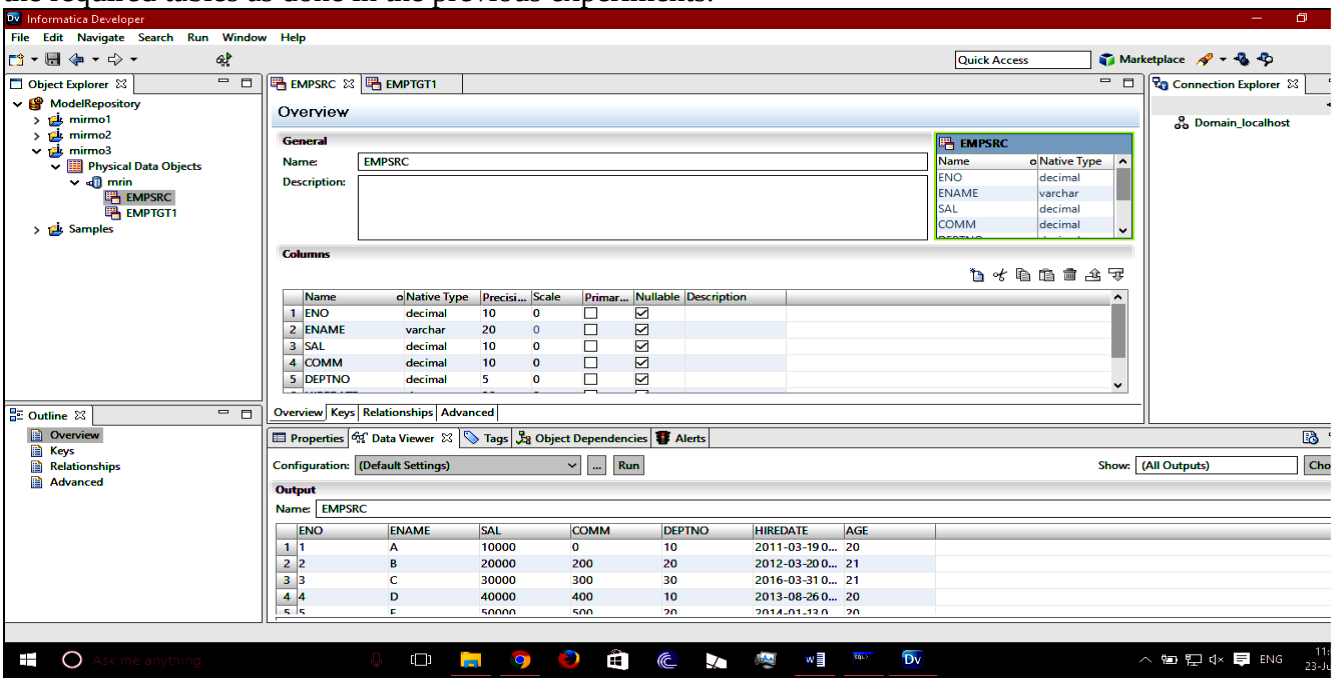
*Step 1:*Create the required tables and populate the source table with values. Make sure that the fields you perform the Expression Transformation on, are of the type Integer/Number as we will be adding two columns and storing the value in another column in the target table.

```
SQL> desc source;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ENO                                                NUMBER(10)
 ENAME                                              VARCHAR2(20)
 SAL                                                NUMBER(10)
 COMM                                               NUMBER(10)
 DEPTNO                                             NUMBER(5)
 HIREDATE                                           DATE
 AGE                                                NUMBER(2)

SQL> desc emptgt1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ENO                                                NUMBER(10)
 ENAME                                              VARCHAR2(20)
 SAL                                                NUMBER(10)
 COMM                                               NUMBER(10)
 DEPTNO                                             NUMBER(5)
 HIREDATE                                           DATE
 AGE                                                NUMBER(2)
 TOTALSAL                                           NUMBER(10)
 EXPERIENCE                                         NUMBER(2)
```

Step 1

Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.

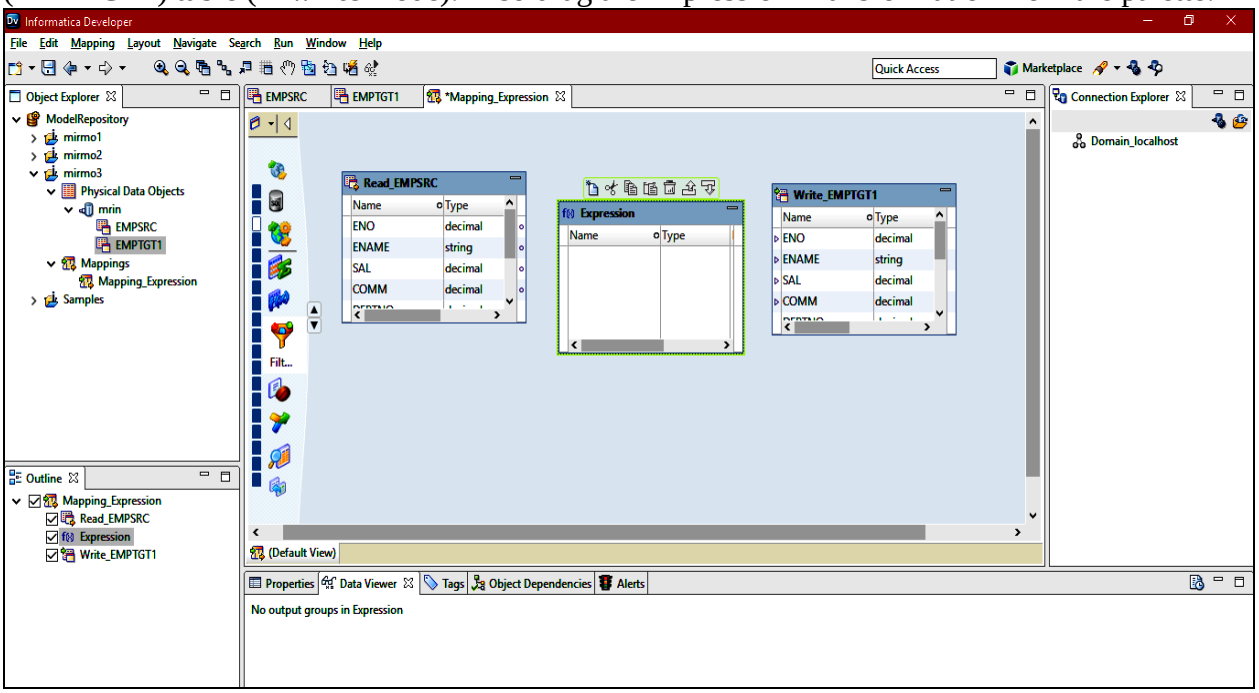
Step 3: Once Informatica Developer has started, create a **new project** (mirmo3) and import all the required tables as done in the previous experiments.



Step 3

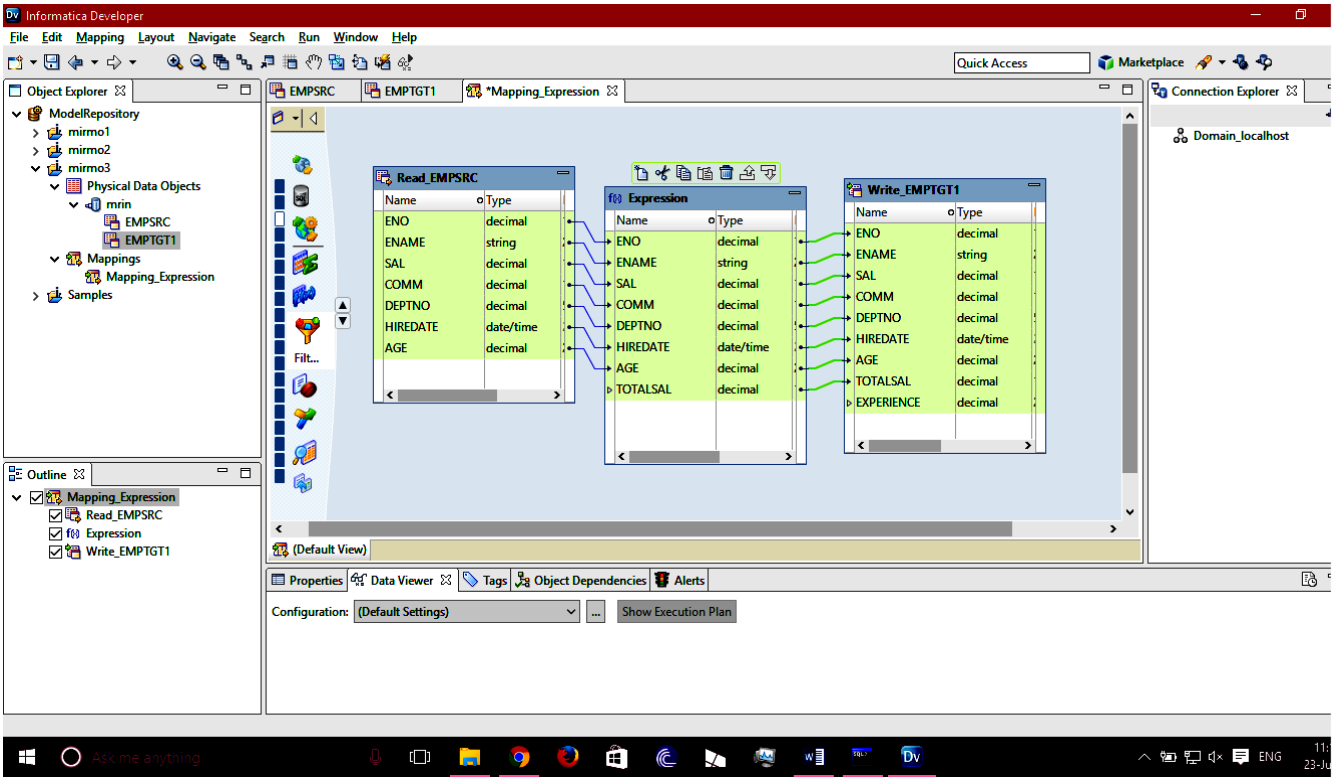
Step 4: Create a new mapping and name it Mapping_Expression.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (EMPTGT1) table (in **write** mode). Also drag the Expression Transformation from the palette.



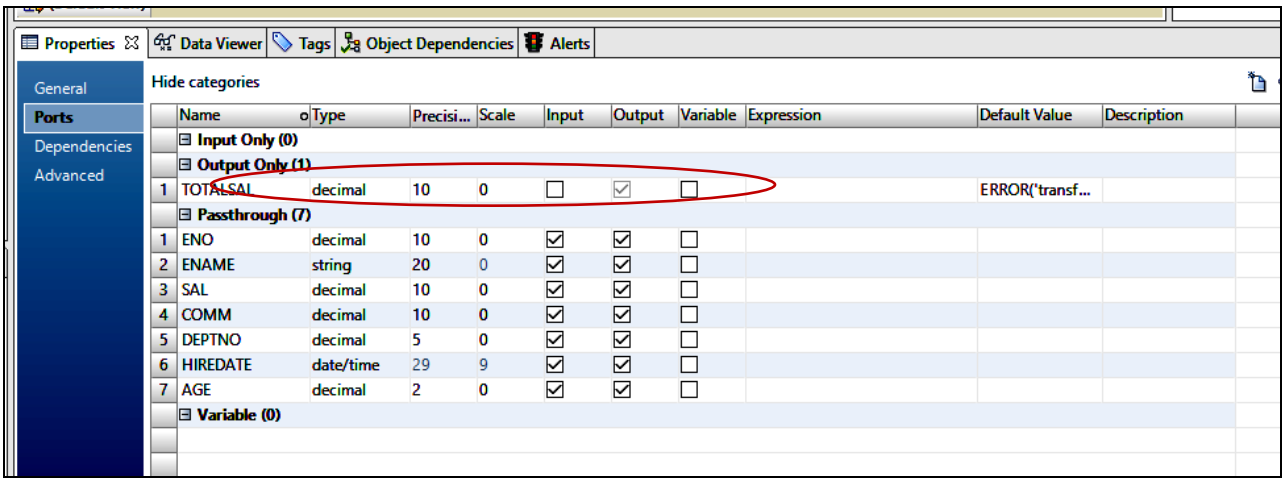
Step 5

Step 6: Drag and drop the values of the target table into the **Expression Transformation**. Delete the **Experience**entry as it is not required for this task. Also link up the input table and the output table via the Expression Transformation appropriately.



Step 6

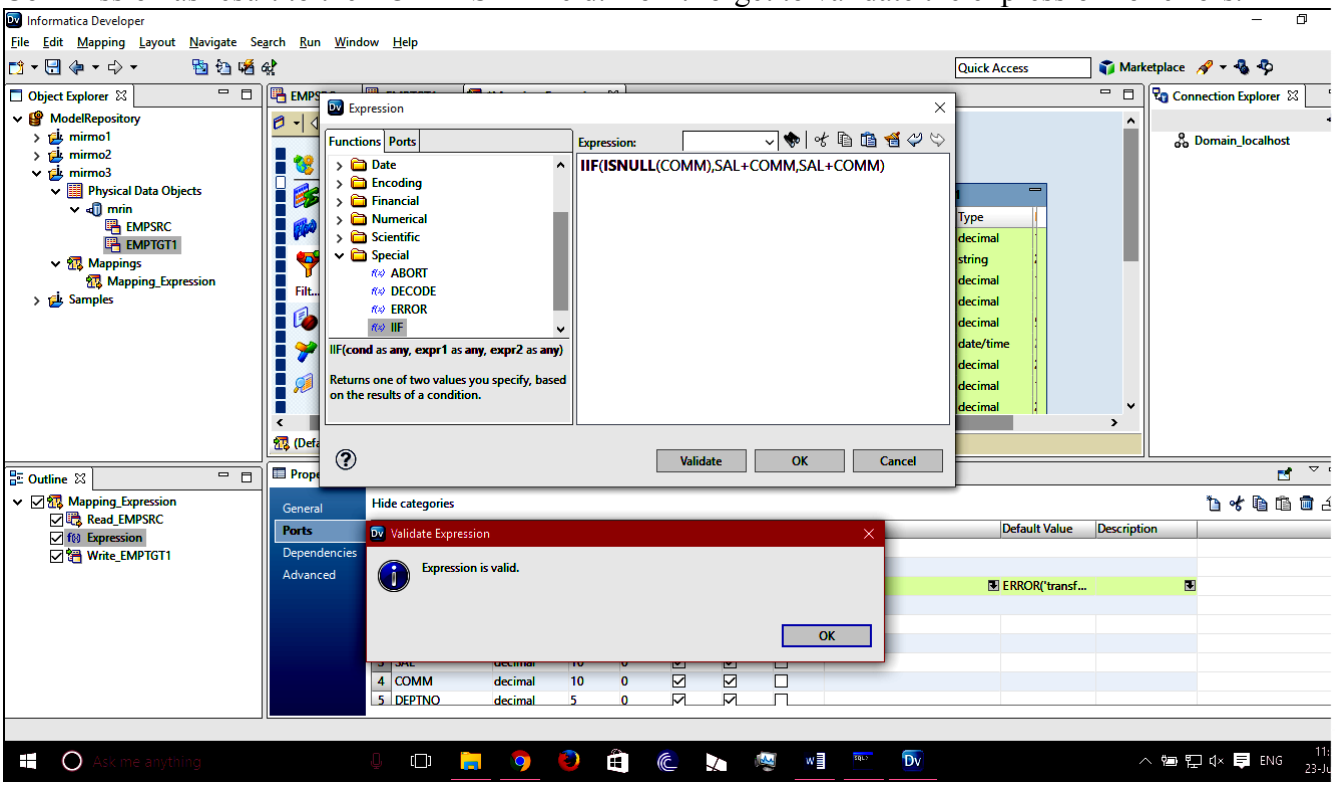
Step 7: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **TOTALSAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



Step 7

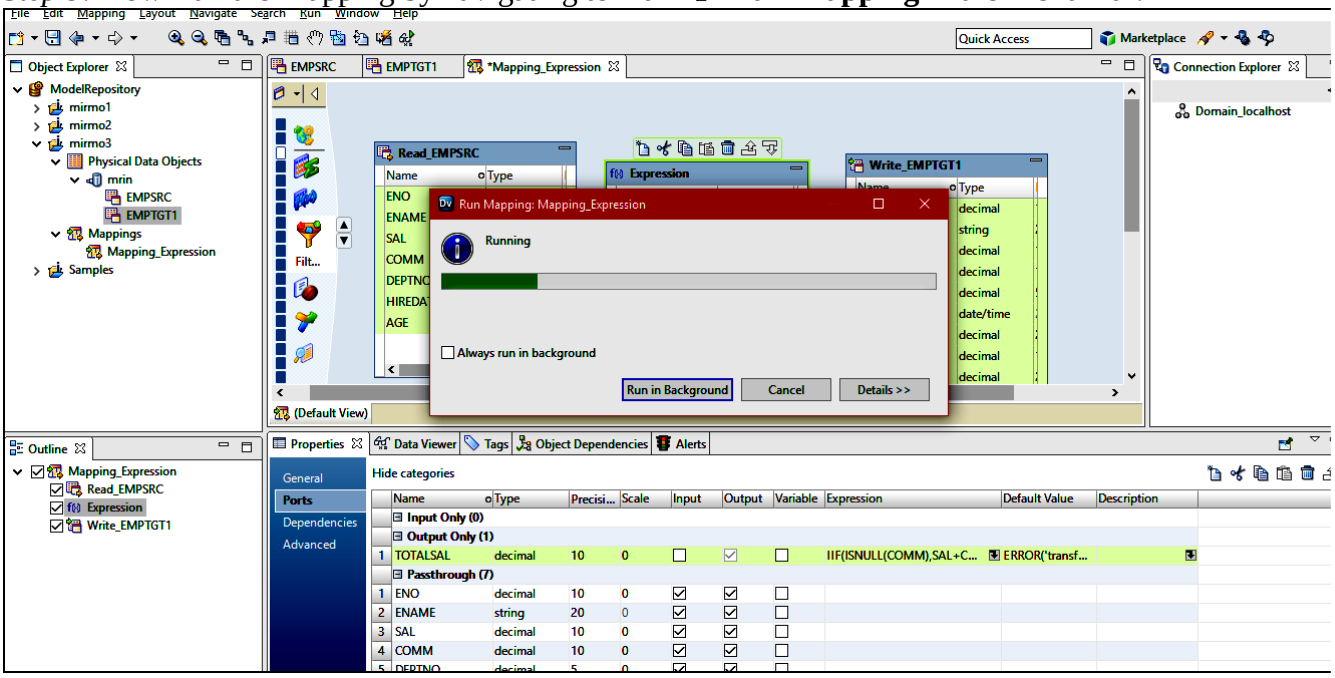
Step 8: In the Expression Section of the entry, click on the small arrow head to open up the **Expression Dialog Box**and enter the following expression:

"**IIF(ISNULL(COMM),SAL+COMM,SAL+COMM)**" This expression checks if the Commission field is NULL, then returns only salary as the result, else returns Salary + Commission as result to the TOTALSAL field. Don't forget to validate the expression for errors.



Step 8

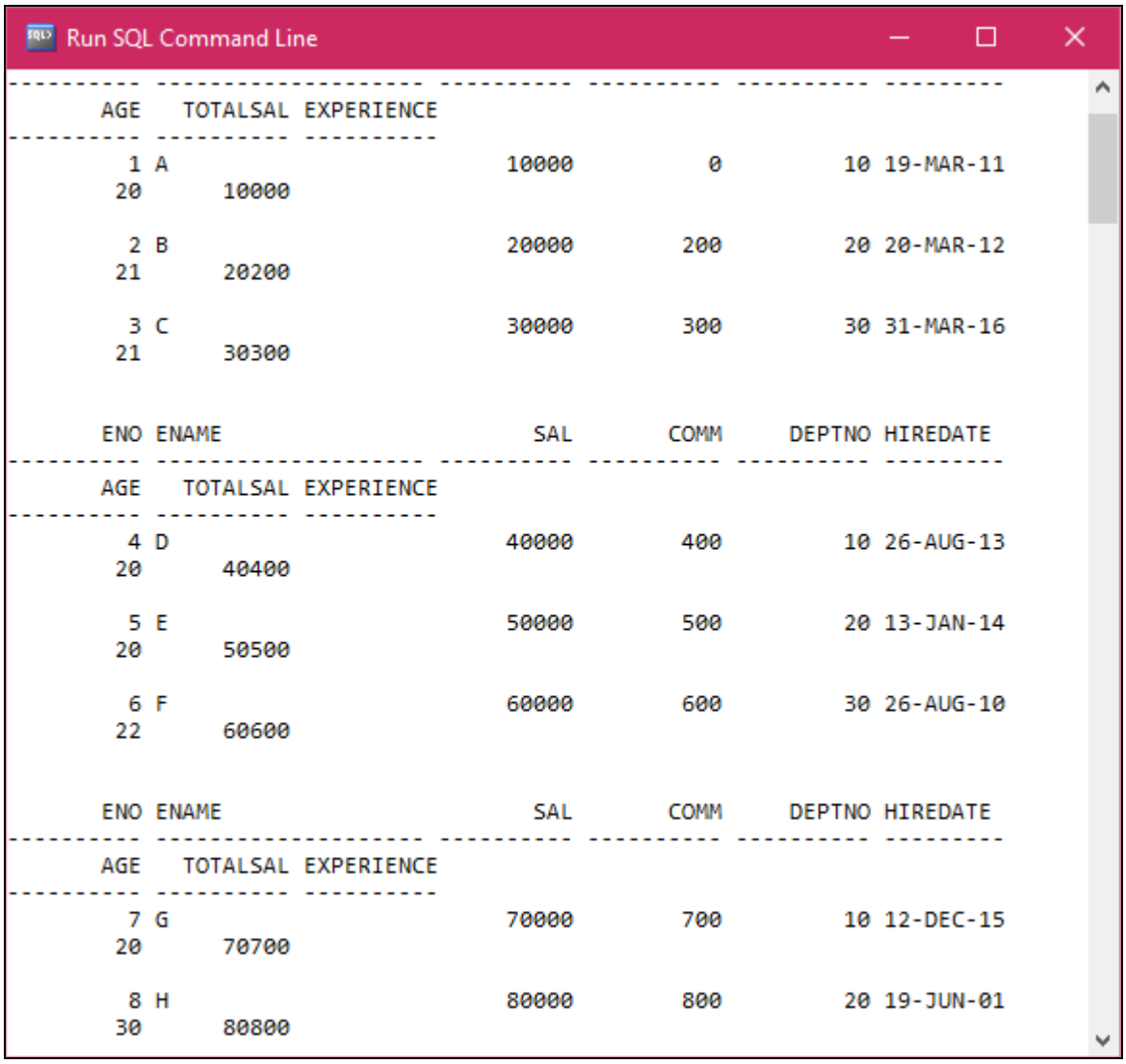
Step 9: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.



Step 9

Step 10: Verify the output of Expression Transformation in the SQL Command Prompt.

Output:



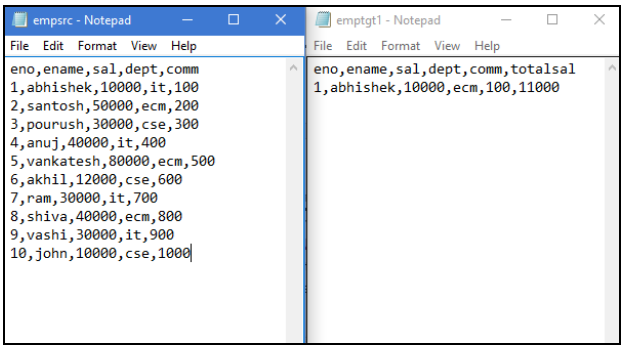
Expression Transformation was successful.

Aim: To apply an Expression Transformation on a flat file to evaluate two columns into an output column in the target table.

Theory: Expression transformation is a connected, passive transformation used to calculate values on a single row. Examples of calculations are concatenating the first and last name, adjusting the employee salaries, converting strings to date etc. Expression transformation can also be used to test conditional statements before passing the data to other transformations.

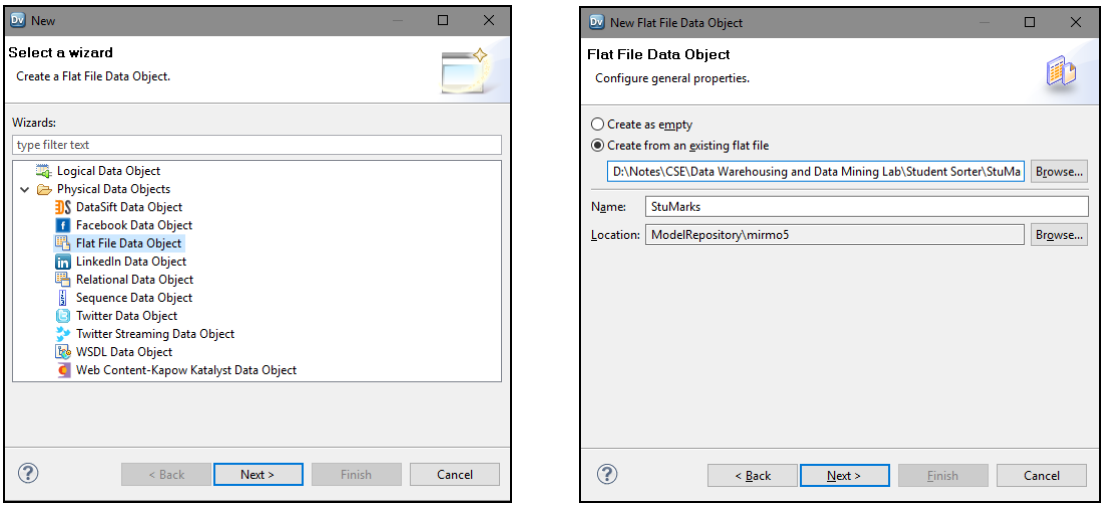
Procedure:

Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ‘,’ as a delimiter.



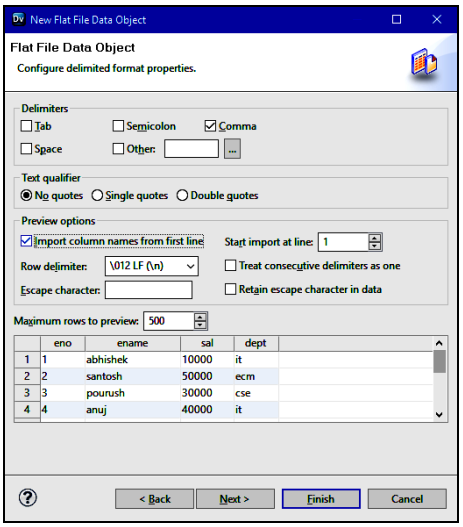
Step 1

Step 2: Start up Informatica Services and Launch Informatica Developer.
Step 3: Import your flat files (both source and target files) into **mirmo2** project.



Step 3

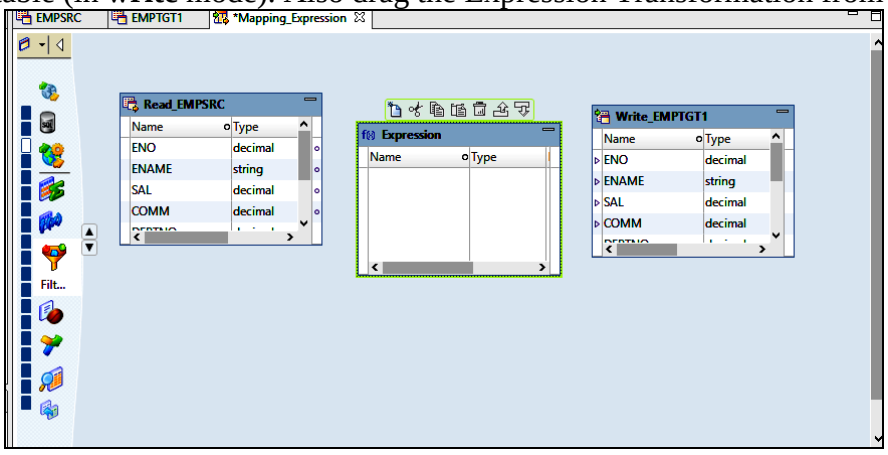
Click on Next and then Next again. In the next dialog box that opens, check the **“Import Column Names from the first line”** checkbox and make sure the delimiter selected is ‘**,**’. Then click on **Finish**.



Step 3

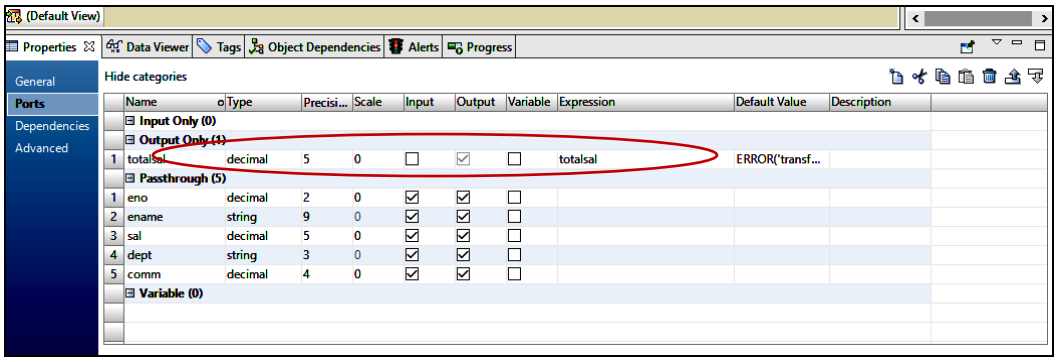
Step 4: Once the files have been imported, create a new mapping and name it **Expression_Mapping**.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (EMPTGT1) table (in **write** mode). Also drag the Expression Transformation from the palette.



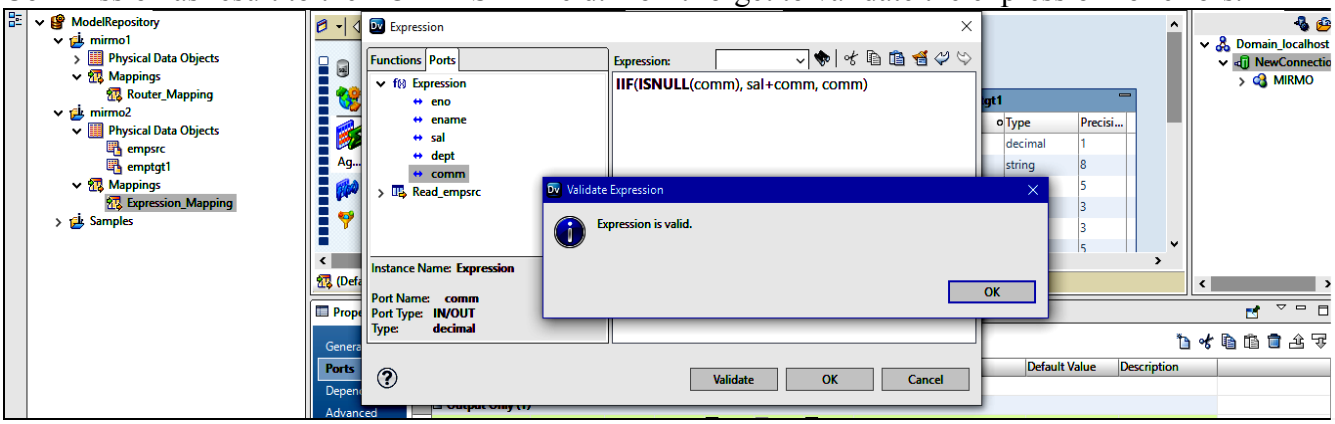
Step 5

Step 6: Drag and drop the values of the target table into the **Expression Transformation**. Also link up the input table and the output table via the Expression Transformation appropriately. Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **TOTALSAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



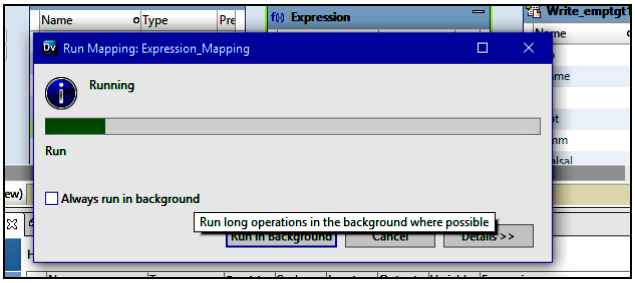
Step 6

Step 7: In the Expression Section of the entry, click on the small arrow head to open up the **Expression Dialog Box**and enter the following expression: "**IIF(ISNULL(COMM),SAL+COMM,SAL+COMM)"** This expression checks if the Commission field is NULL, then returns only salary as the result, else returns Salary + Commission as result to the TOTALSAL field. Don't forget to validate the expression for errors.



Step 7

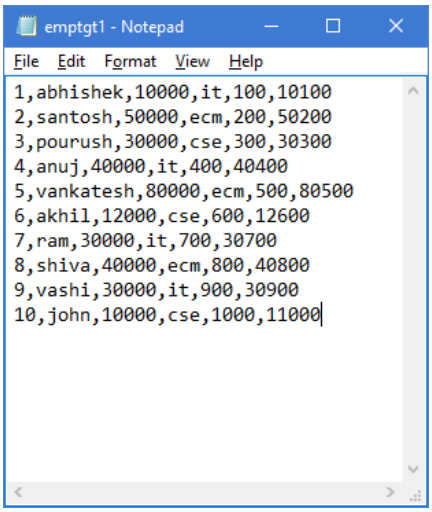
Step 8: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.



Step 8

Step 9: Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target files. Open them up to see the output.

Output:



xpression Transformation was successful.

WEEK-3:

1. Using Aggregator transformation display the average salary of employees in each departments.
2. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.

Aggregator Transformation in Informatica

Aggregator is an active transformation used to calculate group values.

Ports:

· Input

· Output

· Expression

· Variable

· Group By

Properties:

· Data Cache

· Index cache

· Sorted Input

Steps:

1. Define Source

2. Define Target

Write the Target requirements and Generate/Execute SQL.

AGG_FACT

· DEPTNO

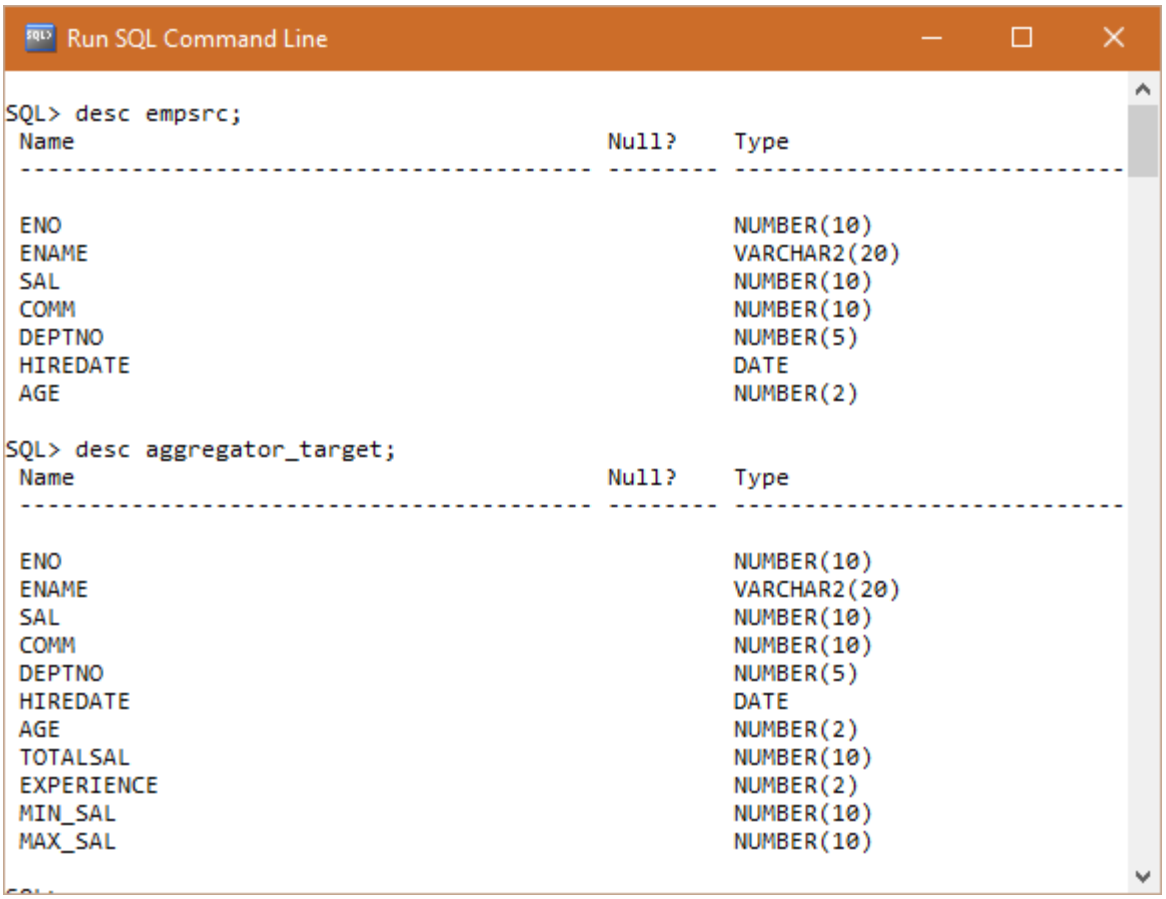
· TOT_SAL

· MAX_SAL

· MIN_SAL

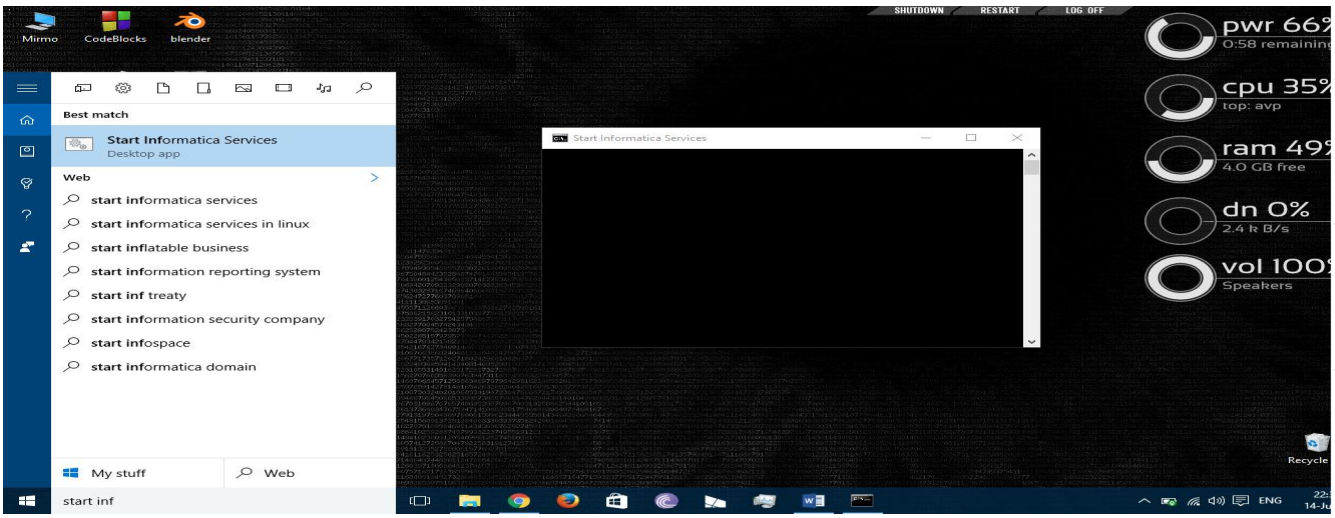
· AVG_SAL
· COUNT_EMP

Aim: To apply an Aggregator Transformation on a Data Set to calculate Minimum and Maximum Salary in the table.
Theory: Aggregator transformation is an active transformation used to perform calculations such as sums, averages, counts on groups of data. The integration service stores the data group and row data in aggregate cache. The Aggregator Transformation provides more advantages than the SQL, you can use conditional clauses to filter rows.
Procedure:
*Step 1:*Create the required tables and populate the source table with values. Add two extra columns namely min_sal and max_sal to the target table (aggregator_target) to store the output values.



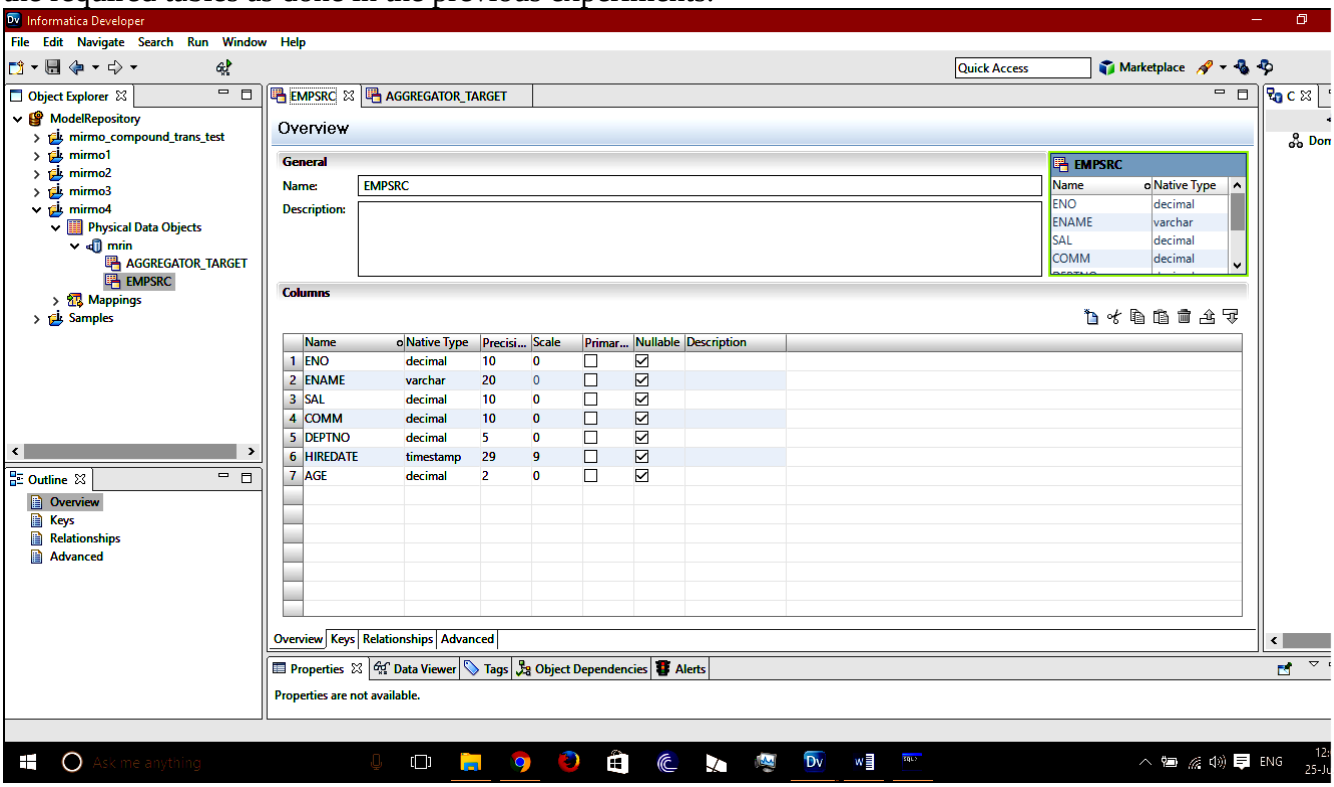
Step 1

Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.



Step 2

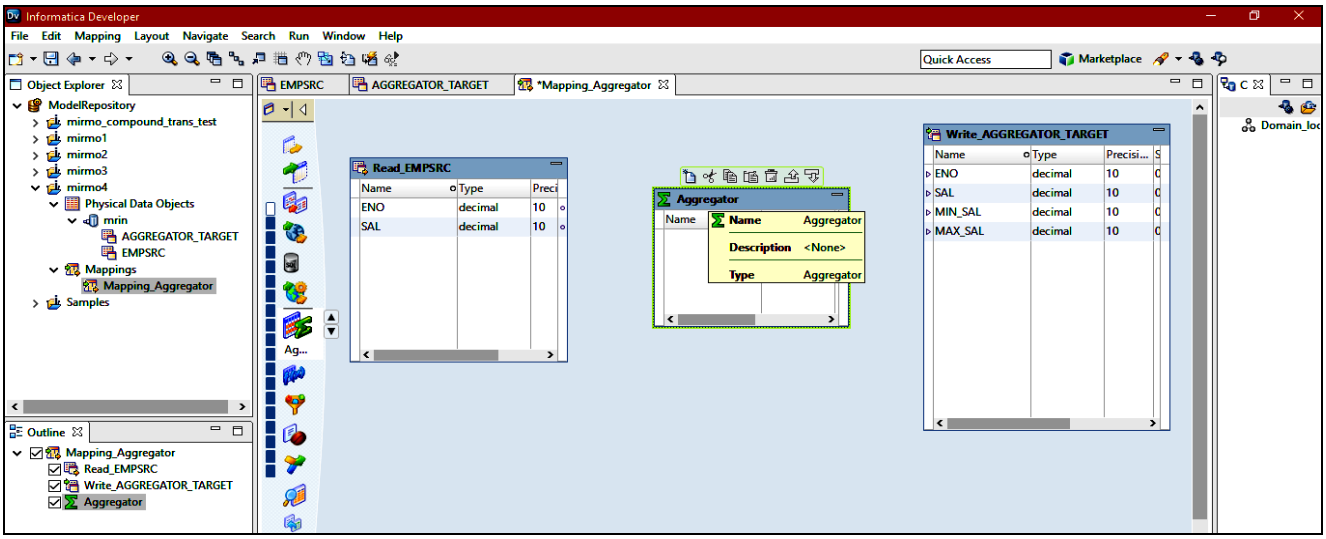
Step 3: Once Informatica Developer has started, create a **new project** (mirmo4) and import all the required tables as done in the previous experiments.



Step 3

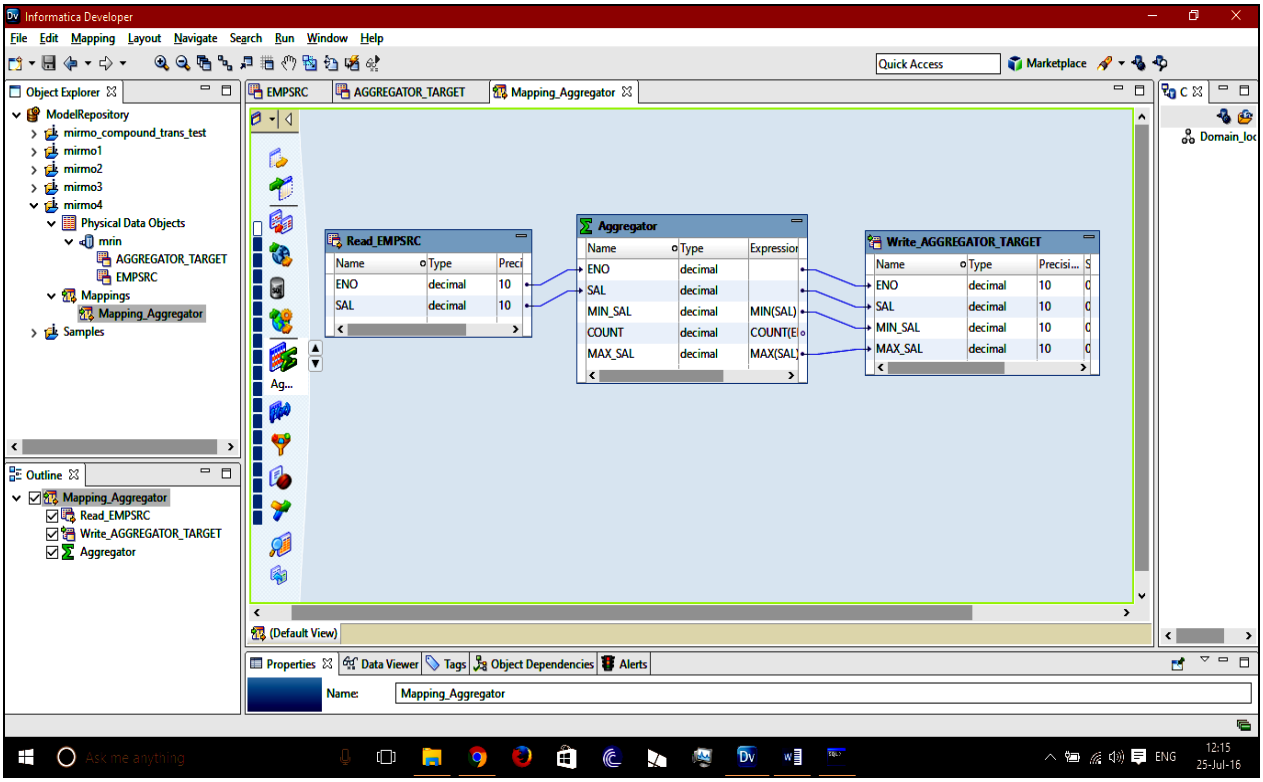
Step 4: Create a new mapping and name it Mapping_Aggregator

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (AGGREGATOR_TARGET) table (in **write** mode). Delete all the non-required entries from the tables. Also drag the Aggregator Transformation from the palette.



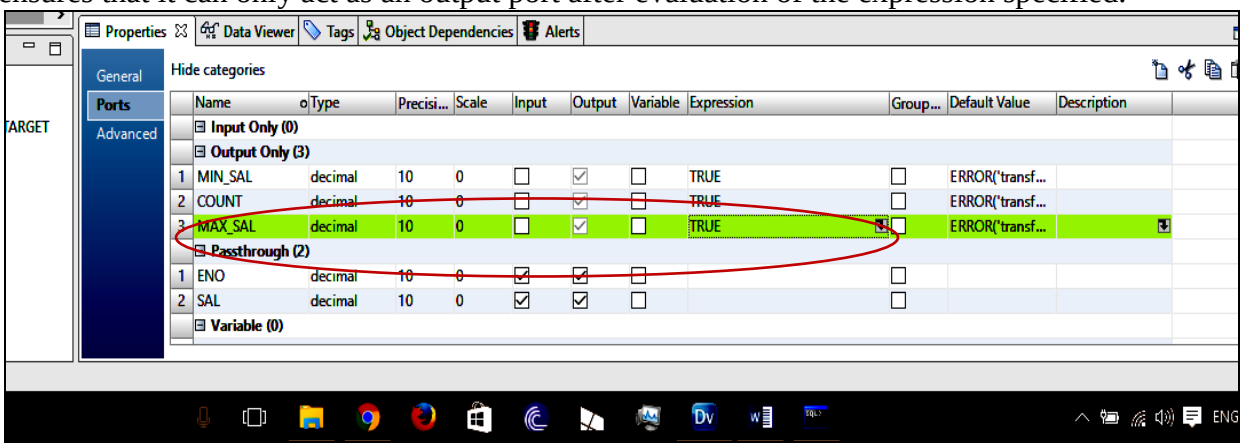
Step 5

Step 6: Drag and drop the values of the target table into the **Aggregator Transformation**. Link up the input table and the output table via the Aggregator Transformation appropriately. Add MIN_SAL, MAX_SAL and COUNT entries to the Transformation as they are required for the evaluation.



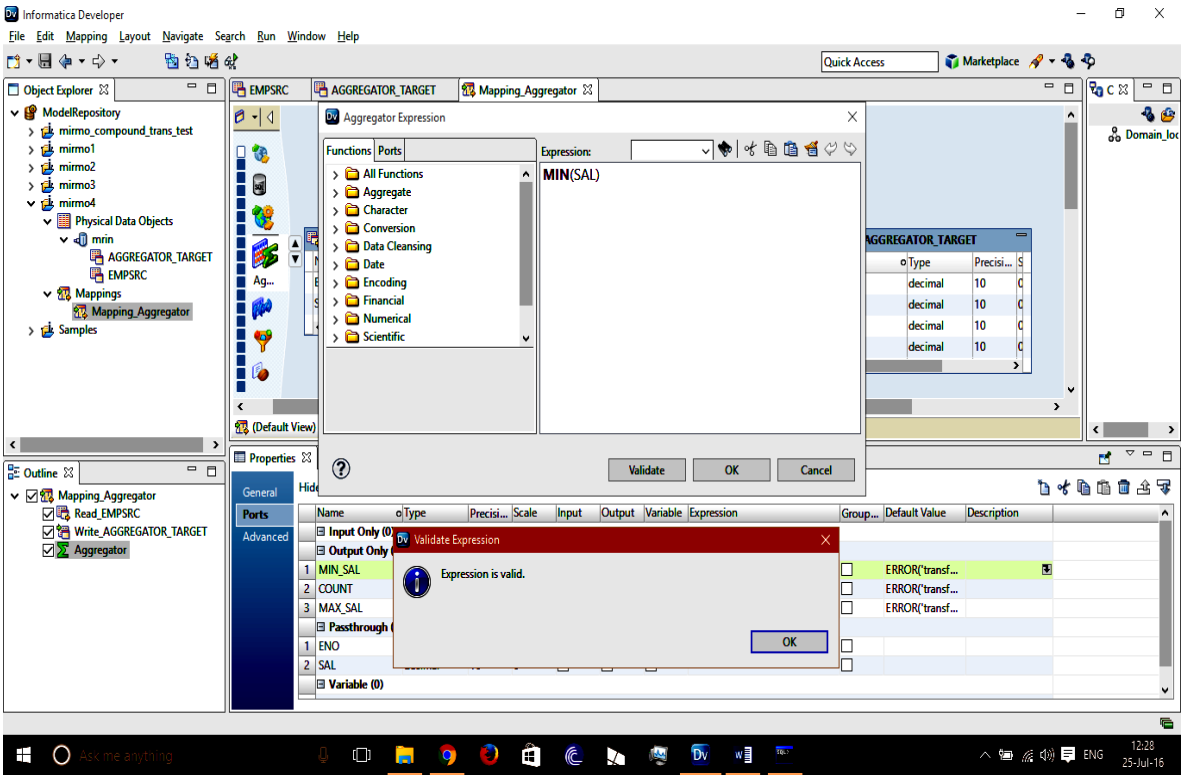
Step 6

Step 7: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **MIN_SAL, MAX_SAL and COUNT**. This ensures that it can only act as an output port after evaluation of the expression specified.



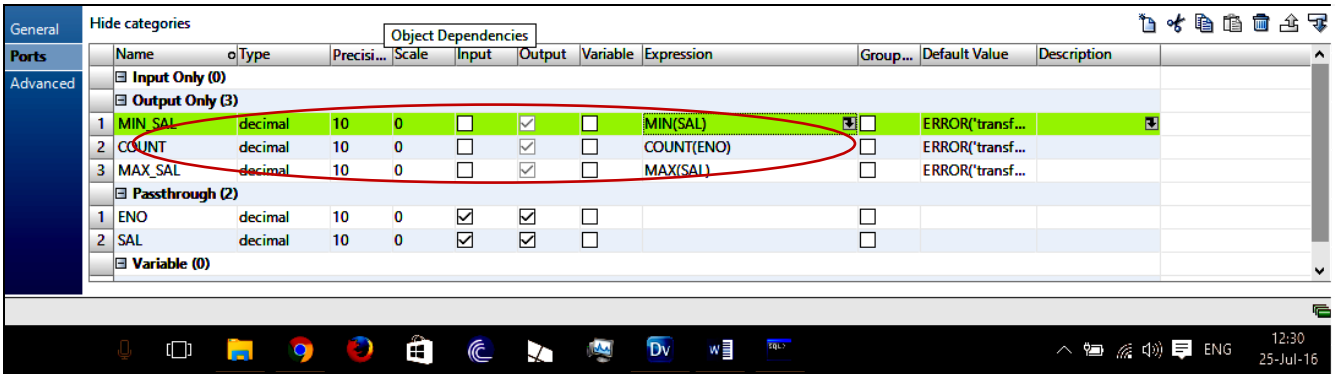
Step 7

Step 8: In the Expression Section of the MIN_SAL entry, click on the small arrow head to open up the **Expression Dialog Box,** navigate to **Functions ➔ Aggregate** and select **MIN** function.This function checks for the least value among all the values and returns it. Then navigate to **Ports** and select **SAL**, so that the expression reads "**MIN(SAL)".** Don't forget to validate the expression for errors.



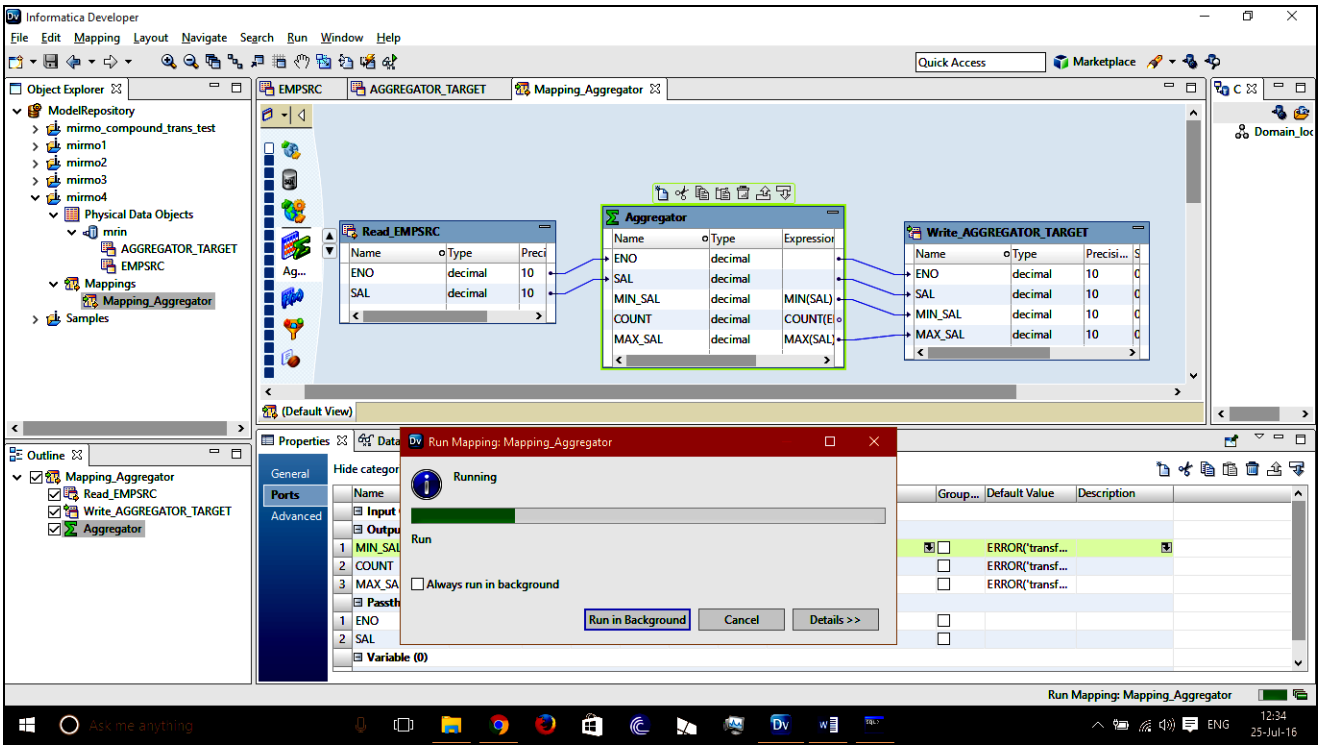
Step 8.1

Similarly, do it for **MAX_SAL** and **COUNT** appropriately.



Step 8.2

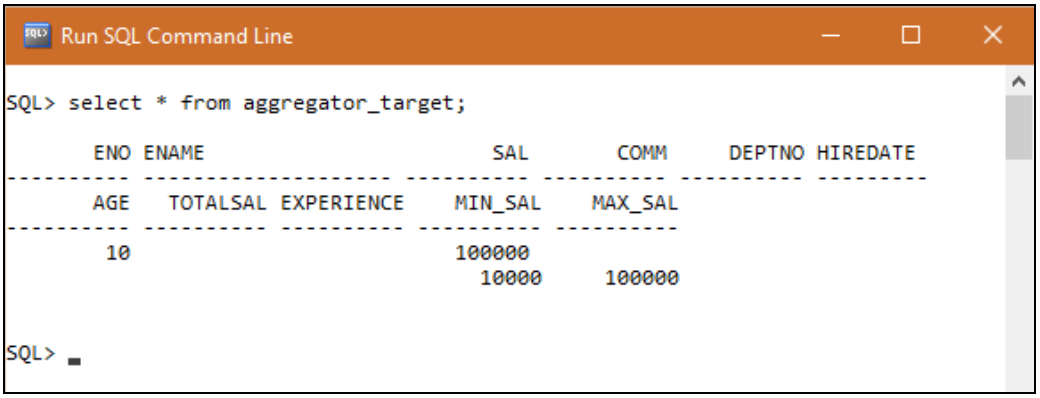
Step 9: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.



Step 9

Step 10: Verify the output of Aggregator Transformation in the SQL Command Prompt.

Output:



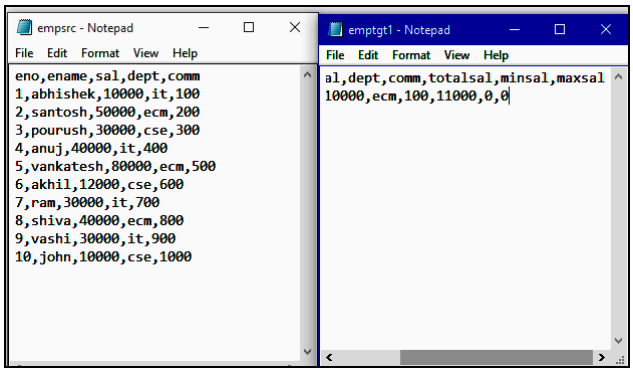
Aggregator Transformation was successful. Max and Min salaries for the table were obtained.

Aim: To apply an Aggregator Transformation on a Data Set to calculate Minimum and Maximum Salary in the table.

Theory: Aggregator transformation is an active transformation used to perform calculations such as sums, averages, counts on groups of data. The integration service stores the data group and row data in aggregate cache. The Aggregator Transformation provides more advantages than the SQL, you can use conditional clauses to filter rows.

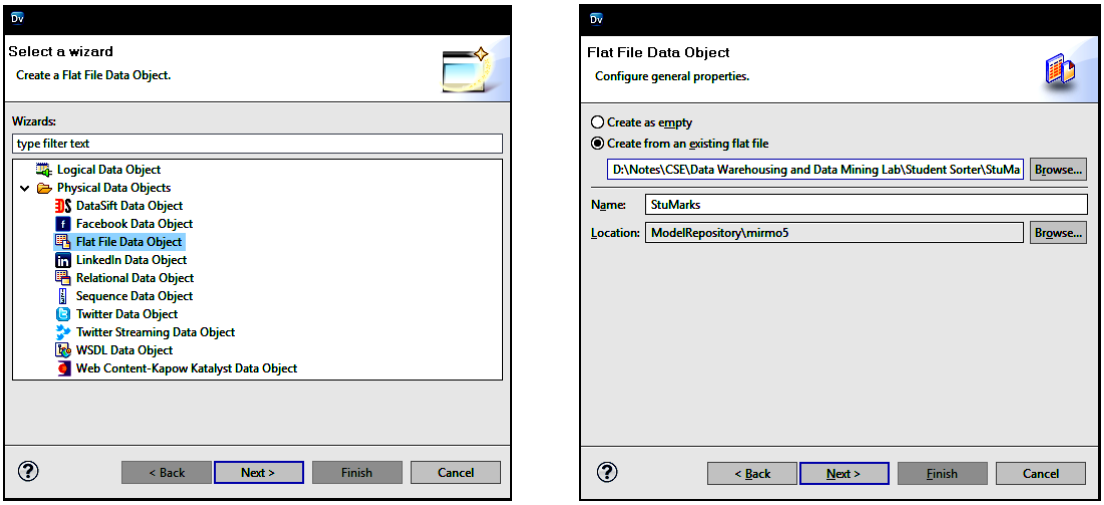
Procedure:

Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ‘,’ as a delimiter.



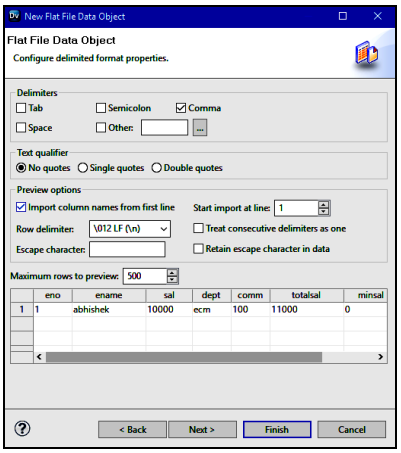
Step 1

Step 2: Start up Informatica Services and Launch Informatica Developer.
Step 3: Import your flat files (both source and target files) into **mirmo2** project.



Step 3

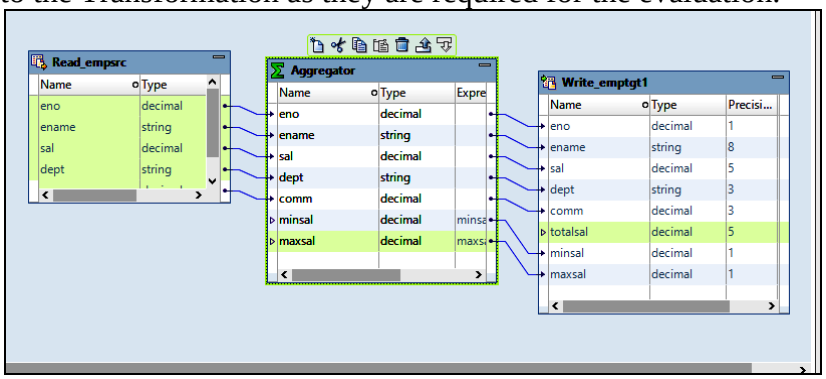
Click on Next and then Next again. In the next dialog box that opens, check the **“Import Column Names from the first line”** checkbox and make sure the delimiter selected is ‘**,**’. Then click on **Finish**.



Step 3

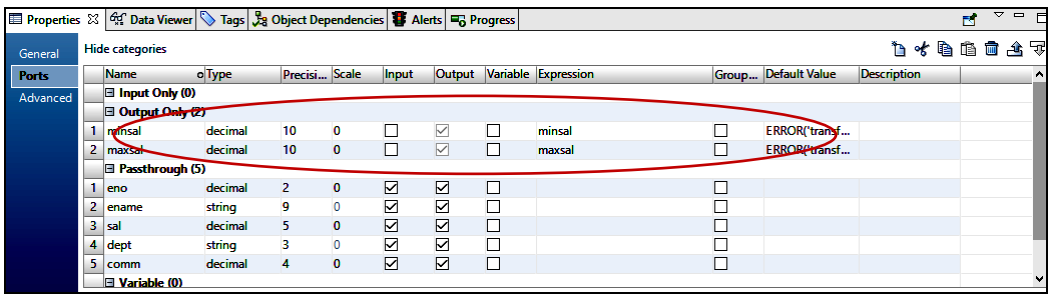
Step 4: Once the files have been imported, create a new mapping and name it **Aggregator_Mapping**.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (AGGREGATOR_TARGET) table (in **write** mode). Delete all the non-required entries from the tables. Also drag the Aggregator Transformation from the palette. Drag and drop the values of the target table into the **Aggregator Transformation**. Link up the input table and the output table via the Aggregator Transformation appropriately. Add MIN_SAL, MAX_SAL and COUNT entries to the Transformation as they are required for the evaluation.



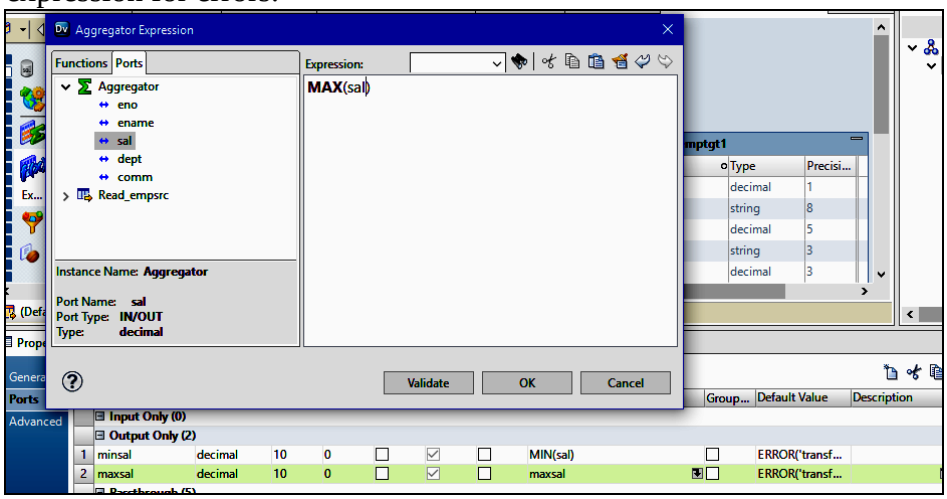
Step 5

Step 6: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **MIN_SAL, MAX_SAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



Step 6

Step 7: In the Expression Section of the MIN_SAL entry, click on the small arrow head to open up the **Expression Dialog Box,** navigate to **Functions ➔ Aggregate** and select **MIN** function.This function checks for the least value among all the values and returns it. Then navigate to **Ports** and select **SAL**, so that the expression reads "**MIN(SAL)".** Don't forget to validate the expression for errors.

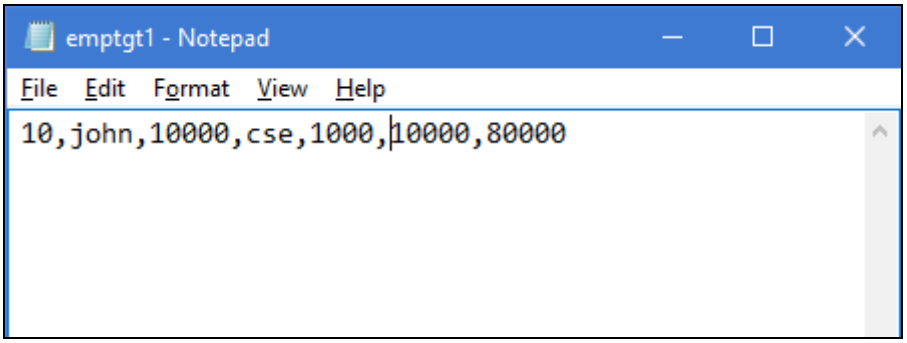


Step 7

Step 8: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.

Step 9: Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Aggregator Transformation was successful. Max and Min salaries for the table were obtained.

Joiner Transformation in Informatica

The joiner transformation is an active and connected transformation used to join two heterogeneous sources. The joiner transformation joins sources based on a condition that matches one or more pairs of columns between the two sources. The two input pipelines include a master and a detail pipeline or branch. To join more than two sources, you need to join the output of the joiner transformation with another source. To join n number of sources in a mapping, you need n-1 joiner transformations.

Creating Joiner Transformation

Follow the below steps to create a joiner transformation in informatica

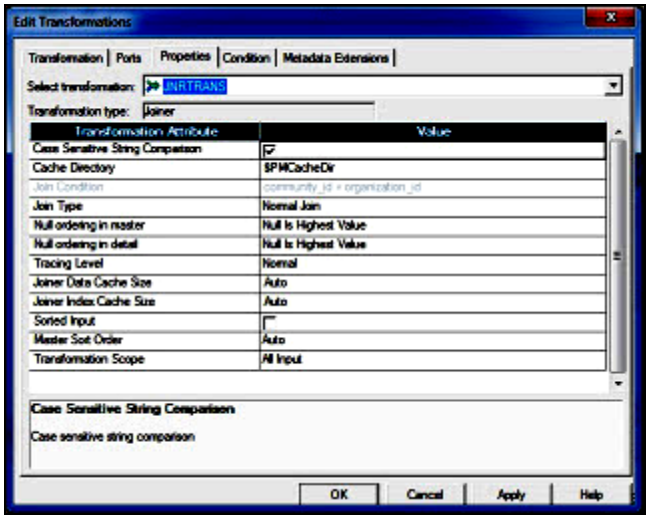
- Go to the mapping designer, click on the Transformation->Create.
- Select the joiner transformation, enter a name and click on OK.
- Drag the ports from the first source into the joiner transformation. By default the designer creates the input/output ports for the source fields in the joiner transformation as detail fields.
- Now drag the ports from the second source into the joiner transformation. By default the designer configures the second source ports as master fields.
- Edit the joiner transformation, go the ports tab and check on any box in the M column to switch the master/detail relationship for the sources.
- Go to the condition tab, click on the Add button to add a condition. You can add multiple conditions.
- Go to the properties tab and configure the properties of the joiner transformation.

Configuring Joiner Transformation

Configure the following properties of joiner transformation:

- **Case-Sensitive String Comparison**: When performing joins on string columns, the integration service uses this option. By default the case sensitive string comparison option is checked.
- **Cache Directory**: Directory used to cache the master or detail rows. The default directory path is $PMCacheDir. You can override this value.
- **Join Type**: The type of join to be performed. Normal Join, Master Outer Join, Detail Outer Join or Full Outer Join.
- **Tracing Level**: Level of tracing to be tracked in the session log file.
- **Joiner Data Cache Size**: Size of the data cache. The default value is Auto.
- **Joiner Index Cache Size**: Size of the index cache. The default value is Auto.

- **Sorted Input**: If the input data is in sorted order, then check this option for better performance.
- **Master Sort Order**: Sort order of the master source data. Choose Ascending if the master source data is sorted in ascending order. You have to enable Sorted Input option if you choose Ascending. The default value for this option is Auto.
- **Transformation Scope**: You can choose the transformation scope as All Input or Row.



Join Condition

The integration service joins both the input sources based on the join condition. The join condition contains ports from both the input sources that must match. You can specify only the equal (=) operator between the join columns. Other operators are not allowed in the join condition. As an example, if you want to join the employees and departments table then you have to specify the join condition as department_id1= department_id. Here department_id1 is the port of departments source and department_id is the port of employees source.

Join Type

The joiner transformation supports the following four types of joins.

- Normal Join
- Master Outer Join
- Details Outer Join
- Full Outer Join

We will learn about each join type with an example. Let say i have the following students and subjects tables as the source.

```
Table Name: Subjects
Subject_Id subject_Name
----------------------
1          Maths
2          Chemistry
3          Physics

Table Name: Students
Student_Id  Subject_Id
--------------------
10           1
20           2
30           NULL
```

Assume that subjects source is the master and students source is the detail and we will join these sources on the subject_id port.

Normal Join:

The joiner transformation outputs only the records that match the join condition and discards all the rows that do not match the join condition. The output of the normal join is

```
Master Ports        |   Detail Ports
-------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
-------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
```

Master Outer Join:

In a master outer join, the joiner transformation keeps all the records from the detail source and only the matching rows from the master source. It discards the unmatched rows from the master source. The output of master outer join is

```
Master Ports        |   Detail Ports
-------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
-------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
NULL       NULL         30         NULL
```

Detail Outer Join:

In a detail outer join, the joiner transformation keeps all the records from the master source and only the matching rows from the detail source. It discards the unmatched rows from the detail source. The output of detail outer join is

```
Master Ports        |   Detail Ports
--------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
--------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
3          Physics      NULL       NULL
```

Full Outer Join:

The full outer join first brings the matching rows from both the sources and then it also keeps the non-matched records from both the master and detail sources. The output of full outer join is

```
Master Ports        |   Detail Ports
--------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
--------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
3          Physics      NULL       NULL
NULL       NULL         30         NULL
```

Sorted Input

Use the sorted input option in the joiner properties tab when both the master and detail are sorted on the ports specified in the join condition. You can improve the performance by using the sorted input option as the integration service performs the join by minimizing the number of disk IOs. you can see good performance when worked with large data sets.

Steps to follow for configuring the sorted input option

- Sort the master and detail source either by using the source qualifier transformation or sorter transformation.
- Sort both the source on the ports to be used in join condition either in ascending or descending order.
- Specify the Sorted Input option in the joiner transformation properties tab.

Why joiner transformation is called as blocking transformation

The integration service blocks and unblocks the source data depending on whether the joiner transformation is configured for sorted input or not.

Unsorted Joiner Transformation

In case of unsorted joiner transformation, the integration service first reads all the master rows before it reads the detail rows. The integration service blocks the detail source while it caches the all the master rows. Once it reads all the master rows, then it unblocks the detail source and reads the details rows.

Sorted Joiner Transformation

Blocking logic may or may not possible in case of sorted joiner transformation. The integration service uses blocking logic if it can do so without blocking all sources in the target load order group. Otherwise, it does not use blocking logic.

Joiner Transformation Performance Improve Tips

To improve the performance of a joiner transformation follow the below tips

- If possible, perform joins in a database. Performing joins in a database is faster than performing joins in a session.
- You can improve the session performance by configuring the Sorted Input option in the joiner transformation properties tab.
- Specify the source with fewer rows and with fewer duplicate keys as the master and the other source as detail.

Limitations of Joiner Transformation

The limitations of joiner transformation are

- You cannot use joiner transformation when the input pipeline contains an update strategy transformation.
- You cannot connect a sequence generator transformation directly to the joiner transformation.

Aim: To apply Joiner Transformation on a Data Set to perform Normal Join.

Theory: The joiner transformation provides you the option to create joins in Informatica. The joins created using joiner transformation are similar to the joins in databases. The advantage of joiner transformation is that joins can be created for heterogeneous systems (different databases). In joiner transformation, there are two sources which we are going to use it for joins. These two sources are called

- Master Source
- Detail Source

In the properties of joiner transformation, you can select which data source can be Master and which source can be detail source.During execution, the master source is cached into the memory for joining purpose. So it is recommended to select the source with less number of records as the master source.

The following joins can be created using joiner transformation

1. **Master outer join:**In Master outer join, all records from the Detail source are returned by the join and only matching rows from the master source are returned.
2. **Detail outer join:**In detail outer join only matching rows are returned from the detail source, and all rows from the master source are returned.
3. **Full outer join:**In full outer join, all records from both the sources are returned. Master outer and Detail outer joins are equivalent to left outer joins in SQL.
4. **Normal join:**In normal join only matching rows are returned from both the sources.

Procedure:

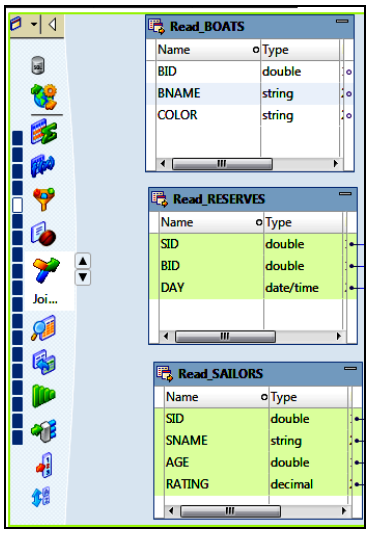
Step 1: Create the following tables Boats(bid number(2), bname varchar2(20), color varchar2(20)), Reserves(sid number(2), bid number(2), day date)), Sailors(sid number(2), sname varchar2(20), age number(2), rating number(2)) and populate them with values.

Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Create a new project (mirmo5) and import your tables from the database (both source and target files).

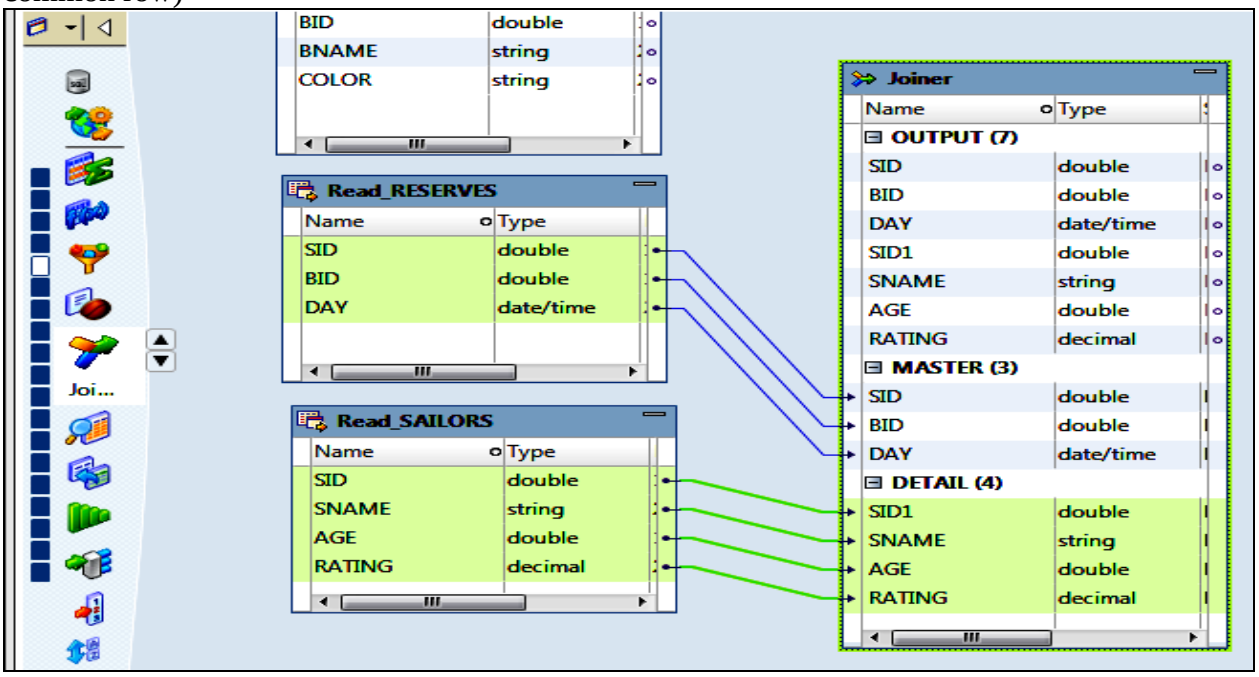
Step 4: Once the files have been imported, create a new mapping and name it **Joiner_Mapping**.

Step 5: Drag all the three tables in read mode and place them in a vertical column.



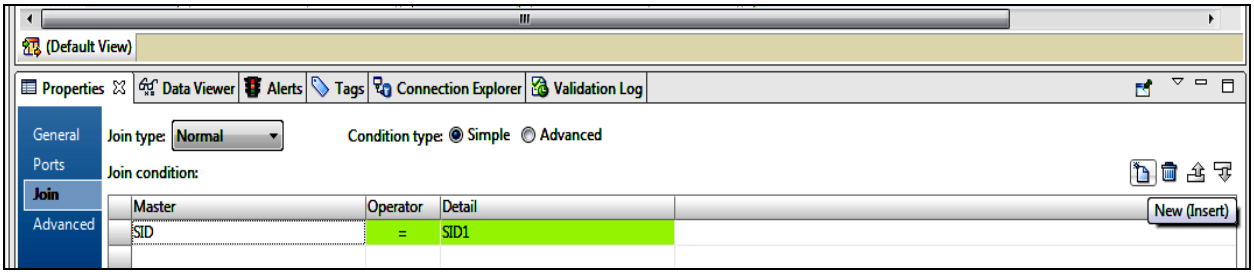
Step 5

Step 6: Now drag joiner from the palette and load two tables into it. (Both tables must have a common row)



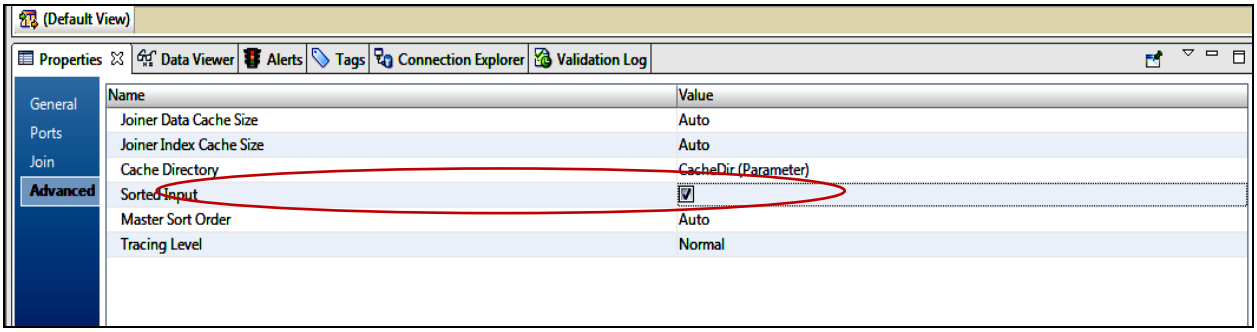
Step 6

Step 7: Navigate to **Properties ➔ Join** in the Properties Tab. Specify the join type. In this case, we use Normal Join. And specify the join condition sid of Reserves = sid of Sailors i.e., Master SID = Detail SID1.



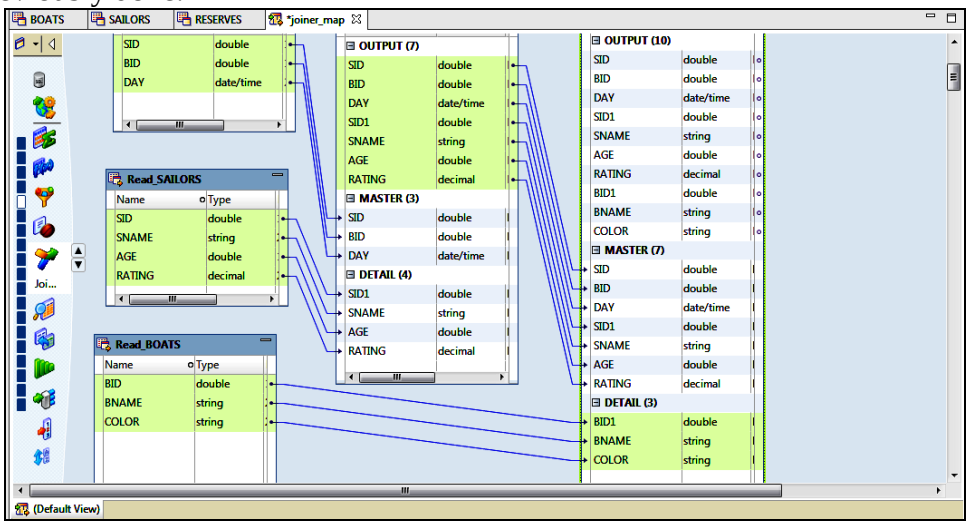
Step 7.1

Navigate to **Properties ➔ Advanced** of the Properties Tab and put a check mark to the Sorted Input option so that the input is sorted.

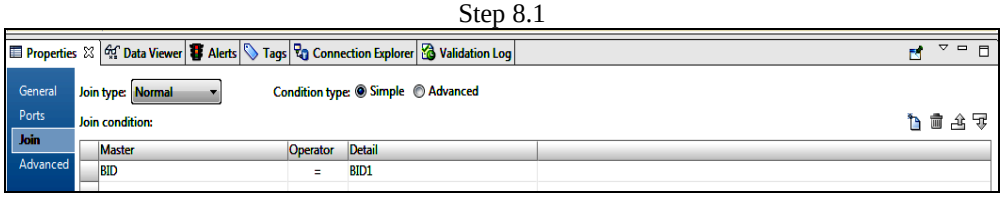


Step 7.2

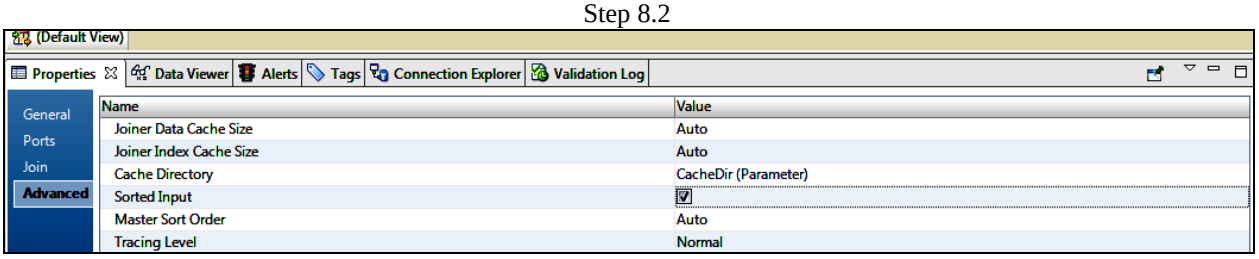
Step 8: Now Drag another Joiner from the palette and give the output of the previous Joiner as the Master and Boats as the Detail. Give the condition as Master BID = Detail BID1 and Sorted Input as previously done.



Step 8.1



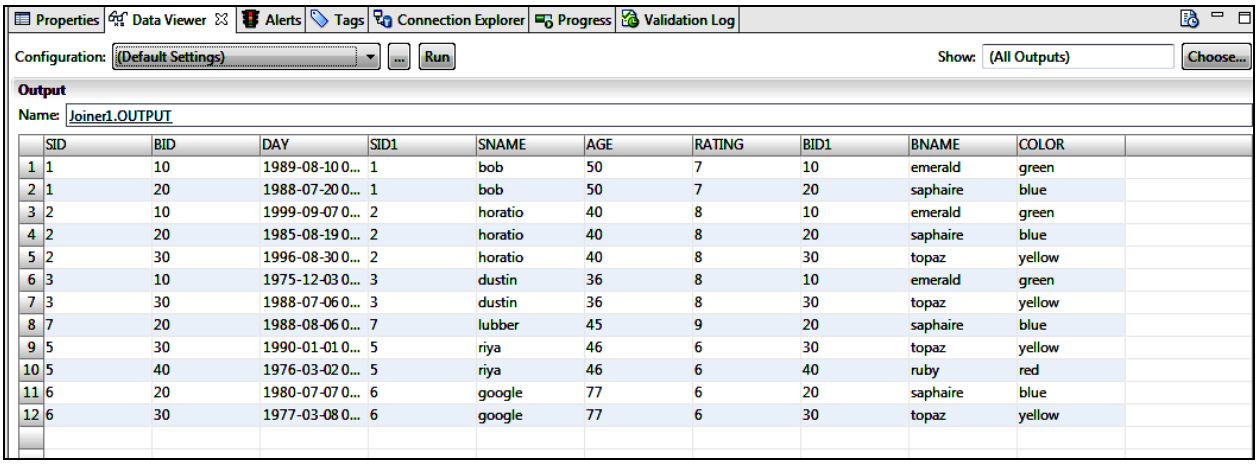
Step 8.2



Step 8.3

Step 9: Now Run the Mapping. Check the Database or the DataViewer for the Output.

Output:



	SID	BID	DAY	SID1	SNAME	AGE	RATING	BID1	BNAME	COLOR
1	1	10	1989-08-10 0...	1	bob	50	7	10	emerald	green
2	1	20	1988-07-20 0...	1	bob	50	7	20	saphaire	blue
3	2	10	1999-09-07 0...	2	horatio	40	8	10	emerald	green
4	2	20	1985-08-19 0...	2	horatio	40	8	20	saphaire	blue
5	2	30	1996-08-30 0...	2	horatio	40	8	30	topaz	yellow
6	3	10	1975-12-03 0...	3	dustin	36	8	10	emerald	green
7	3	30	1988-07-06 0...	3	dustin	36	8	30	topaz	yellow
8	7	20	1988-08-06 0...	7	lubber	45	9	20	saphaire	blue
9	5	30	1990-01-01 0...	5	riya	46	6	30	topaz	yellow
10	5	40	1976-03-02 0...	5	riya	46	6	40	ruby	red
11	6	20	1980-07-07 0...	6	google	77	6	20	saphaire	blue
12	6	30	1977-03-08 0...	6	google	77	6	30	topaz	yellow

Output

Normal Join has been successfully performed.

Aim: To apply Joiner Transformation on a Flat File Database to perform Full Outer Join.

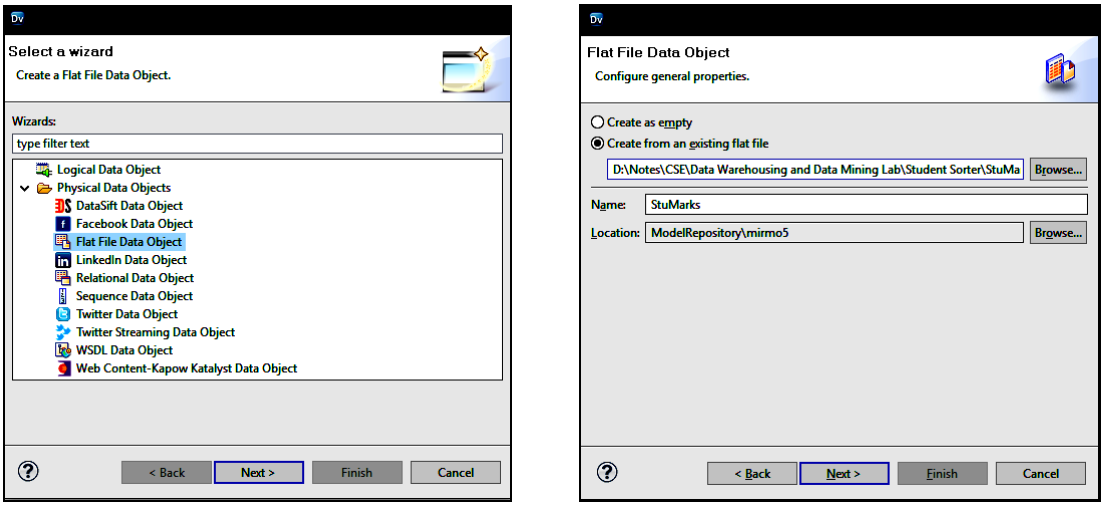
Theory: The joiner transformation provides you the option to create joins in Informatica. The joins created using joiner transformation are similar to the joins in databases. The advantage of joiner transformation is that joins can be created for heterogeneous systems (different databases).

Procedure:

Step 1: Create the followingflatfilesReserves(sid,bid,day), Sailors(sid,sname,age,rating) and populate them with values.

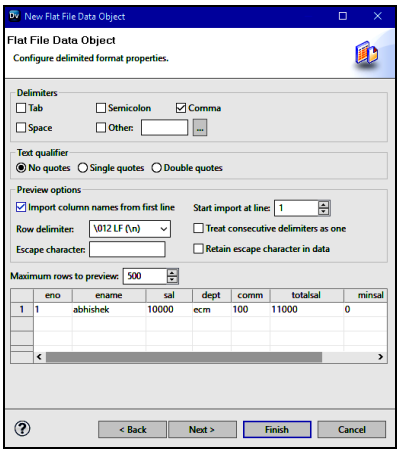
Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Import your flat files into **mirmo2** project.



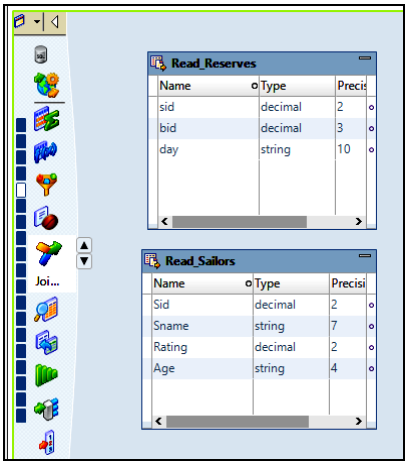
Step 3

Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.



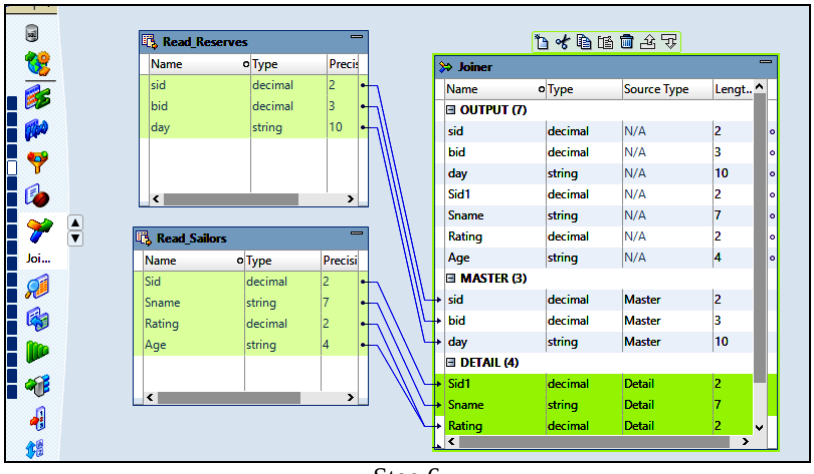
Step 3

Step 4: Once the files have been imported, create a new mapping and name it **Joiner_Mapping**.
Step 5: Drag all the files in read mode and place them in a vertical column.



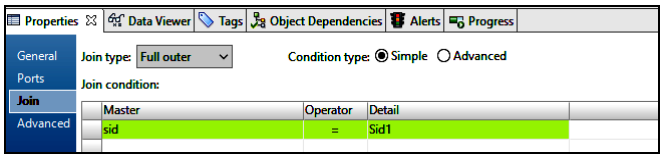
Step 5

Step 6: Now drag joiner from the palette and load the tables into it. (Both tables must have a common row)



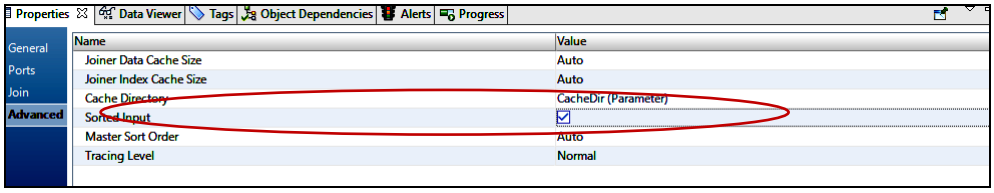
Step 6

Step 7: Navigate to **Properties ➔ Join** in the Properties Tab. Specify the join type. In this case, we use Full Outer Join. And specify the join condition sid of Reserves = sid of Sailors i.e., Master SID = Detail SID1.



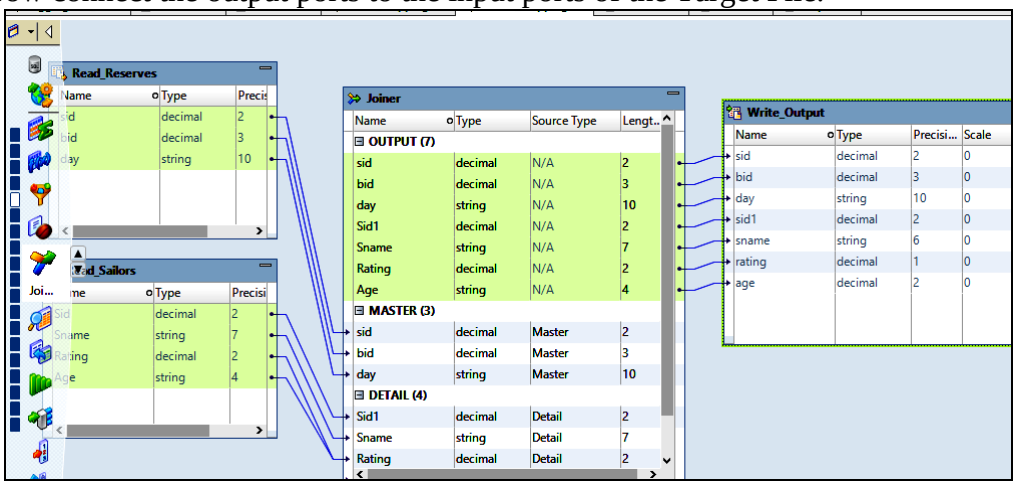
Step 7.1

Navigate to **Properties ➔ Advanced** of the Properties Tab and put a check mark to the Sorted Input option so that the input is sorted.



Step 7.2

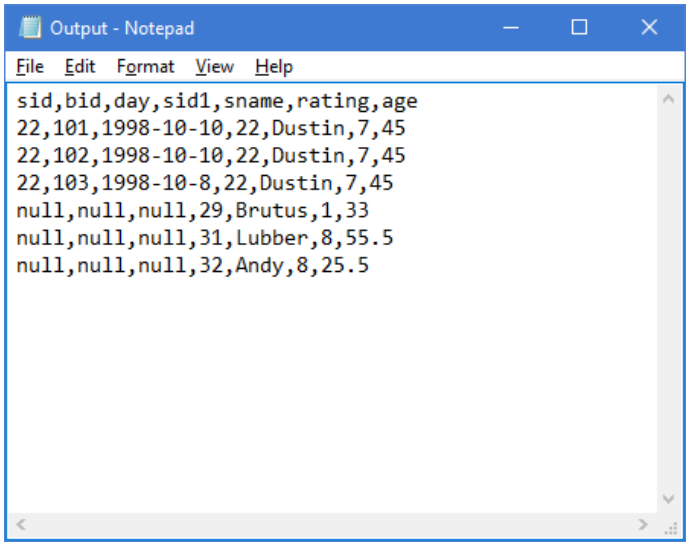
Step 8: Now connect the output ports to the input ports of the Target File.



Step 8

Step 9: Now Run the Mapping. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Output

Full Outer Join has been successfully performed.

Exploring understanding Clementine Tool

INTRODUCING CLEMENTINE

CLEMENTINE AND CLEMENTINE SERVER

By default, Clementine will run in local mode on your desktop machine. If Clementine Server has been installed, then Clementine can be run in local mode or in distributed (Client-Server) mode. In this latter mode, Clementine streams are built on the client machine, but executed by Clementine Server.

To determine in which mode Clementine is running on your machine, examine the connection status area of the Clementine status bar (left-most area of status bar) or click Tools..Server Login(from within Clementine) if the choice is active; if it is not active, then Clementine Server is not licensed. See that the Server Login dialog whether the Connection is set to Local or Network.

Note :

Data for this course are assumed to be stored in the directory c:\programfiles\SPSSClementine\10.1 .

STARTING CLEMENTINE

To run Clementine:

From the Start button, click **Programs..Clementine..Clementine**

Clementine enables you to mine data by visual programming techniques using the Stream Canvas.This is the main work area in Clementine and can be thought of as a surface on which to place icons. These icons represent operations to be carried out on the data and are often referred to as nodes.

The nodes are contained in palettes, located across the bottom of the Clementine window. Once nodes have been placesd on the Stream Canvas, they can be linked together to form a stream.

READING DATA FILES

READING DATA FILES INTO CLEMENTINE

Clementine reads a variety of different file types, including data stored in spreadsheets and databases, using the nodes within the Sources palette.

Data can be read in from text files, in either free-field or fixed-field format, using the Var. File and Fixed File source nodes.

Reading Data from Free-Field Text Files

The Var. File node reads data from a free-field (delimited) text file.

We will read this file into Clementine, using the Var File source node.

If the Stream Canvas isn't empty, clear the Stream Canvas by choosing **Edit..Clear Stream**
Click the **Var. File** node in the Sources palette

Position the cursor on the left side of the Stream Canvas and click once(not shown)
Right-click on the **Var. File** node, then click **Edit**

Click the file list button ☐ , and then move to the
c:\programfiles\SPSS\clementine\10.1 directory
Click Drug.txt and then click **Open**
Click **OK**
Click the **Table** node from the Output palette
Click to the right of the Var. File node named Drug.txt

Right-click the **Var. File** node, select **Connect** from the context pop-up menu, and then click

the **Table** node in the Stream Canvas

Click the **Execute** button in the Toolbar

After having executed the Table using any of the methods mentioned, the table window opens.

Exercise : 1

1. Execute the above steps and at each step display the screen.
2. Read the Fixed data file and display the data output on the screen in the table format.
3. Read the data using ODBC.

DATA QUALITY

In Clementine there are a number of different types of representation of missing data. First, a field may be completely blank. Clementine calls such missing information white space string if the field is symbolic and null value if the field is numeric. Clementine also refers to this as a null value or non-numeric missing. Finally, when entering data, predefined codes may be used to represent missing or invalid information.Clementine refers to such codes as value balnks.

Assessing Data Quality

To illustrate the handling of missing data we will open Drug1n.txt and assess the quality of the data.

If the Stream Canvas is not empty, start a new stream by clicking **File..New Stream**
Select the **Var. File** node and place it on the Stream Canvas

Edit the node and set the file to **Drug1n.txt** held in the
c:\programfiles\SPSS\Clementine\10.1\Demos directory

Make sure the **Read field names from file** option is checked
Click the **Types** tab, then right-click any field and click **Select All** from the context menu
Right-click any field, and then click **Set Values..<Read>** from the context menu

Click **OK**
Add a Table node and connect the **Var. File** node to the **Type** node

Execute the **Table** node
Click **File..Close** to close the **Table** node
Double-click the **Var. File** node and click the **Types** tab
Place a **Quality** node from the Output palette into the Strem Canvas and connect the

Var.File node to it.
Edit the **Quality** node
Deselect the **White space** option
Deselect **the Empty string** option
Click the **Execute** button

Exercise : 2

1. Execute the above steps and display the screens for each step.
2. For the above data check for all missing value types i.e Null value, Empty String, White Space, Blank Value and display the screens
3. Examine the distribution of data fields for drug4n.txt in Clementine by using Data Audit node.

DATA MANIPULATION

Once checking the quality of data has been completed it is often necessary to manipulate the data further.

Such techniques are available within Clementine and can be found in either the Record Ops palette (containing tools for manipulating records) or Field Ops palette (Containing tools for manipulating fields).

Record operations and the Select Node

The select node, which allows you to either select or eliminate a group of records based on a specified condition.

If the Stream Canvas is not empty, choose **File..Close Stream**

Click **File..Open Stream**, move to the **c:\programfiles\SPSS\Clementine\10.1\demos** directory and double-click **Drug4n**

Place a **Select** node from the Record Ops palette to the right of the **Type** node
Connect the **Type** node to the **Select** node

Right-click the **Select** node, then click **Edit**
Type **Age < 25** in the **Condition** text box
Check that **Mode:** is set to **Include**

Place a **Table** node from the Output palette to the right of the **Select** node
Connect the **Select** node to the new **Table** node
Right –click the new **Table** node, then click **Execute**

Next, using a Distribution node, we will compare the distribution of risk for the entire sample to the subgroup with Age < 25.

Click **File..Close** to close the Table window
Delete the **Table** node

Drag the **select** node below the **Type** node
Place **a Distribution** node from the Graphs palette to the right of the **Type** node
Connect the **type** node to the **Distribution** node
Double-click the **Distribution** node
Click the field list button and select **Drug4n**
Click **OK**
Copy the **Distribution** node and **Paste** it to the right of the **Select** node
Connect the **Select** node to the pasted **Distribution** node
Execute the two **Distribution** nodes.

Field Operations and the Filter Node

The Filter node allows data to pass through it and has two main functions:

- To filter out unwanted fields
- To rename fields

Place a **Filter** node from the **Field Ops** palette to the right of the **Type** node
Connect the **Type** node to the **filter** node
Right-click on the **Filter** node, and then click **Edit**

To change the Name of a Field

Click the text in the right column

Type the new name in the text box

To Filter Out Fields

Click on the arrow next to **Age**
Click **OK** to close the **Filter** dialog box

Field Reordering

Place a **Field Reorder** node from the Field Ops . Palette to the right of the **Filter** node
Connect the **Filter** node to **Field reorder** node
Right –click the **Field Reorder** node, and then click **Edit**
Click the field list button
Select **Na, Drug** in the Select Fields dialog
Click **OK**

Click **Drug** in the Field Reorder dialog
Click **Ok**

Place a **Table** node from the Output palette to the right of the **Field Reorder** node
Connect the **Field Rorder** node to the **Table** node
Right-click the new **Table** node, then click **Execute.**

The Derive Node

The Derive node calculates a new value, based on a CLEM expression for every record passed through it. To enter the CLEM expression and the new field name you need to edit the Derive node.

Click **Insert..Field Ops..Derive**
Right-click the **Derive** node, then click **Edit**
Select **Formula from the Derive as :** drop-down list
Type **SUM NAK** in the **Derive field**: text box

Click the Expression Builder button
Click **Na** in the Fields list box, then click the **Insert** button
Click the plus button
Click **Ki**n the Fields list box, then click the **Insert** button
Click the plus button

Click **OK**

Click **OK**

Exercise :3

1. For the above steps display the screens step wise.
2. Write the steps for Derive Type Flag and display the screens
3. Write the steps for Derive Type set and display the screens
4. Write the steps for Derive Type Conditional and display the screens
5. Write the steps for Derive Type Flag and display the screens

Reclassify Node

The Reclassify node allows you to recode or reclassiry the data values for fields of set or flag type.

Place a **Reclassify** node from the **Field Ops** palette to the right of the **Derive** node named **Drug** in the Stream Canvas
Connect the **Derive** node named **Drug** to the **Reclassify** node
Right-click the **Reclassify** node, then click **Edit**
Click the field list button for **Reclassify field** and select **Drug**

Type **Drug_Cat** in the **New field name** box
Click the **Get** button
Click the For unspecified values use: **Default value** option button
Type **Da** in the **New value box** for the **DrugA** row
Click in the **right side** of the **New value box** for the **DrugB** row, then click the drop-**down arrow**, and select **Db** from the drop-down value list
Type **Dc** in the **New value** box for the **DrugC** row

Click **OK**

Place a **Table** from the Output palette above the **Reckassify** node
Connect the **Reclassify** node to new **Table** node
Right-click the new **Tabl**e node, then click **Execute**

Exercise : 4

1. Generate the Screens at each step of Reclassify node.

RELATIONSHIPS IN DATA

There are two methods for examining whether symbolic fields are related. The first is the Matrix node used to display the relation between a pair of symbolic fields. We will extend this to more than two symbolic fields with the graphical Web node.

Matrix Node : Relating Two Symbolic Fields

The Matrix node performs crosstabulations of two symbolic fields within the data, and shows how values of one field relate to those of a second field.

Click File..Open Stream, move to c:\programsfile\SPSS\Clementine\10.1\demos directory and double-click on **Drug4n**
Place a **Matrix** node from the **Output** palette to the right of the **Type** node
Connect the **Type** node to the **Matrix** node
Double-click on the **Matrix** node

Click the Fields list button in the **Rows**: list and select **Drug**
Click the Fields list button in the **Columns:** list and select **Gender**

Click the **Appearance t**ab
Select **Counts**
Select **Percentage of column**

Click the **Execute** button

Click **file..Close** to close the Matrix output window
Double-click on the **Matrix** node, and then click the **settings** tab
Click the Fields list button in the Columns: list and select **Na**
Click **Execute**

Web Node

The Web node, located in the Graphs pallete, can be used to show the strength of connection between values of two or more symbolic fields.

Place a **Web** node from the **Graphs** palette near the **Type** node
Connect the **Type** node to the **Web** node
Double-click the **Web** node

Click the Field List button

Select **Age, Sex** and **Drug**
Click **Ok** to return to the Web dialog box

Click the **Line values are** drop-down list

Click the **options** tab

Click **show only links above** option button
Type **300** in the **Show only links** above box
Type **400** in the **Weak links below**: box
Type **600** in the **Strong links below**: box

Click **Execute**
Drag the lower corner of the resulting Web graph window to enlarge it.

Click >>
Use **Slider control** to discard links weaker than 450

Click the **Controls** tab in the Web Summary output
Set the **Web Display** option to **Size shows strong/normal/weak**
Use slider control or text box to set the value for **strong links above** to 1500 and the value for **weak links below to 1000**

Click the link connecting **DrugA and F**
Click the link connecting **DrugB and F**
Click **Generate..Derive Node("And")**

Click **File..Close** to close the Web graph window

Exercise : 5

1. Generate the screens at each step of Matrix node .
2. Generate the screens at each step of Web node .