LECTURE

Graphical User Interface Testing

Software UI

A User Interface (UI) is the visual part of a software application that determines how a user interacts with an application or a website and how information is displayed on the screen.

GUI Testing

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc. GUI Testing test the user interface by considering parameters like consistency, usability, visibility, alignment, spell check, etc.

Need of GUI Testing

- GUI testing is a process to test application's user interface and to detect if application is functionally correct.
- GUI testing involves carrying set of tasks and comparing the result of same with the expected output and ability to repeat same set of tasks multiple times with different data input and same level of accuracy.
- GUI testing includes how the application handles keyboard and mouse events, how different GUI components like menu bars, toolbars, dialogs, buttons, edit fields, list controls, images etc. reacts to user input and whether or not it performs in the desired manner.
- Implementing GUI testing early in the software development cycle speeds up development, improves quality and reduces risks towards the end of the cycle.

GUI Testing Steps

- Determine what to test by defining coverage criteria. A GUI coverage criterion might require the execution of each user interface event to determine whether it behaves correctly.
- Generate test case inputs from software specifications and structure. For GUIs, these inputs consist of events such as mouse clicks, menu selections, and object manipulations.
- Generate expected output to compare with actual output. In GUIs, the expected output includes screen snapshots and window positions and titles.

- Execute test cases and verify output. Test cases execute on the software, and the tester compares the output with the expected output from, for example, an oracle.
- Determine whether the GUI was adequately tested. Once all test cases have executed, the tester analyzes the software to check which of its parts were actually tested. In GUIs, the analysis checks events and resulting GUI states.
- The last step is especially important in GUI testing, where coverage criteria may not always be available or sufficient.

What to Check in GUI Testing

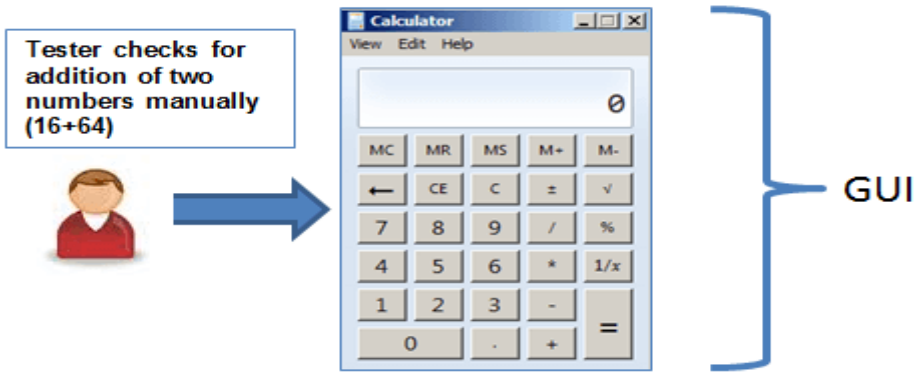
- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

GUI Testing Techniques

GUI Testing Techniques can be categorized into three parts:

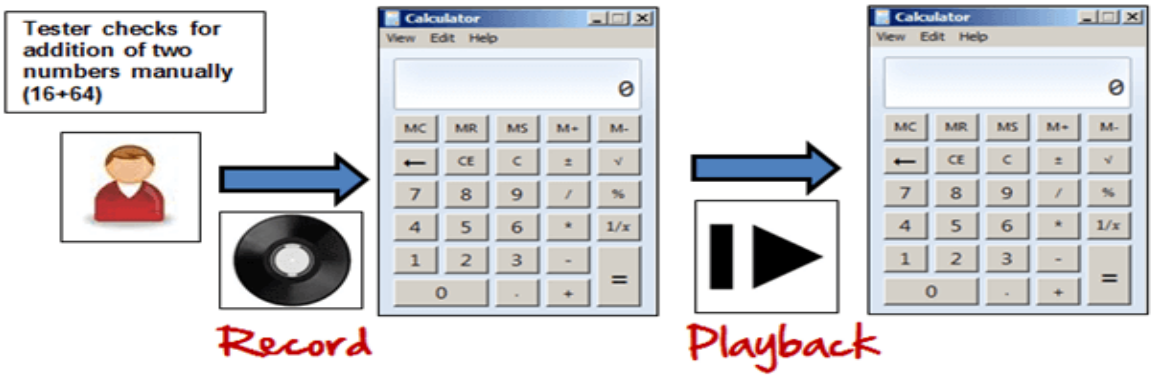
1. **Manual Based Testing**

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in the business requirements document.



2. Record and Replay

GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured by the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

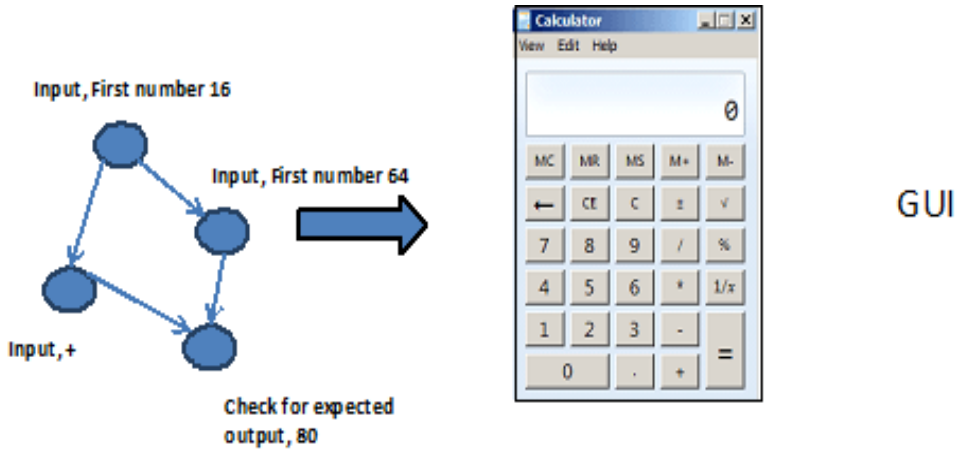


3. Model Based Testing

A model is a graphical description of a system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.

Following needs to be considered for this model based testing:

- Build the model
- Determine Inputs for the model
- Calculate expected output for the model
- Run the tests
- Compare the actual output with the expected output
- Decision on further action on the model



Model Based Testing

Some of the modeling techniques from which test cases can be derived:

- Charts – Depicts the state of a system and checks the state after some input.
- Decision Tables – Tables used to determine results for each input applied

Model based testing is an evolving technique for generating the test cases from the requirements. Its main advantage, compared to above two methods, is that it can determine undesirable states that your GUI can attain.

WEB APPLICATION UI CHECKLIST

1.1 COLORS

1.1.1 Are hyperlink colors standard?
1.1.2 Are the field backgrounds the correct color?
1.1.3 Are the field prompts the correct color?
1.1.4 Are the screen and field colors adjusted correctly for non-editable mode?

1.1.5 Does the site use (approximately) standard link colors?
1.1.6 Are all the buttons are in standard format and size?
1.1.7 Is the general screen background the correct color?
1.1.8 Is the page background (color) distraction free?

1.2 CONTENT

1.2.1 All fonts to be the same
1.2.2 Are all the screen prompts specified in the correct screen font?
1.2.3 Does content remain if you need to go back to a previous page, or if you move forward to another new page?
1.2.4 Is all text properly aligned?
1.2.5 Is the text in all fields specified in the correct screen font?
1.2.6 Is all the heading are left aligned
1.2.7 Does the first letter of the second word appears in lowercase?

1.3 IMAGES

1.3.1 Are all graphics properly aligned?
1.3.2 Are graphics being used the most efficient use of file size?
1.3.3 Are graphics optimized for quick downloads?
1.3.4 Assure that command buttons are all of similar size and shape, and same font & font size.
1.3.5 Banner style & size & display exact same as existing windows
1.3.6 Does text wrap properly around pictures/graphics?
1.3.7 Is it visually consistent even without graphics?

1.4 INSTRUCTIONS

1.4.1 Is all the error message text spelt correctly on this screen?
1.4.2 Is all the micro-help text (i.e tool tip) spelt correctly on this screen?
1.4.3 Microhelp text (i.e tool tip) for every enabled field & button
1.4.4 Progress messages on load of tabbed (active screens) screens

1.5 NAVIGATION

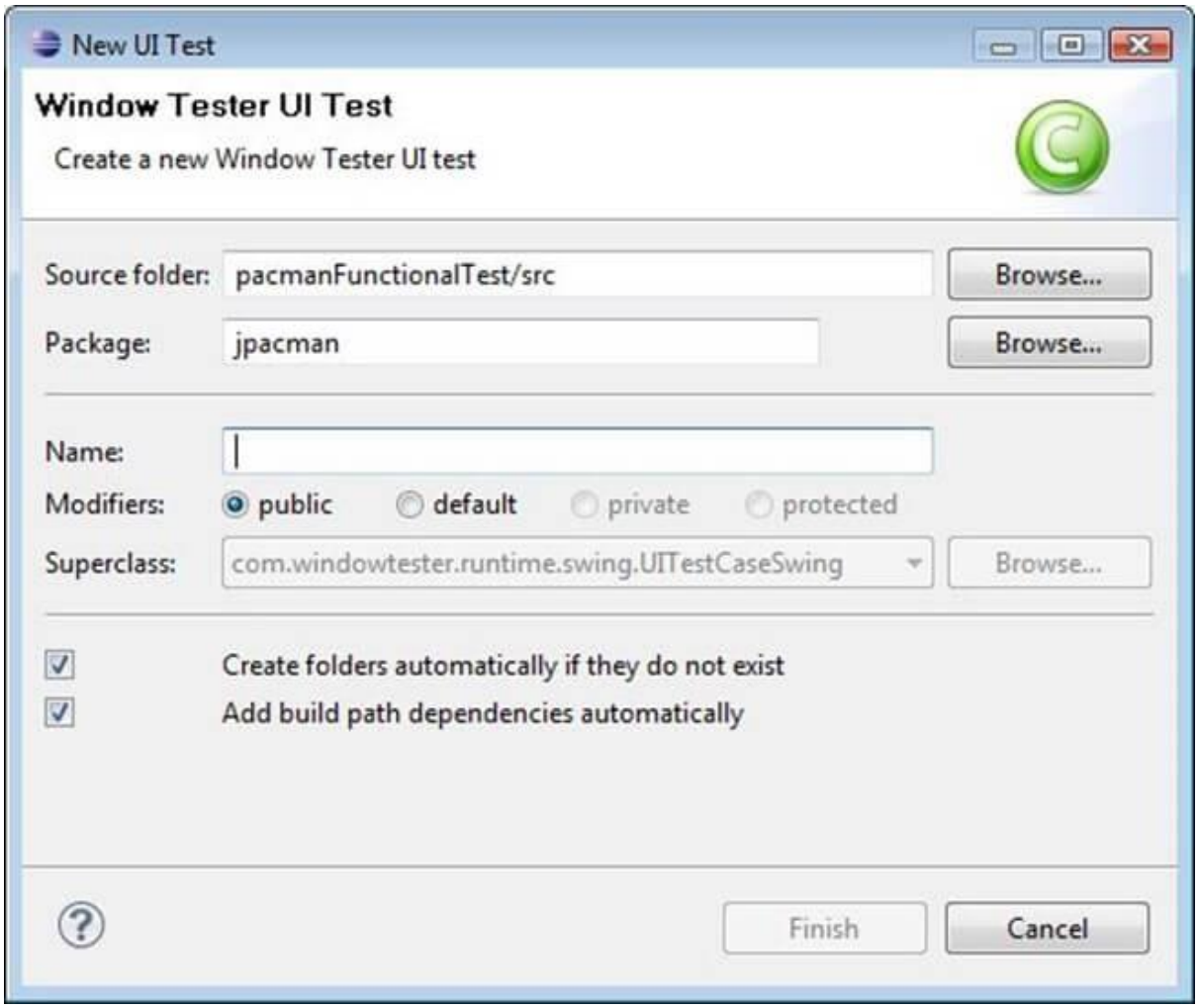
1.5.1 Are all disabled fields avoided in the TAB sequence?
1.5.2 Are all read-only fields avoided in the TAB sequence?
1.5.3 Can all screens accessible via buttons on this screen be accessed correctly?
1.5.4 Does a scrollbar appear if required?
1.5.5 Does the Tab Order specified on the screen go in sequence from Top Left to bottom right? This is the default unless otherwise specified.
1.5.6 Is there a link to home on every single page?
1.5.7 On open of tab focus will be on first editable field
1.5.8 When an error message occurs does the focus return to the field in error when the user cancels it?

1.6 USABILITY

1.6.1 Are all the field prompts spelt correctly?
1.6.2 Are fonts too large or too small to read?
1.6.3 Are names in command button & option box names are not abbreviations.
1.6.4 Assure that option boxes, option buttons, and command buttons are logically grouped together in clearly demarcated areas "Group Box"
1.6.5 Can the typical user run the system without frustration?
1.6.6 Do pages print legibly without cutting off text?
1.6.7 Does the site convey a clear sense of its intended audience?
1.6.8 Does the site have a consistent, clearly recognizable "look-&-feel"?
1.6.9 Does User cab Login Member Area with both UserName/Email ID ?
1.6.9 Does the site look good on 640 x 480, 600x800 etc.?
1.6.10 Does the system provide or facilitate customer service? i.e. responsive, helpful, accurate?
1.6.11 Is all terminology understandable for all of the site’s intended users?

How to do GUI Test

Here we will use some sample test cases for the following screen.



Following below is the example of the Test cases, which consists of UI and Usability test scenarios.

TC 01- Verify that the text box with the label "**Source Folder**" is aligned properly.

TC 02 - Verify that the text box with the label "**Package**" is aligned properly.

TC 03 – Verify that label with the name "**Browse**" is a button which is located at the end of TextBox with the name "**Source Folder**."

TC 04 – Verify that label with the name "**Browse**" is a button which is located at the end of TextBox with the name "**Package**."

TC 05 – Verify that the text box with the label "**Name**" is aligned properly.

TC 06 – Verify that the label "**Modifiers**" consists of 4 radio buttons with the name public, default, private, protected.

TC 07 – Verify that the label "**Modifiers**" consists of 4 radio buttons which are aligned properly in a row.

TC 08 – Verify that the label "**Superclass**" under the label "**Modifiers**" consists of a dropdown which must be properly aligned.

TC 09 – Verify that the label "**Superclass**" consists of a button with the label "**Browse**" on it which must be properly aligned.

TC 10 – Verify that clicking on any radio button the default mouse pointer must be changed to the hand mouse pointer.

TC 11 – Verify that user must not be able to type in the dropdown of "**Superclass**."

TC 12 – Verify that there must be a proper error generated if something has been mistakenly chosen.

TC 13 - Verify that the error must be generated in the RED color wherever it is necessary.

TC 14 – Verify that proper labels must be used in the error messages.

TC 15 – Verify that the single radio buttons must be selected by default every time.

TC 16 – Verify that the TAB button must be work properly while jumping on another field next to previous.

TC 17 – Verify that all the pages must contain the proper title.

TC 18 – Verify that the page text must be properly aligned.

TC 19 – Verify that after updating any field a proper confirmation message must be displayed.

TC 20 - Verify that only 1 radio button must be selected, and more than single checkboxes may be selected.