**UFT Assignment**

### **1. Introduction to UFT:**

**Q1: What is UFT (Unified Functional Testing)? How is it different from other test automation tools like Selenium or QTP?**

**Unified Functional Testing (UFT)** is a comprehensive software testing tool developed by Micro Focus. It automates functional and regression testing of desktop, web, and mobile applications. UFT was formerly known as QuickTest Professional (QTP) before being integrated and expanded to support a wider range of applications.

**Differences between UFT, Selenium, and QTP:**

| **Feature** | **UFT** | **Selenium** | **QTP (Legacy)** |
| --- | --- | --- | --- |
| **License** | Paid, commercial | Open-source | Paid, commercial |
| **Technology Support** | Supports desktop, web, API, and mobile apps | Web applications only | Mainly desktop and limited web apps |
| **Scripting Language** | VBScript | Java, Python, C#, etc. | VBScript |
| **Object Recognition** | Advanced, supports UI and non-UI objects | Relies on web elements | Basic object recognition |
| **IDE** | Integrated with Micro Focus ALM | Requires external IDEs like Eclipse | Standalone tool |
| **Ease of Use** | User-friendly with built-in libraries | Requires advanced coding knowledge | User-friendly |

**Q2: List the key features of UFT. Explain how it supports functional, regression, and GUI testing.**

**Key Features of UFT:**

1. **Cross-Platform Support:**
   * Automates desktop, web, and mobile applications across multiple environments and platforms.
2. **Object Repository:**
   * Centralized storage for objects, which allows reuse and easy maintenance of scripts.
3. **Keyword-Driven Testing:**
   * Provides a simple interface for non-technical testers to create scripts using keywords.
4. **Integration with ALM (Application Lifecycle Management):**
   * Seamless integration with Micro Focus ALM for test management and reporting.
5. **API Testing:**
   * Supports web services and API testing to validate back-end systems.
6. **Smart Object Recognition:**
   * Advanced algorithm to recognize dynamic and complex UI elements.
7. **Test Reporting:**
   * Generates detailed and customizable test execution reports.
8. **VBScript Support:**
   * Uses VBScript as its scripting language, offering simplicity and flexibility.

**Support for Testing Types:**

* **Functional Testing:** UFT validates that application functionality works as intended by simulating user interactions.
* **Regression Testing:** Automates the retesting of applications to ensure that new code changes have not introduced defects.
* **GUI Testing:** Tests the graphical user interface by interacting with visual components like buttons, text boxes, and menus.

**Q3: What are the different types of objects that UFT can recognize? Give examples of each type.**

UFT can recognize several types of objects, categorized as follows:

1. **Standard Objects:**
   * Objects commonly found in applications.
   * **Examples:** Buttons, Text Boxes, Drop-downs, Checkboxes.
2. **Custom Objects:**
   * Objects created specifically for a particular application.
   * **Examples:** Custom-built grids, controls in proprietary frameworks.
3. **Web Objects:**
   * Objects on web pages.
   * **Examples:** Links, Web Tables, Web Buttons, Web Edit Fields.
4. **Windows Objects:**
   * Objects in desktop applications.
   * **Examples:** Dialog Boxes, Windows, File Menus.
5. **Mobile Objects:**
   * Objects in mobile applications.
   * **Examples:** Mobile Buttons, Mobile Edit Boxes, List Views.
6. **API Objects:**
   * Used for API testing.
   * **Examples:** REST and SOAP web service endpoints.
7. **Bitmap/Visual Objects:**
   * Objects identified through their visual properties (bitmaps).
   * **Examples:** Static images, dynamic icons.

By maintaining a rich object repository and advanced recognition techniques, UFT ensures reliable and scalable test automation.

### **2. Creating and Running a Basic Test in UFT:**

**Q4: Create a simple test in UFT to open the Notepad application, type a text message, and save the file. Include the steps to record and run the test.**

**Steps to Record and Run the Test:**

1. **Open UFT:**
   * Launch UFT and ensure that the "Windows Applications" add-in is loaded.
2. **Start a New Test:**
   * Click on **File → New → Test**.
   * Choose **GUI Test**.
3. **Enable Recording:**
   * Click on the **Record** button or press F6.
4. **Record the Actions:**
   * In the **Record and Run Settings**, choose **Record on any Windows-based application**.
   * Click **OK** to start recording.
   * Perform the following steps:
     + Open Notepad (you can press Win+R, type notepad, and hit Enter).
     + Type a message (e.g., "Hello, this is a UFT test").
     + Save the file by clicking **File → Save**, provide a file name (e.g., TestFile.txt), and select a location.
     + Close Notepad.
5. **Stop Recording:**
   * Click the **Stop** button or press F4.
6. **Enhance the Script (Optional):**
   * Add checkpoints or parameterize the script as needed.
7. **Run the Test:**
   * Click the **Run** button or press F5.
   * Observe the execution and view the results in the Test Results Viewer.

**Script (Example):**

vbscript

Copy code

SystemUtil.Run "notepad.exe"

Window("Notepad").Type "Hello, this is a UFT test."

Window("Notepad").WinMenu("Menu").Select "File;Save"

Window("Save As").WinEdit("File name:").Set "TestFile.txt"

Window("Save As").WinButton("Save").Click

Window("Notepad").Close

**Q5: Write a simple UFT script to open a web browser, navigate to a website (e.g., www.google.com), and perform a Google search.**

**Steps to Create and Run the Test:**

1. **Open UFT:**
   * Launch UFT and ensure the **Web** add-in is loaded.
2. **Start a New Test:**
   * Click on **File → New → Test**.
   * Choose **GUI Test**.
3. **Write the Script:**
   * Manually enter the script in the **Editor** tab of UFT.

**UFT Script Example:**

vbscript

Copy code

' Open the browser and navigate to Google

SystemUtil.Run "iexplore.exe", "http://www.google.com"

' Wait for the Google homepage to load

Browser("Google").Page("Google").Sync

' Enter a search query

Browser("Google").Page("Google").WebEdit("q").Set "UFT automation testing"

' Click the Google Search button

Browser("Google").Page("Google").WebButton("Google Search").Click

' Wait for search results to load

Browser("Google").Page("Search Results").Sync

' Close the browser

Browser("Google").Close

1. **Run the Test:**
   * Click the **Run** button or press F5.
   * Monitor the execution in real-time and review the results in the Test Results Viewer.

### **3. Object Repository and Object Identification:**

**Q6: What is an object repository in UFT? Explain the difference between "Local Object Repository" and "Shared Object Repository."**

**Object Repository in UFT:**

An **Object Repository (OR)** in UFT is a storage location where UFT keeps information about the objects it interacts with during a test. Each object in the application is identified by its properties (e.g., name, ID, class, etc.), which are stored in the Object Repository. UFT uses these properties to locate and interact with objects during test execution.

**Types of Object Repositories in UFT:**

1. **Local Object Repository (LOR):**
   * Stores object information specific to a single test.
   * Automatically created when you record or add objects during the test.
   * Modifications to the repository apply only to that test.
   * Easier to use for smaller tests or where object reusability is not needed.
2. **Shared Object Repository (SOR):**
   * Stores object information in an external file (with .tsr extension) that can be shared among multiple tests.
   * Requires manual association with tests that use it.
   * Suitable for large projects with reusable components.

**Comparison Table:**

| **Feature** | **Local Object Repository (LOR)** | **Shared Object Repository (SOR)** |
| --- | --- | --- |
| **Scope** | Specific to one test | Can be used across multiple tests |
| **Storage** | Stored within the test itself | Stored as a separate .tsr file |
| **Reusability** | Limited | High |
| **Maintenance** | Difficult for large projects | Easier to maintain |
| **Best Use Case** | Small and independent tests | Large projects with reusable components |

**Q7: Explain the concept of "Object Identification" in UFT. How does UFT recognize objects on the application being tested?**

**Object Identification in UFT:**

Object Identification refers to the process by which UFT recognizes and interacts with the objects in an application during test execution. UFT identifies objects based on their properties and uses these properties to create a unique representation of each object in the Object Repository.

**How UFT Recognizes Objects:**

1. **Mandatory Properties:**
   * UFT first uses mandatory properties (defined in the Object Identification settings) to recognize an object.
   * Example: For a button, properties like html tag and name may be mandatory.
2. **Assistive Properties:**
   * If mandatory properties are insufficient to uniquely identify an object, UFT uses assistive properties.
   * Example: For a text box, additional properties like index or html id might be considered.
3. **Ordinal Identifiers:**
   * If both mandatory and assistive properties fail to uniquely identify an object, UFT uses an ordinal identifier (e.g., location or index) as the last resort.
4. **Smart Identification:**
   * A fallback mechanism where UFT uses a combination of base filter properties and optional filter properties to identify an object if the regular method fails.

**Steps for Object Identification in UFT:**

1. **Spy the Object:**
   * Use the **Object Spy** tool to inspect the object’s properties.
   * View the mandatory, assistive, and ordinal identifier properties.
2. **Store the Object:**
   * Add the object to the Object Repository manually or automatically during recording.
3. **Use the Object in Scripts:**
   * Refer to the object by its logical name in the Object Repository.
4. **Execution:**
   * During test execution, UFT matches the properties stored in the Object Repository with those in the application to locate and interact with the object.

**Example of Object Identification:**

For a login button on a web page:

* **Mandatory Properties:**
  + html tag = "BUTTON"
  + name = "Login"
* **Assistive Properties:**
  + class = "btn-primary"
* **Ordinal Identifier (if needed):**
  + index = 1

### **4. Checkpoints and Verification:**

**Q10: What are checkpoints in UFT? Write a script to add a "Text Checkpoint" to verify that a specific text appears on a web page.**

**Checkpoints**

Checkpoints are verification points in UFT that validate whether the expected results match the actual application behavior during test execution. They allow testers to verify properties of objects, text, tables, databases, and more.

**Script to Add a Text Checkpoint:**

Here is an example of adding a "Text Checkpoint" to verify specific text on a web page:

**Steps to Create a Text Checkpoint:**

1 Open UFT and create a new test.

2 Record or manually write a script to navigate to the desired web page.

3 Add a "Text Checkpoint" using the **Insert → Checkpoint → Text Checkpoint** option.

4 Highlight the text on the web page to verify and configure the checkpoint.

**Script Example:**

vbscript

Copy code

' Open the browser and navigate to the website

SystemUtil.Run "iexplore.exe", "https://example.com"

' Wait for the page to load

Browser("Example").Page("Example").Sync

' Add a Text Checkpoint to verify if "Welcome to Example" appears on the page

Browser("Example").Page("Example").Check CheckPoint("Welcome to Example")

' Close the browser

Browser("Example").Close

**Q11: Explain the difference between "Standard Checkpoints" and "Database Checkpoints" in UFT. Give an example of when you would use each.**

|  | **Standard Checkpoints** | **Database Checkpoints** |
| --- | --- | --- |
| **Purpose** | Verifies properties of objects (e.g., text, images, buttons). | Validates data from a database or query results. |
| **Scope** | GUI elements like buttons, links, text, etc. | Backend data or database table contents. |
| **Use Case** | Verify if a login button is enabled. | Verify if a customer record exists in a database. |
| **Configuration** | Created by selecting an object and defining properties to check. | Created by connecting to a database and specifying queries. |
| **Example** | Check if "Submit" button is visible. | Check if "Order ID 12345" exists in the orders table. |

**Example Scenarios:**

1. **Standard Checkpoint Example:**
   * **Scenario:** Verify that the "Login" button is enabled on the login page.
   * **Checkpoint Type:** Standard Checkpoint.

vbscript

Copy code

Browser("App").Page("Login").WebButton("Login").Check CheckPoint("Enabled")

1. **Database Checkpoint Example:**
   * **Scenario:** Verify that the order total for "Order ID 12345" in the database is $100.
   * **Checkpoint Type:** Database Checkpoint.
   * **Steps:**
     + Use **Insert → Checkpoint → Database Checkpoint** in UFT.
     + Configure the database connection and SQL query (e.g., SELECT Total FROM Orders WHERE OrderID = '12345').
   * **Example Query Result Validation:**
     + UFT will validate if the result from the query matches the expected value.

**Q12: How can you handle dynamic objects using UFT? Explain with an example of handling dynamic buttons that change text based on user interactions.**

**Handling Dynamic Objects in UFT:**

Dynamic objects have properties (e.g., IDs, text, or attributes) that change during runtime. UFT provides several strategies to handle such objects:

1. **Regular Expressions in Object Properties:**
   * Use wildcards or regular expressions to match patterns in dynamic properties.
2. **Descriptive Programming:**
   * Directly describe objects in the script without relying on the Object Repository.
3. **Smart Identification:**
   * Enable Smart Identification to use alternative properties when primary identification fails.

**Example: Handling a Dynamic Button:**

**Scenario:** A button changes its text dynamically based on user interactions (e.g., from "Submit" to "Processing..." to "Completed").

1. **Using Regular Expressions:**
   * In the Object Repository:
     + For the button's **text** property, use a regular expression like Submit|Processing...|Completed.
   * Example Script:

vbscript

Copy code

Browser("App").Page("Form").WebButton("DynamicButton").Click

1. **Using Descriptive Programming:**

vbscript

Copy code

' Click the dynamic button using its partial text

Browser("App").Page("Form").WebButton("innertext:=.\*Submit.\*").Click

1. **Smart Identification:**
   * Enable Smart Identification for the button in the Object Repository.
   * If UFT cannot find the exact object, it will use additional properties to locate it.

### **5. Parameterization:**

**Q13: What is parameterization in UFT? Why is it important for automating tests? Demonstrate how to parameterize a test using input data (e.g., user credentials for a login page).**

Parameterization is the process of running a test with multiple sets of data by passing different input values during test execution. This helps to avoid hardcoding data in the script and supports **data-driven testing**, making the tests reusable and efficient.

**Parameterization Importance**

1. **Data-Driven Testing:**
   * Allows testing with multiple data sets.
   * Reduces the need to write separate scripts for each test case.
2. **Improves Flexibility:**
   * Makes scripts dynamic and adaptable.
3. **Enhances Reusability:**
   * The same script can be used for different input data scenarios.

**Demonstration: Parameterizing Login Credentials**

1. **Scenario:**
   * Login to a web application using different sets of usernames and passwords.
2. **Steps:**
   * Create a test in UFT.
   * Add input parameters for Username and Password.
   * Replace hardcoded values in the script with parameters.
3. **Script Example:**

vbscript

Copy code

' Open the browser and navigate to the login page

SystemUtil.Run "iexplore.exe", "https://example-login.com"

' Parameterize username and password

Browser("Login").Page("Login").WebEdit("Username").Set DataTable("Username", dtGlobalSheet)

Browser("Login").Page("Login").WebEdit("Password").Set DataTable("Password", dtGlobalSheet)

Browser("Login").Page("Login").WebButton("Login").Click

' Validate login success

If Browser("Login").Page("Dashboard").Exist Then

Reporter.ReportEvent micPass, "Login", "Login was successful."

Else

Reporter.ReportEvent micFail, "Login", "Login failed."

End If

' Close the browser

Browser("Login").Close

1. **Adding Data in the Data Table:**
   * In UFT, go to **Data → Data Table**.
   * Add Username and Password columns with test data.

**Q14: Create a test that accepts input parameters (e.g., username and password) from an Excel file and performs a login using that data.**

1. **Steps to Use Excel for Parameterization:**
   * Save the test data in an Excel file (e.g., TestData.xlsx).
   * Use the **DataTable.Import** method to load the Excel file into UFT's Data Table.
2. **Sample Excel File:**

| **Username** | **Password** |
| --- | --- |
| user1 | pass1 |
| user2 | pass2 |

1. **Script to Perform Login:**

vbscript

Copy code

' Import the Excel file into the Data Table

DataTable.Import "C:\Path\To\TestData.xlsx"

' Loop through each row of the data table

RowCount = DataTable.GetRowCount

For i = 1 To RowCount

DataTable.SetCurrentRow(i)

' Open the browser and navigate to the login page

SystemUtil.Run "iexplore.exe", "https://example-login.com"

' Enter credentials from the Excel file

Browser("Login").Page("Login").WebEdit("Username").Set DataTable("Username", dtGlobalSheet)

Browser("Login").Page("Login").WebEdit("Password").Set DataTable("Password", dtGlobalSheet)

Browser("Login").Page("Login").WebButton("Login").Click

' Validate login success

If Browser("Login").Page("Dashboard").Exist Then

Reporter.ReportEvent micPass, "Login", "Login for user " & DataTable("Username", dtGlobalSheet) & " was successful."

Else

Reporter.ReportEvent micFail, "Login", "Login for user " & DataTable("Username", dtGlobalSheet) & " failed."

End If

' Close the browser

Browser("Login").Close

Next

**Q15: What are the different types of parameters available in UFT (e.g., test, action, and data table parameters)? Explain their use with examples**

1. **Test Parameters:**
   * Defined at the test level and can be passed as arguments when calling the test.
   * **Use Case:** Pass environment-specific values like URLs or credentials.

**Example:**

* + Define a test parameter AppURL.
  + Access it in the script: SystemUtil.Run Environment("AppURL").

1. **Action Parameters:**
   * Defined at the action level and used to pass data between actions in a test.
   * **Use Case:** Pass login credentials from one action to another.

**Example:**

* + Create input parameters Username and Password in Action1.
  + Access them in the script: Parameter("Username").

1. **Data Table Parameters:**
   * Use the Data Table (Global or Local sheets) to store multiple sets of input data.
   * **Use Case:** Perform data-driven testing.

**Example:**

* + Access data from the Data Table: DataTable("Username", dtGlobalSheet).

1. **Environment Parameters:**
   * Used to define global values that are environment-specific, such as file paths or URLs.
   * **Use Case:** Share values across multiple tests.

**Example:**

* + Define an environment parameter FilePath.
  + Access it in the script: Environment.Value("FilePath").

**Comparison of Parameter Types:**

| **Parameter Type** | **Scope** | **Example Use Case** |
| --- | --- | --- |
| **1.Test Parameters** | Entire test | Pass URLs or environment data. |
| **2. Action Parameters** | Specific to an action | Pass credentials between actions. |
| **3.Data Table Parameters** | Multiple iterations of data | Data-driven testing with user data. |
| **4.Environment Parameters** | Shared across tests | Define global paths or configurations. |

### **6. Actions and Function Libraries:**

**Q16: What is an action in UFT? How does it help in organizing your test scripts? Create an example of a reusable action for logging into a web application.**

**Action**

An **action** in UFT is a modular part of a test that encapsulates a set of operations or steps. Actions are used to divide test scripts into smaller, reusable components, improving test organization, reusability, and maintainability.

**Benefits of Actions:**

1. **Reusability:**
   * Actions can be reused in multiple tests, reducing redundancy.
2. **Modularity:**
   * Organizes the test into logical sections, making it easier to debug and maintain.
3. **Parameterization:**
   * Allows passing input and output values between actions.
4. **Improved Readability:**
   * Test scripts are easier to read and understand.

**Creating a Reusable Action for Login:**

1. **Scenario:**
   * Create a reusable action that performs a login operation.
2. **Steps:**
   * Record or manually create steps for logging into the application.
   * Convert the action to reusable by right-clicking the action and selecting **Action Properties → Reusable Action**.
3. **Reusable Login Action Example:**

vbscript

Copy code

' Action Name: Login

' Input Parameters: Username, Password

Browser("App").Page("Login").WebEdit("Username").Set Parameter("Username")

Browser("App").Page("Login").WebEdit("Password").Set Parameter("Password")

Browser("App").Page("Login").WebButton("Login").Click

' Validate Login

If Browser("App").Page("Dashboard").Exist Then

Reporter.ReportEvent micPass, "Login", "Login successful."

Else

Reporter.ReportEvent micFail, "Login", "Login failed."

End If

1. **Calling the Reusable Action in Another Test:**

vbscript

Copy code

' Call the reusable login action

RunAction "Login", oneIteration, "user1", "pass1"

**Q17: Explain the concept of "Function Libraries" in UFT. How do you create and associate a function library with your test?**

**What is a Function Library in UFT?**

A **Function Library** in UFT is a file that contains custom VBScript functions. It enables creating reusable code that can be used across multiple tests, enhancing code modularity and reusability.

**How to Create a Function Library:**

1. **Create a New Function Library:**
   * Go to **File → New → Function Library**.
   * Write custom functions in the editor.
2. **Save the Function Library:**
   * Save the file with a .vbs extension (e.g., CommonFunctions.vbs).
3. **Associate the Function Library with a Test:**
   * Go to **File → Settings → Resources → Add**.
   * Add the path of the .vbs file.
4. **Call Functions from the Library:**
   * Use the function name in the test script.

**Example: Function Library**

File: CommonFunctions.vbs

vbscript

Copy code

' Function to add two numbers

Function AddNumbers(a, b)

AddNumbers = a + b

End Function

**Associating and Calling the Function in a Test Script:**

vbscript

Copy code

' Call the function from the associated function library

Dim result

result = AddNumbers(5, 10)

Reporter.ReportEvent micPass, "Addition", "The sum is: " & result

**Q18: Write a simple function in a UFT function library that accepts two numbers as inputs and returns their sum. Call this function from your test script**.

1. **Create the Function Library:**
   * Save the following code in a file named MathFunctions.vbs.

vbscript

Copy code

' Function to calculate the sum of two numbers

Function CalculateSum(num1, num2)

CalculateSum = num1 + num2

End Function

1. **Associate the Function Library with Your Test:**
   * Add the MathFunctions.vbs file under **File → Settings → Resources → Add**.
2. **Test Script to Call the Function:**

vbscript

Copy code

' Call the CalculateSum function

Dim number1, number2, sum

number1 = 20

number2 = 30

sum = CalculateSum(number1, number2)

' Report the result

Reporter.ReportEvent micPass, "Sum Calculation", "The sum of " & number1 & " and " & number2 & " is: " & sum

**Output in the UFT Test Report:**

* **Step:** Sum Calculation
* **Result:** The sum of 20 and 30 is: 50

### **7. Descriptive Programming:**

**Q19: What is Descriptive Programming in UFT, and when would you use it? Write a UFT script using descriptive programming to click a button on a webpage (e.g., a "Submit" button).**

**Descriptive Programming** in UFT allows testers to identify and interact with objects dynamically without relying on the Object Repository. Instead of storing object properties in the repository, you directly specify object properties and their values in the script.

**When to Use Descriptive Programming:**

1. When the Object Repository becomes too large, impacting performance.
2. When the objects are dynamic and their properties change frequently.
3. For reusable scripts where objects might not always be pre-defined in the repository.
4. When you need to work with applications with a large number of similar objects.

**Example Script Using Descriptive Programming:** To click a "Submit" button on a webpage:

vbscript

Copy code

' Launch the browser and navigate to a webpage

SystemUtil.Run "iexplore.exe", "https://example.com"

' Use descriptive programming to identify and click the Submit button

Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").WebButton("name:=Submit").Click

' Verify the action

If Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").Exist(10) Then

Reporter.ReportEvent micPass, "Submit Button", "Successfully clicked the Submit button."

Else

Reporter.ReportEvent micFail, "Submit Button", "Failed to click the Submit button."

End If

**Q20: Explain the syntax for Descriptive Programming in UFT. Write a script that uses** **descriptive programming to interact with a web element based on its properties (e.g., link** **text, tagname, etc.).**

The syntax for Descriptive Programming involves specifying the object type and a set of properties (property-value pairs) to identify the object uniquely.

**Syntax:**

vbscript

Copy code

ObjectType("PropertyName:=PropertyValue", "PropertyName2:=PropertyValue2")

* **ObjectType:** The type of the object, such as Browser, Page, WebEdit, WebButton.
* **PropertyName:=PropertyValue:** A property and its value to identify the object. Multiple properties can be specified to make the identification more precise.

**Example Script Using Descriptive Programming:** To interact with a web element based on properties:

vbscript

Copy code

' Launch the browser and navigate to a webpage

SystemUtil.Run "iexplore.exe", "https://example.com"

' Enter text into a text box using descriptive programming

Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").WebEdit("name:=username").Set "TestUser"

' Click a link using descriptive programming

Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").Link("text:=Forgot Password").Click

' Verify the action

If Browser("title:=.\*Example.\*").Page("title:=.\*Password Recovery.\*").Exist(10) Then

Reporter.ReportEvent micPass, "Link Click", "Successfully navigated to the Password Recovery page."

Else

Reporter.ReportEvent micFail, "Link Click", "Failed to navigate to the Password Recovery page."

End If

**Q21: How does UFT handle dynamic objects with Descriptive Programming? Provide an example using a dynamic link or button.**

Dynamic objects often have properties that change at runtime, such as IDs or names generated dynamically. UFT can handle such objects by:

1. **Using Regular Expressions**:
   * Use a pattern to match dynamic properties.
   * Example: id:=btn\_\d+ matches IDs like btn\_123, btn\_456.
2. **Using Index Property**:
   * Use the index property to identify an object based on its position in the object hierarchy.
3. **Using a Combination of Properties**:
   * Use multiple stable properties to uniquely identify an object.

**Example: Handling Dynamic Buttons Using Regular Expressions**

vbscript

Copy code

' Launch the browser and navigate to a webpage

SystemUtil.Run "iexplore.exe", "https://example.com"

' Use regular expressions to identify a dynamic button

Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").WebButton("id:=submit\_\d+").Click

' Verify the action

If Browser("title:=.\*Example.\*").Page("title:=.\*Success.\*").Exist(10) Then

Reporter.ReportEvent micPass, "Dynamic Button", "Successfully clicked the dynamic Submit button."

Else

Reporter.ReportEvent micFail, "Dynamic Button", "Failed to click the dynamic Submit button."

End If

**Example: Handling Dynamic Links Using Index Property**

vbscript

Copy code

' Launch the browser and navigate to a webpage

SystemUtil.Run "iexplore.exe", "https://example.com"

' Use index property to identify the second link on the page

Browser("title:=.\*Example.\*").Page("title:=.\*Example.\*").Link("index:=1").Click

' Verify the action

If Browser("title:=.\*Example.\*").Page("title:=.\*Next Page.\*").Exist(10) Then

Reporter.ReportEvent micPass, "Dynamic Link", "Successfully clicked the second link."

Else

Reporter.ReportEvent micFail, "Dynamic Link", "Failed to click the second link."

End If

### **8. Synchronization and Wait Statements:**

**Q22: Why is synchronization important in UFT? What are the different synchronization techniques you can use to make sure your script waits for an element to be available?**

**Synchronization Importance**

Synchronization ensures that the script waits for the application under test to be ready before performing actions. Without synchronization, UFT might try to interact with objects that are not yet loaded, leading to test failures.

**Synchronization Techniques in UFT:**

1. **Default Synchronization Timeout:**
   * UFT has a default synchronization timeout (20 seconds by default). If an object is not found within this time, the test fails.
   * You can configure it under **File → Settings → Run → Object Synchronization Timeout**.
2. **Sync Method:**
   * Ensures the browser or page is fully loaded before performing actions.
   * **Use Case:** Wait for a web page to load.
   * **Example:**

vbscript

Copy code

Browser("App").Page("Login").Sync

1. **Wait Method:**
   * Pauses the script for a specified amount of time (in seconds).
   * **Use Case:** Temporarily halt execution to give the application time to process.
   * **Example:**

vbscript

Copy code

Wait 5 ' Waits for 5 seconds

1. **WaitProperty Method:**
   * Waits for an object’s property to achieve a specific value within a timeout period.
   * **Use Case:** Wait for a button to become enabled.
   * **Example:**

vbscript

Copy code

Browser("App").Page("Dashboard").WebButton("Submit").WaitProperty "enabled", True, 10000

1. **Exist Method:**
   * Checks if an object exists within a specified timeout.
   * **Use Case:** Wait until a specific object appears.
   * **Example:**

vbscript

Copy code

If Browser("App").Page("Dashboard").WebButton("Next").Exist(10) Then

' Perform action

End If

**Q23: Write a script that uses the Sync method and Wait method to ensure UFT waits for a page to load before performing actions like clicking a button.**

**Example Script Using Sync and Wait Methods:**

vbscript

Copy code

' Open the browser and navigate to a website

SystemUtil.Run "iexplore.exe", "https://example.com"

' Wait for the page to load using the Sync method

Browser("Example").Page("Example").Sync

' Wait explicitly for 5 seconds using the Wait method

Wait 5

' Click the "Login" button after ensuring the page is loaded

Browser("Example").Page("Example").WebButton("Login").Click

' Close the browser

Browser("Example").Close

**Q24: How would you handle synchronization issues when testing a slow application or a page with dynamic content?**

**Handling Synchronization Issues in Slow or Dynamic Applications:**

1. **Dynamic Waits Using WaitProperty:**
   * Use the WaitProperty method to wait until specific properties (e.g., visibility, enablement) of objects meet desired conditions.
   * **Example:**

vbscript

Copy code

Browser("App").Page("Loading").WebElement("Status").WaitProperty "innertext", "Ready", 30000

1. **Increasing the Default Synchronization Timeout:**
   * Increase the default timeout for object synchronization in UFT settings for slow-loading applications.
   * **Steps:**
     + Go to **File → Settings → Run → Object Synchronization Timeout**.
2. **Using Exist Method:**
   * Use the Exist method to check for object availability within a timeout.
   * **Example:**

vbscript

Copy code

If Browser("App").Page("Dashboard").WebButton("Submit").Exist(20) Then

Browser("App").Page("Dashboard").WebButton("Submit").Click

End If

1. **Smart Synchronization with Conditional Loops:**
   * Implement a custom loop to poll for object availability.
   * **Example:**

vbscript

Copy code

For i = 1 To 20

If Browser("App").Page("Dashboard").WebElement("Status").Exist Then

Exit For

End If

Wait 1 ' Wait for 1 second before checking again

Next

1. **Smart Identification:**
   * Enable **Smart Identification** in the object repository. If primary object identification fails, UFT uses backup properties to find the object.
2. **Handling Ajax or Dynamic Elements:**
   * Wait for Ajax-based elements to load fully by checking specific properties like innertext or visible.

**Example:**

vbscript

Copy code

Browser("App").Page("SearchResults").WebElement("LoadingSpinner").WaitProperty "visible", False, 15000

1. **Combination of Techniques:**
   * Use a mix of Sync, Wait, WaitProperty, and Exist to handle synchronization comprehensively.

### **9. Error Handling and Recovery:**

**Q25: How can you add exception handling in UFT to handle pop-ups or alerts that appear unexpectedly during the test execution?**

In UFT, exception handling can be implemented using various mechanisms to deal with unexpected situations like pop-ups, alerts, or application errors during test execution. Below are the key approaches:

### **1. Using Recovery Scenarios**

**What is a Recovery Scenario?**

A **Recovery Scenario** in UFT is a mechanism to handle unexpected events and errors during test execution, such as pop-ups, alerts, or application crashes. UFT can perform predefined actions to recover from such situations and continue the test.

**Steps to Create a Recovery Scenario for Handling Pop-ups:**

1. **Open Recovery Scenario Manager:**
   * Go to **Resources → Recovery Scenario Manager**.
2. **Create a New Scenario:**
   * Click **New Scenario** and follow the wizard.
3. **Define the Trigger Event:**
   * Select the type of event to handle (e.g., pop-up window or application crash).
   * For handling pop-ups, choose **Pop-up window**.
4. **Specify the Recovery Operation:**
   * Define the action UFT should perform when the pop-up appears:
     + Click a button (e.g., "OK" or "Cancel").
     + Close the window.
     + Call a custom function.
5. **Post-Recovery Test Run Options:**
   * Specify whether to continue or restart the test after handling the exception.
6. **Associate the Scenario with Your Test:**
   * In the test settings, go to **File → Settings → Recovery → Add** and attach the recovery scenario.

**Example: Recovery Scenario to Handle a Browser Alert Pop-up**

* **Trigger Event:** Browser pop-up appears.
* **Recovery Operation:** Click "OK" on the alert.
* **Post-Recovery:** Continue the test.

### **2. Using VBScript Error Handling with** On Error Resume Next

**What is VBScript Error Handling?**

UFT supports VBScript error handling using the On Error Resume Next statement, which allows the test to continue executing even after an error occurs.

**Example Script for Handling Alerts Using VBScript:**

vbscript

Copy code

' Enable error handling

On Error Resume Next

' Perform an action that may trigger a pop-up

Browser("App").Page("Dashboard").WebButton("Submit").Click

' Check if an alert exists and handle it

If Browser("App").Dialog("Alert").Exist Then

Browser("App").Dialog("Alert").WinButton("OK").Click

Reporter.ReportEvent micPass, "Pop-up Handling", "Alert handled successfully."

Else

Reporter.ReportEvent micInfo, "Pop-up Handling", "No alert found."

End If

' Disable error handling

On Error GoTo 0

### **3. Using Custom Functions**

For advanced or reusable error handling, you can create a custom function.

**Example: Function to Handle Alerts**

vbscript

Copy code

Function HandleAlert()

If Browser("App").Dialog("Alert").Exist Then

Browser("App").Dialog("Alert").WinButton("OK").Click

Reporter.ReportEvent micPass, "Alert Handling", "Alert handled successfully."

Else

Reporter.ReportEvent micInfo, "Alert Handling", "No alert found."

End If

End Function

**Call the Function in Your Test Script:**

vbscript

Copy code

Browser("App").Page("Dashboard").WebButton("Submit").Click

HandleAlert

### **4. Using** Exist **Method for Conditional Error Handling**

**Example: Handling Pop-ups with Timeout**

vbscript

Copy code

' Check for a pop-up with a timeout

If Browser("App").Dialog("Alert").Exist(10) Then

Browser("App").Dialog("Alert").WinButton("OK").Click

Reporter.ReportEvent micPass, "Pop-up Handling", "Pop-up handled successfully."

Else

Reporter.ReportEvent micInfo, "Pop-up Handling", "No pop-up found."

End If

### **5. Using** RecoveryScenario **Programmatically**

Recovery scenarios can also be added programmatically during runtime.

**Example: Add Recovery Scenario During Test Execution**

vbscript

Copy code

' Load the recovery scenario file

Recovery.Load "C:\RecoveryScenario\HandlePopUp.qrs"

' Enable the recovery scenario

Recovery.Activate

' Perform test steps

Browser("App").Page("Dashboard").WebButton("Submit").Click

' Deactivate the recovery scenario after use

Recovery.Deactivate

### **Best Practices for Exception Handling in UFT**

1. **Prioritize Predictable Errors:**
   * Handle frequently occurring errors (e.g., alerts or specific pop-ups) with direct scripts.
2. **Use Recovery Scenarios for Unpredictable Errors:**
   * Use recovery scenarios to manage rare or unpredictable errors.
3. **Log All Exceptions:**
   * Use Reporter.ReportEvent to log details of handled exceptions for debugging.
4. **Optimize Wait Times:**
   * Avoid long wait times; use conditional methods like Exist or WaitProperty.
5. **Regularly Update Scenarios:**
   * Modify recovery scenarios as the application evolves.

### **10. Test Results and Reporting:**

**Q26: Explain how UFT generates test results. How do you view and analyze the test results after running a test in UFT?**

**How UFT Generates Test Results:**

1. **Execution Log:**
   * During test execution, UFT automatically logs each action, checkpoint, and error.
2. **Test Results Viewer:**
   * After the test execution, UFT generates a detailed results report in the **Test Results Viewer**, which is stored in the specified results folder.
3. **Report Contents:**
   * The report includes:
     + **Test Summary:** Start and end time, total steps, passed/failed steps.
     + **Step Details:** Status of individual steps (Pass/Fail/Warning), executed actions, and checkpoints.
     + **Screenshots:** Optional snapshots of the application at critical steps.
     + **Error Details:** Messages for failures or unexpected conditions.

**Viewing and Analyzing Test Results:**

1. **Open Results:**
   * After a test run, the results open automatically. You can also access them from **File → Open → Test Results**.
2. **Test Results Viewer Features:**
   * **Summary Tab:** Provides an overview of the test run.
   * **Steps Tab:** Details of every executed step, including the input data, expected results, and actual results.
   * **Checkpoints Tab:** Displays the status of each checkpoint.
3. **Debugging a Failing Test:**
   * Analyze failed steps and review error messages.
   * Check screenshots to understand the state of the application at the time of failure.

**Q27: What is the difference between the "Test Results" tab and the "Run-Time Data Table" in UFT? How would you use them to debug a failing test?**

| **Feature** | **Test Results Tab** | **Run-Time Data Table** |
| --- | --- | --- |
| **Purpose** | Shows the overall test execution status and detailed step results. | Displays the data used or generated during the test run. |
| **Content** | Step-by-step execution details, including status, messages, and screenshots. | Input/output data for iterations of a test. |
| **Debugging Use** | Analyze failed steps, review error messages, and identify errors. | Verify that the correct data was used during execution. |
| **Accessibility** | Open via **Test Results Viewer** after test execution. | Accessed in UFT during or after the test run. |

**Debugging a Failing Test:**

1. Use the **Test Results Tab**:
   * Identify the failed step.
   * Review error messages and screenshots to understand the failure cause.
2. Use the **Run-Time Data Table**:
   * Verify input/output data for failed iterations.
   * Ensure that correct parameters or variables were used.

**Q28: Write a script that generates a custom report in UFT after executing a test case. This report should include test steps, status (pass/fail), and any relevant messages**

**Custom Report Script Example:**

This script generates a simple custom report in a .txt file after executing a test.

**Steps:**

1. Create a custom function for reporting.
2. Log each step's status (Pass/Fail) and message.
3. Save the report to a file.

**Script:**

vbscript

Copy code

' Initialize the custom report

Dim reportFile

reportFile = "C:\CustomTestReport.txt"

' Create or overwrite the file

Set fso = CreateObject("Scripting.FileSystemObject")

Set report = fso.CreateTextFile(reportFile, True)

report.WriteLine "Custom Test Report"

report.WriteLine "=================="

report.WriteLine "Test Run Time: " & Now

report.WriteLine

' Function to log test steps

Function LogStep(stepDescription, stepStatus, stepMessage)

Dim logEntry

logEntry = "Step: " & stepDescription & vbCrLf & \_

"Status: " & stepStatus & vbCrLf & \_

"Message: " & stepMessage & vbCrLf & \_

"--------------------------------" & vbCrLf

report.WriteLine logEntry

If stepStatus = "Pass" Then

Reporter.ReportEvent micPass, stepDescription, stepMessage

Else

Reporter.ReportEvent micFail, stepDescription, stepMessage

End If

End Function

' Example Test Case Execution

On Error Resume Next

' Step 1: Open browser

SystemUtil.Run "iexplore.exe", "https://example.com"

If Browser("Example").Exist(10) Then

LogStep "Open Browser", "Pass", "Browser opened successfully."

Else

LogStep "Open Browser", "Fail", "Browser did not open."

End If

' Step 2: Click login button

Browser("Example").Page("Home").WebButton("Login").Click

If Browser("Example").Page("Login").Exist(5) Then

LogStep "Navigate to Login Page", "Pass", "Login page displayed."

Else

LogStep "Navigate to Login Page", "Fail", "Login page not displayed."

End If

' Close the report

report.WriteLine "Test Execution Completed."

report.Close

' View the report file

Reporter.ReportEvent micInfo, "Custom Report", "Report saved to: " & reportFile

SystemUtil.Run "notepad.exe", reportFile

**Output in the Report File:**

markdown

Copy code

Custom Test Report

==================

Test Run Time: 01/01/2025 10:00:00

Step: Open Browser

Status: Pass

Message: Browser opened successfully.

--------------------------------

Step: Navigate to Login Page

Status: Fail

Message: Login page not displayed.

--------------------------------

Test Execution Completed.