

ROBOTIC PROCESS AUTOMATION

INDEX

Sr. No	Practical	Date	Sign
1	a. Create a simple sequence based project. b. Create a flowchart-based project. c. Create an UiPath Robot which can empty a folder in Gmail solely on basis of recording.		
2	a. Automate UiPath Number Calculation (Subtraction, Multiplication, Division of numbers). b. Create an automation UiPath project using different types of variables (number, datetime, Boolean, generic, array, data table)		
3	a. Create an automation UiPath Project using decision statements b. Create an automation UiPath Project using looping statements		
4	a. Automate any process using basic recording. b. Automate any process using desktop recording. c. Automate any process using web recording.		
5	a. Consider an array of names. We have to find out how many of them start with the letter "a". Create an automation where the number of names starting with "a" is counted and the result is displayed.		
6	a. Create an application automating the read, write and append operation on excel file. b. Automate the process to extract data from an excel file into a data table and vice versa		
7	a. Implement the attach window activity.		

	<p>b. Find different controls using UiPath.</p> <p>c. Demonstrate the following activities in UiPath:</p> <ul style="list-style-type: none"> i. Mouse (click, double click and hover) ii. Type into iii. Type Secure text 		
8	<p>a. Demonstrate the following events in UiPath:</p> <ul style="list-style-type: none"> i. Element triggering event ii. Image triggering event iii. System Triggering Event <p>b. Automate the following screen scraping methods using UiPath</p> <ul style="list-style-type: none"> i. Full Test ii. Native iii. OCR <p>c. Install and automate any process using UiPath with the following plug-ins:</p> <ul style="list-style-type: none"> i. Java Plugin ii. Mail Plugin iii. PDF Plugin iv. Web Integration v. Excel Plugin vi. Word Plugin vii. Credential Management 		
9	<p>a. Automate the process of send mail event (on any email).</p> <p>b. Automate the process of launching an assistant bot on a keyboard event.</p> <p>c. Demonstrate the Exception handing in UiPath.</p> <p>d. Demonstrate the use of config files in UiPath.</p>		
10	<p>a. Automate the process of logging and taking screenshots in UiPath.</p> <p>b. Automate any process using State Machine in UiPath</p> <p>c. Demonstrate the use of publish utility.</p> <p>d. Create and provision Robot using Orchestrator.</p>		

Practical 1A

AIM : Create a simple sequence based project.

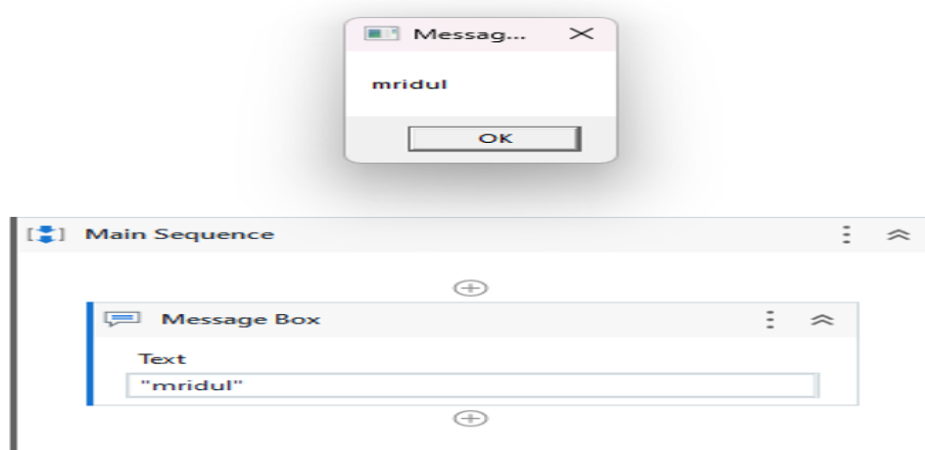
Theory:

A sequence is a basic automation process in RPA. It is used when the process is divided into steps and sequential. It is simpler and easier to use.

Steps:

1. Open UI path studio.
2. In uipath studio start page, under new project click on process
3. A dialog box appears, give a name to the process and select the language as C# and click on create.
4. Once a process is created, go to activity panel and add new sequence
5. Add name to the sequence and click on create. A sequence is created.
6. Once the sequence is created, add a message box.
7. Add the text you want to display and click on debug at the top ribbon.
8. This is how we create a simple sequence.

Output:



Practical 1B

AIM : Create a flowchart-based project.

Theory:

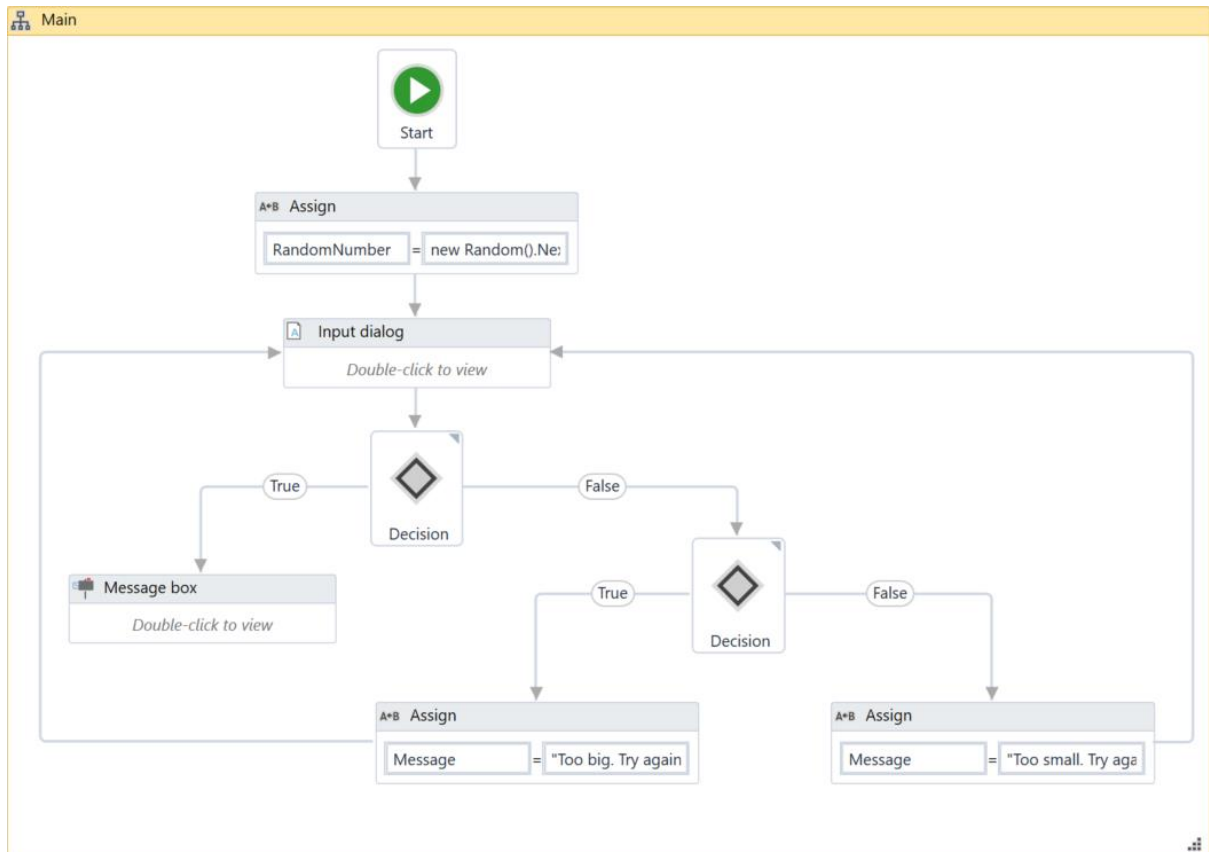
Flowcharts can be used in a variety of settings, from large jobs to small projects that you can reuse in other projects. The most important aspect of flowcharts is that, unlike sequences, they present multiple branching logical operators, that enable you to create complex business processes and connect activities in multiple ways.

Steps:

1. Create a blank process and from the **Design** tab, in the **File** group, select **New > Flowchart**. The **New Flowchart** window is displayed.
2. In the **Name** field type a name for the automation, such as "First Flowchart", and leave the default project location or add a subfolder. Click **Create**. The **Designer** panel is updated accordingly.
3. Create two **Int32** variables (RandomNumber, GuessNumber) and a **String** one (Message).
4. Set the default value of the Message variable to "Guess a number from 1 to 999."
The RandomNumber stores a random number between 1 and 999, GuessNumber stores the user's guess and Message stores the message that is going to be displayed to prompt the user.
5. Add an **Assign** activity to the **Designer** panel, and connect it to the **Start** node.
6. In the **Properties** panel, in the **To** field add the RandomNumber variable.
7. In the **Value** field, type new Random().Next(1,999).
8. Add an **Input Dialog** activity to the **Designer** panel and connect it to the **Assign** one.
9. In the **Properties** panel, in the **Label** field, add the Message variable.
10. In the **Result** field, add the GuessNumber variable. This activity asks and stores the user's guesses in the GuessNumber variable.
11. Add a **Flow Decision** activity and connect it to the **Input Dialog**. This activity enables you to tell the user if he correctly guessed the number or not.
12. In the **Properties** panel, in the **Condition** field, type GuessNumber = RandomNumber. This enables you to verify if the number added by the user is the same as the randomly-generated one.
13. Add a **Message Box** activity and connect it to the **True** branch of the **Flow Decision**.
14. In the **Properties** panel, in the **Text** field, type "Congratulations! You guessed correctly! The number was " + RandomNumber.ToString + ".". This is the message that is going to be displayed if the user correctly guessed the number.
15. Add a new **Flow Decision** activity and connect it to the **False** branch of the previously added **Flow Decision**.
16. In the **Properties** panel, in the **Condition** field, type GuessNumber > RandomNumber. This activity enables you to check if the number the user added is bigger than the randomly-generated one.
17. In the **DisplayName** field, type **Comparison**. This enables you to easily tell the difference between the two **Flow Decisions** used.
18. Add an **Assign** activity and connect it to the **True** branch of the **Comparison** activity.
19. In the **To** field, type the Message variable, and in the **Value** field, type a message indicating that the guess was too high, such as "Too big. Try again.".
20. Select the **Assign** activity and press Ctrl+C. The entire activity and its properties are copied to the Clipboard.
21. Press Ctrl + V. A duplicate of the previous **Assign** activity is displayed.

22. Connect it to the **False** branch of the **Comparison** activity and, in the **Properties** panel, in the **Value** field, type "Too small. Try again."
23. Connect the **Assign** activities created at steps 18-22 to the **Input Dialog**. A loop is created, asking the user to type a smaller or bigger number, until he guesses correctly.
The final project should look as in the screenshot below.

OUTPUT :



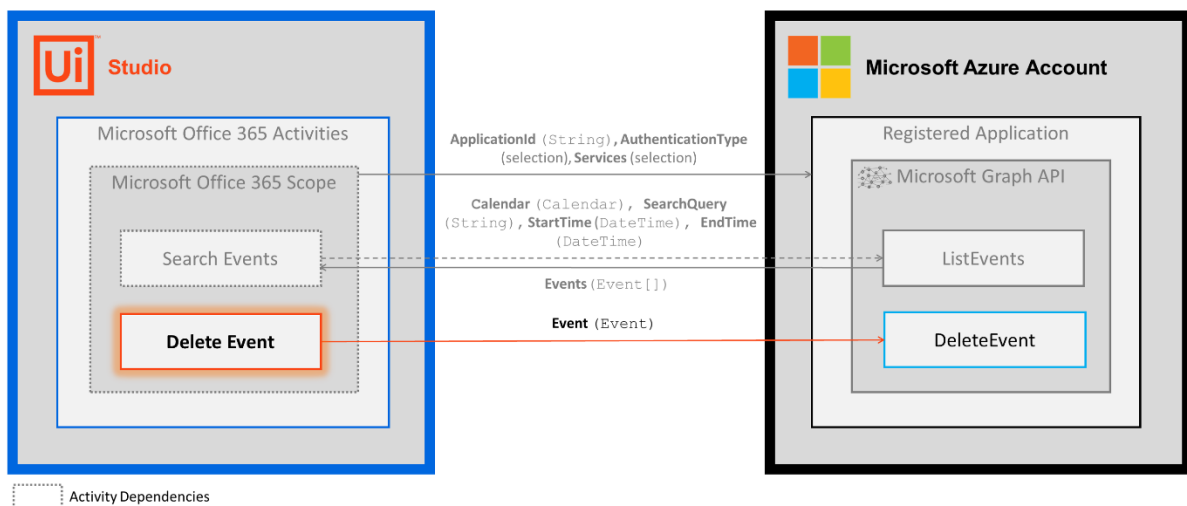
Practical 1C

AIM : An UiPath Robot which can empty a folder in Gmail solely on basis of recording.

STEPS :

1. Complete the Setup steps.
2. Add the Microsoft Office 365 Scope activity to your project.
3. Add an activity or run an external process that outputs a `Event` object (e.g., Search Events).
4. Add the **Delete Event** activity inside the **Microsoft Office 365 Scope** activity.
5. Enter values for the Input properties.
6. Run the activity.
 - Your input property values are sent to the DeleteEvent API.

OUTPUT :



Practical 2A

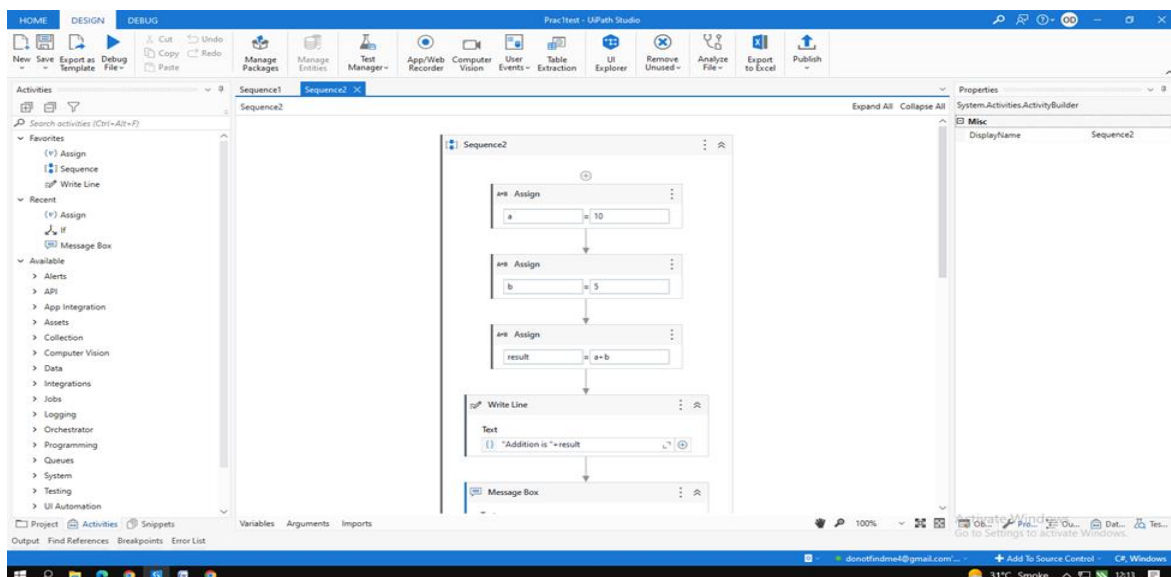
AIM : Automate UiPath Number Calculation (Subtraction, Multiplication, Division of numbers).

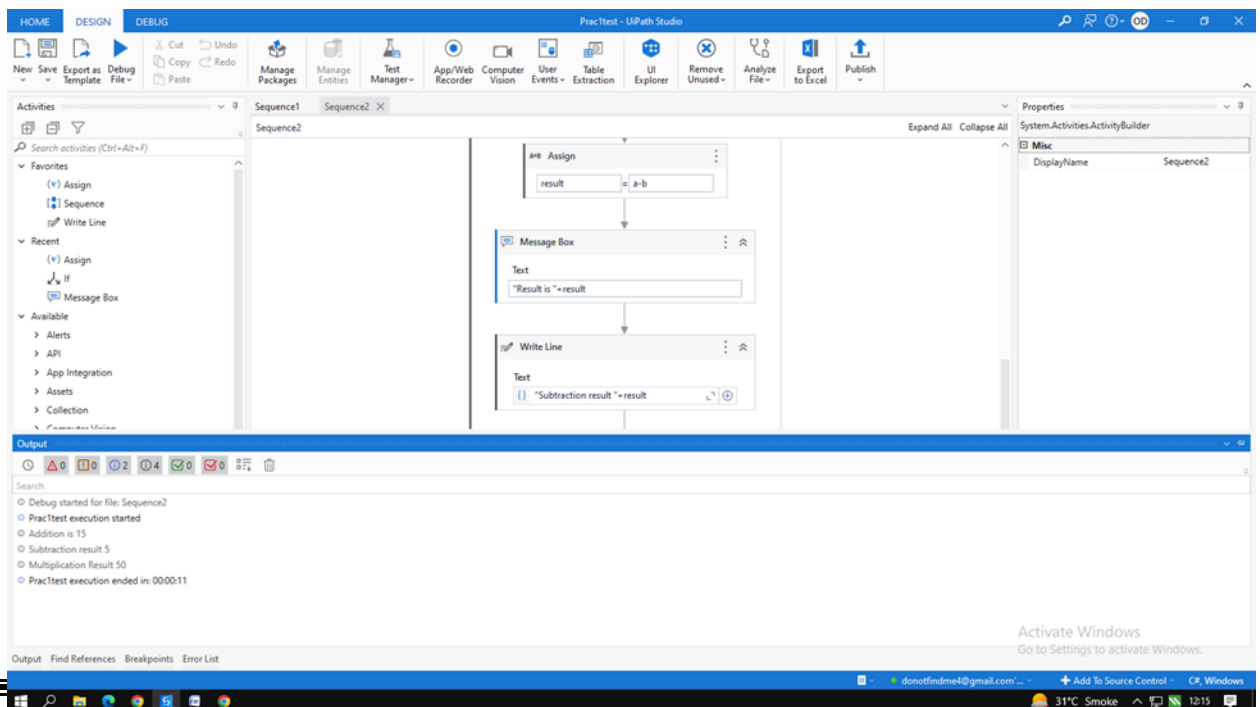
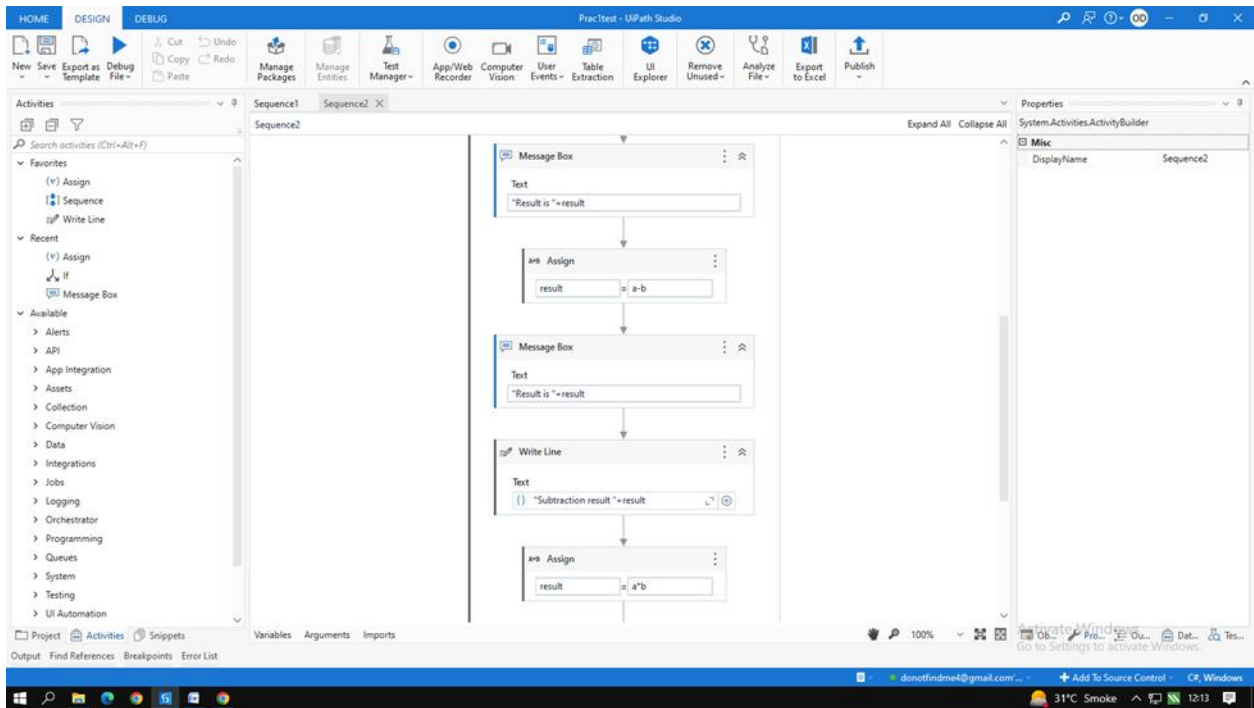
Steps:

1. Create a new sequence.
2. In the sequence, drag and drop an assign box to assign a variable. Assign a variable “a” with value 10.
3. On the variables tab, create the variable and assign the variable type as Int32.
4. Repeat steps 2 and 3 to create another variable “b” of integer type and give it the value 15.
5. Add a third assign box and create a variable “result” that will store the value of the operation between a and b. (a+b). This variable will also be integer.
6. Once done, drag and drop a write line box and add: “Addition is” + result since the output will be in string format.
7. Repeat steps from 5 and 6 thrice and change the operation as result=a-b, result=a*b, result=a/b.
8. Run the automation.

OUTPUT :

This will give the output of addition, subtraction, multiplication, division between variables a and b





Practical 2B

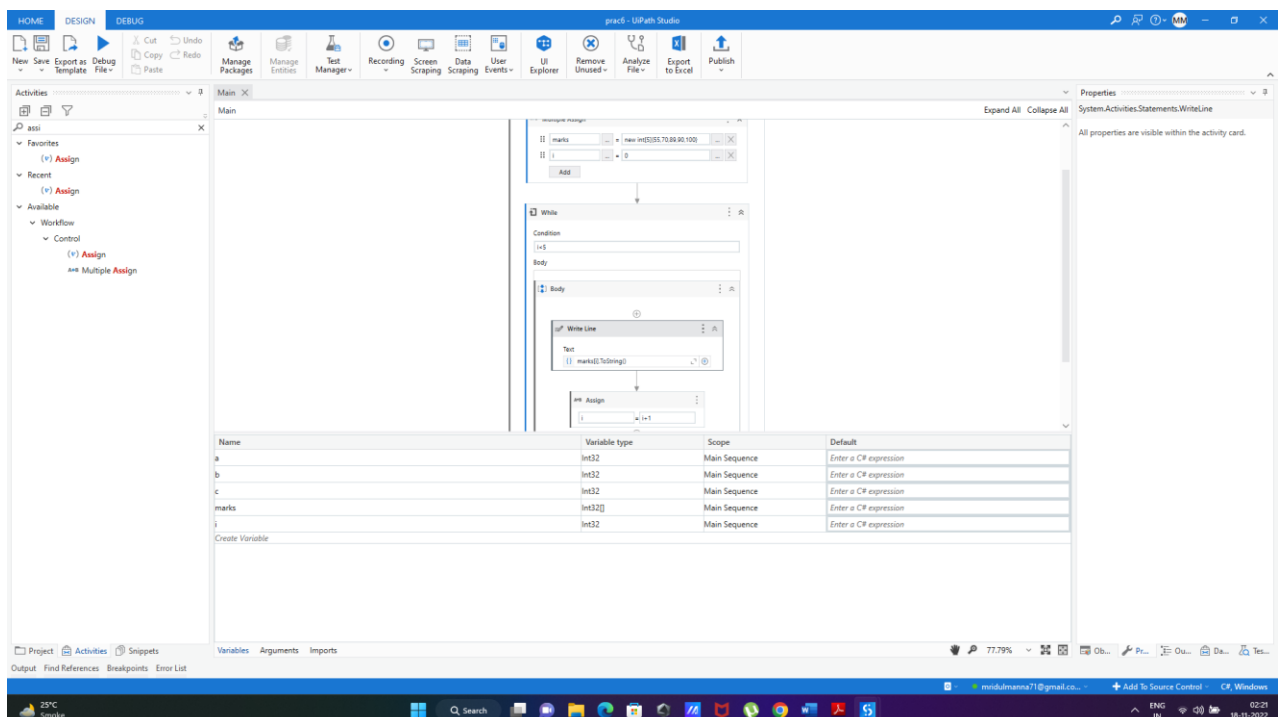
AIM :An automation UiPath project using different types of variables (number, datetime, Boolean, generic, array, data table)

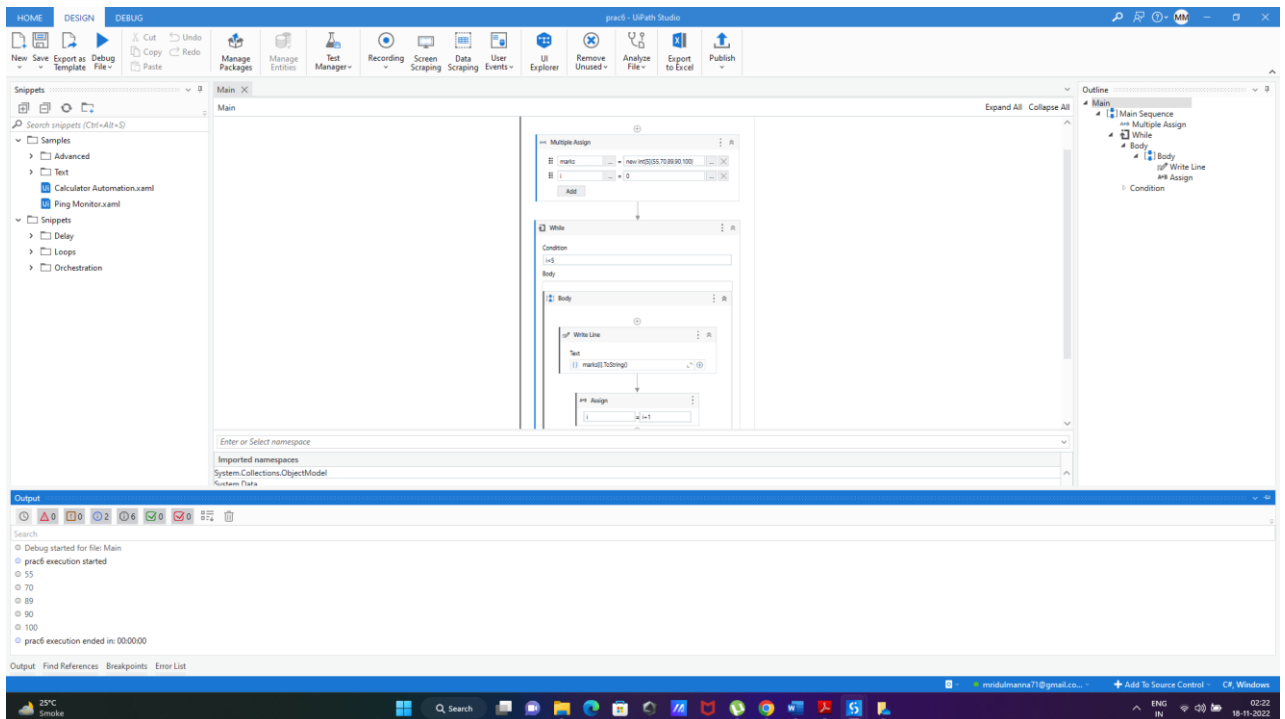
STEPS :

To create array

1. Create a new sequence.
2. Drag and drop multiple assign activity.
3. Create an array variable a where,
 - `marks = new int[5]{99,100,50,22,98}`
4. Once done, go to variable option and change the data type of the array to array of [T].
Click on the option and select the array type as int 32.
5. Create another variable `i=0` where `i` will be the iteration.
6. Drag and drop while activity. In the condition part add `i<5`
7. In the body drag and drop writeline and add `marks[i].ToString()` to display the array
8. Finally drag and drop a new assign and add `i=i+1` to increase the iteration by 1. This will display all the members of the array.
9. Run the automation and check the output.

OUTPUT :





Practical 3A

AIM : An automation UiPath Project using decision statements.

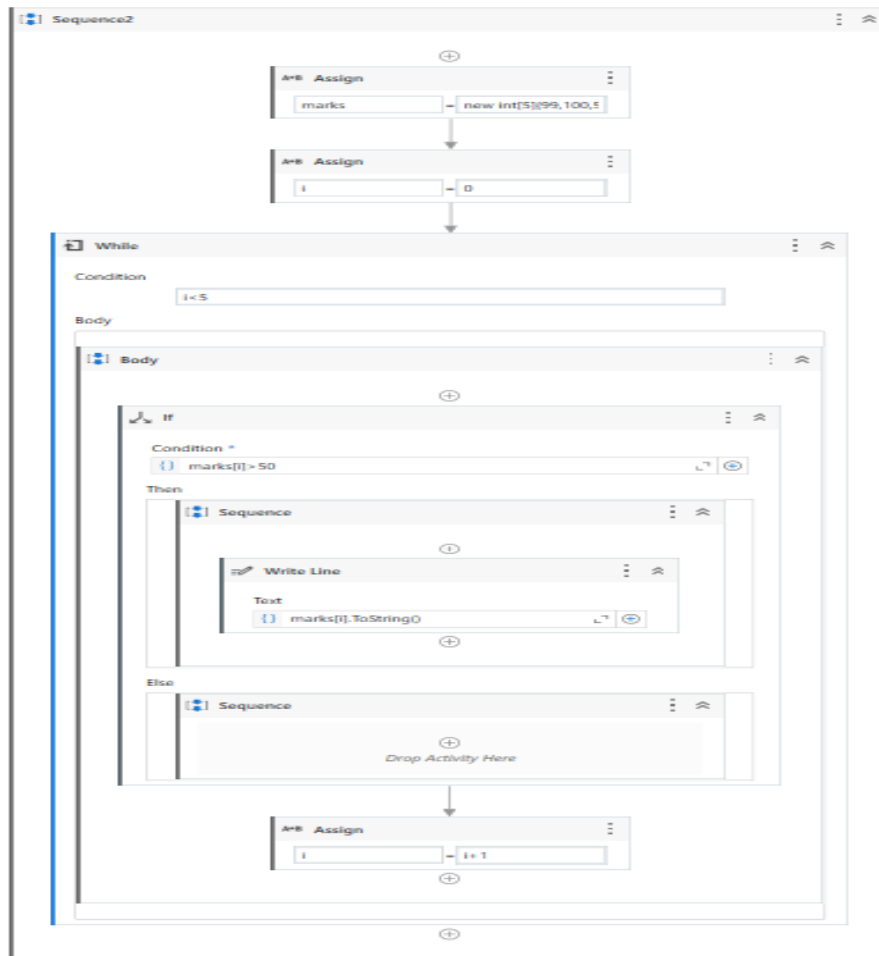
Theory:

The If activity contains a statement and two conditions. The first condition (the activity in the Then section) is executed if the statement is true, while the second one (the activity in the optional Else section) is executed if the statement is false.

If activities can be useful to make decisions based on the value of variables.

STEPS:

1. Create a new sequence.
2. Drag and drop multiple assign activity.
3. Create an array variable a where,
`marks = new int[5]{99,100,50,22,98}`
4. Once done, go to variable option and change the data type of the array to array of [T].
Click on the option and select the array type as int 32.
5. Create another variable i=0 where I will be the iteration.
6. Drag and drop while activity. In the condition part add `i<5`
7. In the body of the while activity, drag and drop the if activity and add the condition:
`Marks[i]>50`
8. In the “then” panel drag and drop writeline activity and add `marks[i].ToString()` to display the array
9. Finally drag and drop a new assign outside the if activity and add `i=i+1` to increase the iteration by 1. This will display all the members of the array.
10. Run the automation and check the output.



Output:

This automation will print the numbers in the array greater than 50.

- ① Debug started for file: Sequence2
- ② rpa1-2 execution started
- ③ 99
- ④ 100
- ⑤ 98
- ⑥ rpa1-2 execution ended in: 00:00:01

Practical 3B

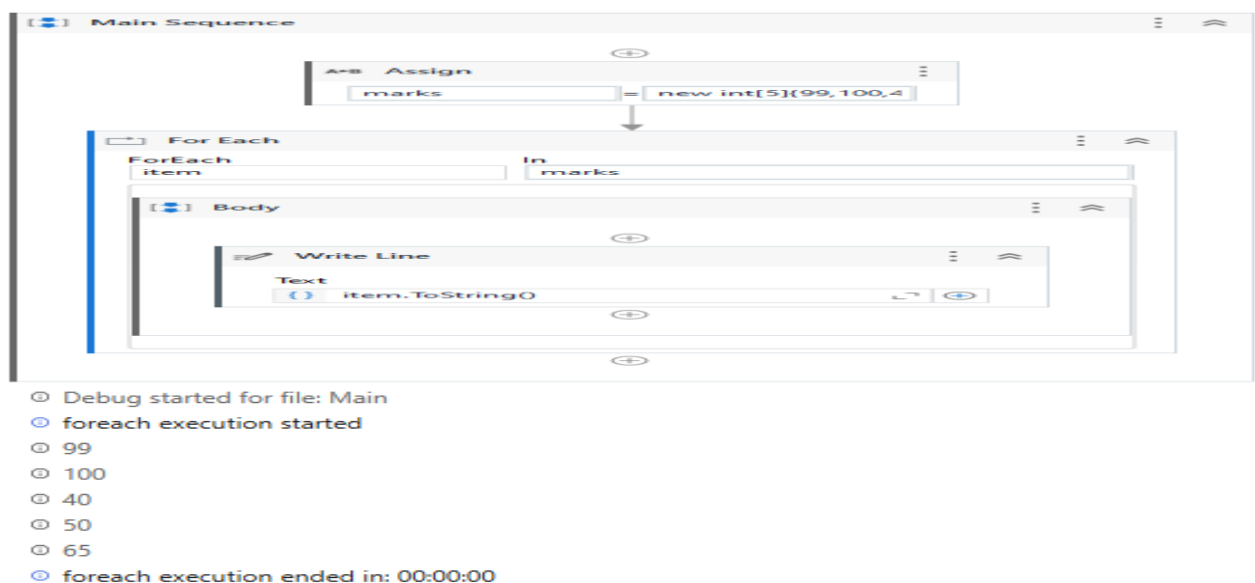
AIM : An automation UiPath Project using looping statements.

STEPS :

For-each activity

1. Create a new sequence.
2. Drag and drop assign activity.
3. Create an array variable marks where,
 - a. `marks = new int[5]{99,100,50,22,98}`
4. Once done, go to variable option and change the data type of the array to array of [T].
Click on the option and select the array type as int 32.
5. Drag and drop for each activity.
6. In the ForEach activity add the following statements:
 - In “for each” field add item
 - Add marks in “In” field
7. In the body drag and drop writeline and add `item.ToString()` to display the array
8. Run the automation and check the output.

OUTPUT : This automation will print an array using for each activity.



Practical 4A

AIM : Automate any process using basic recording.

Theory :

Recording is an important part of UiPath Studio, that can help you save a lot of time when automating your business processes. This functionality enables you to easily capture a user's actions on the screen and translates them into sequences.

These projects can be modified and parameterized so that you can easily replay and reuse them in as many other processes as you need.

All user interface elements are highlighted while you record, as you can see in the following screenshot, so that you can be sure the correct buttons, fields or menus are selected.

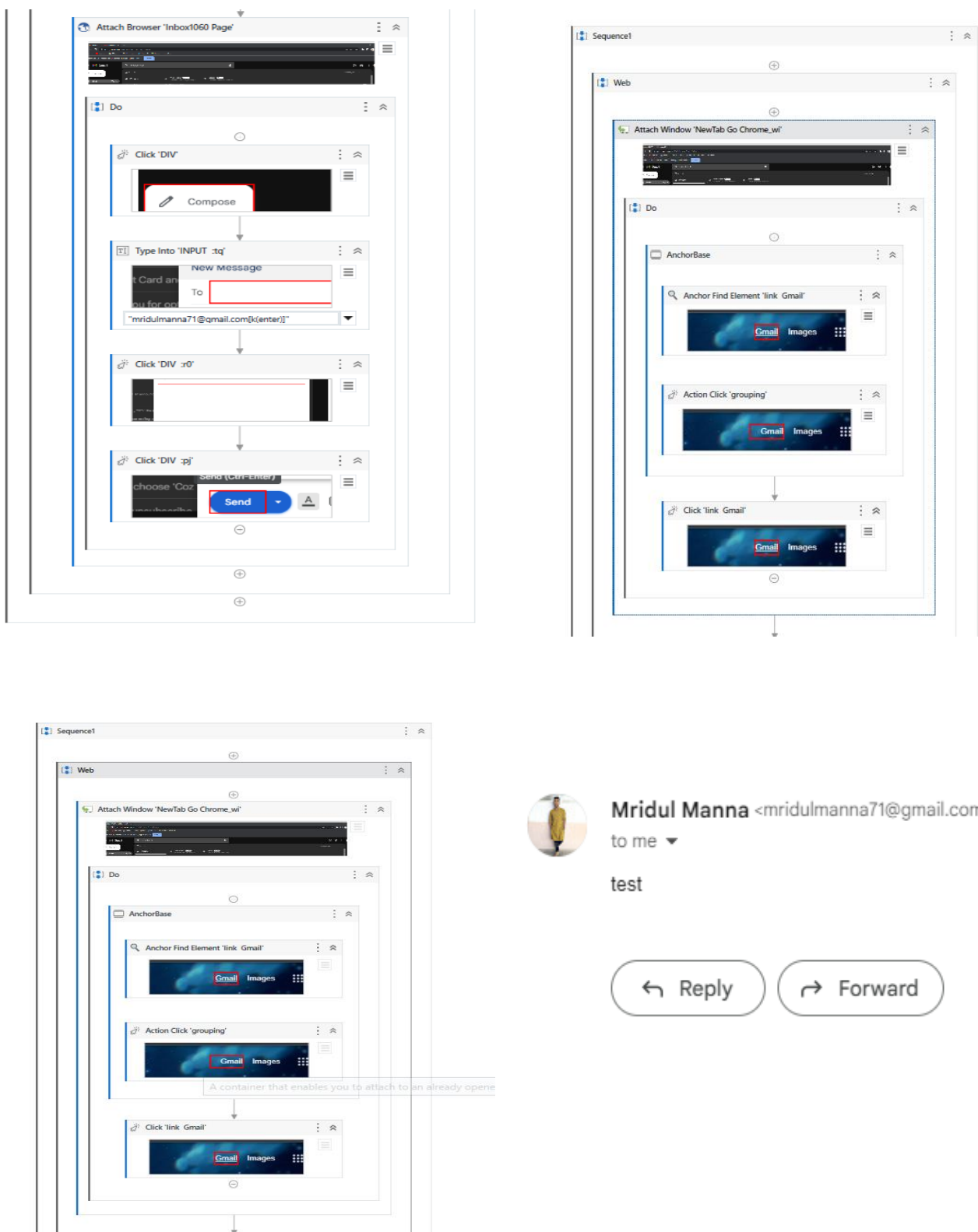
Steps:

Recording steps to send an email

1. Create a new sequence.
2. On the top in the ribbon, click on the recording option. A drop down will appear.
3. Select web recording option from the dropdown.
4. Open the Google chrome window.
5. Start recording
6. On the chrome window, click on gmail link
7. After clicking on gmail link, the gmail webpage opens. Click on compose.
8. After compose, a box appears to enter the mail.
9. Click on the To: option and enter the email address you want the mail to send to
10. After adding the receiver's email address, press enter
11. Click on the body of the email and add any text and click on send
12. Press esc and click on save and exit
13. Run the automation and check the output

Output:

This system will send an email automatically using automation.



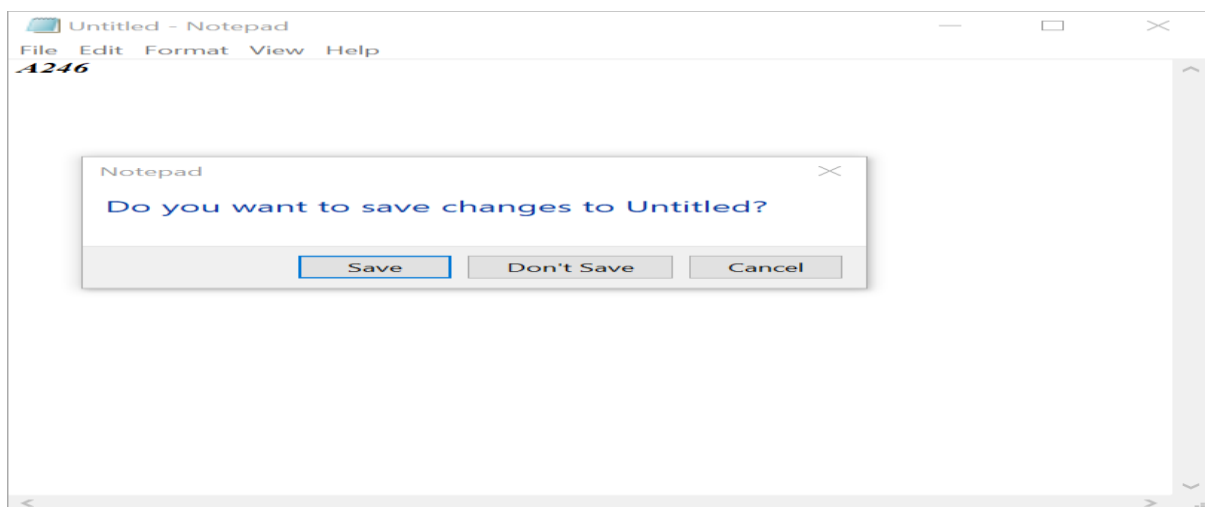
Practical 4B

AIM : Automate any process using desktop recording.

STEPS :

1. Open Notepad.
2. In UiPath Studio, create a new sequence.
3. a. In the **Design** ribbon tab, in the **Wizards** group, select **Record > Basic**. The **Basic Recording** toolbar is displayed and the main view is minimized.
b. In the **Design** ribbon tab, in the **Wizards** group, select **Record > Desktop**. The **Desktop Recording** toolbar is displayed and the main view is minimized.
4. In the **Wizards** group, click **Automatic Recorder**. The automating recording process starts.
5. In Notepad, click on the main panel. A pop-up window is displayed.
6. Type a custom text and press Enter. The string is displayed in Notepad.
7. From the **Format** menu, select **Font**. The **Font** window is displayed.
8. Select a different font style, such as Bold Italic, and click **OK**.
9. Press Esc two times. You exit the recording view and the saved project is displayed in the **Designer** panel.
10. Press F5. The automation is executed as expected.
11. Add an **Open Application** activity between **Excel Application Scope** and the Recording sequence.
12. Use **Indicate window on screen** to select the active **Notepad** window.
13. Place the Recording sequence inside the **Open Application** activity.
14. Add a **Close Application** activity after **Open Application**.
15. Use **Indicate window on screen** again to select the active **Notepad** window to be closed.
16. Make sure the **OffsetX** and **OffsetY** properties (**Cursor Position**) are empty.
What was added to the project should look as in the following screenshot.

OUTPUT :



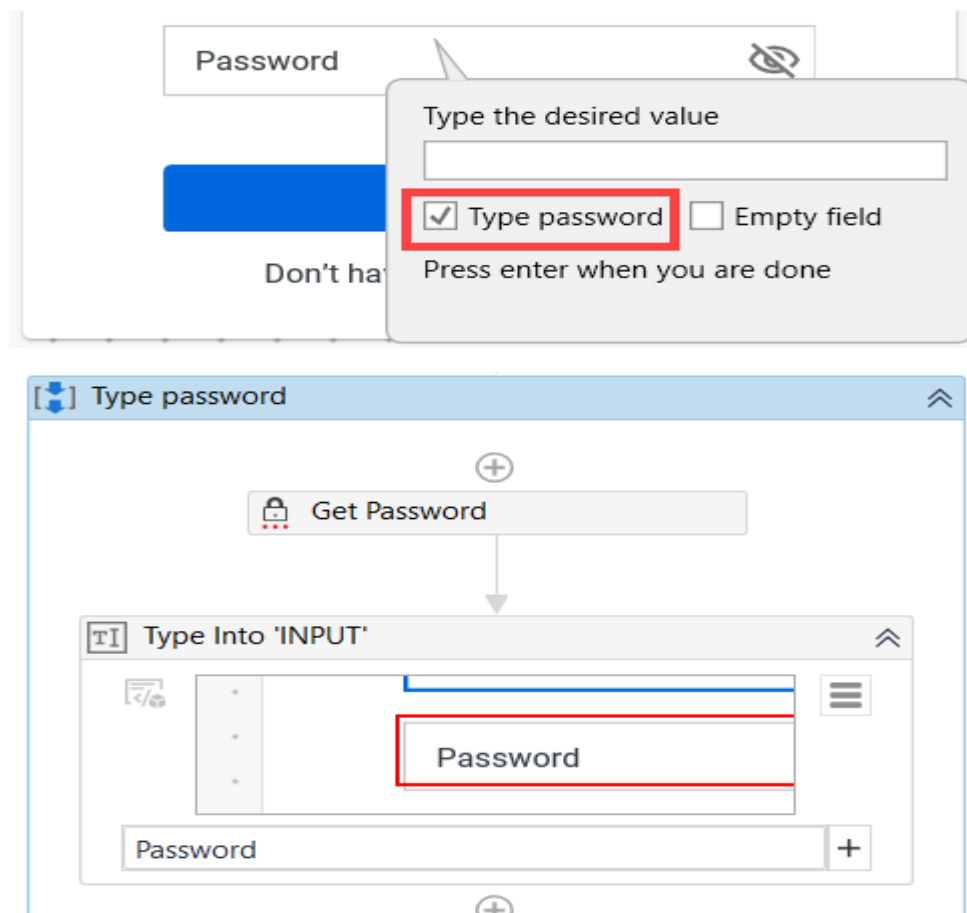
Practical 4C

AIM : Automate any process using web recording.

STEPS :

1. Open an Internet Explorer instance and navigate to <https://academy.uipath.com>.
2. In UiPath Studio, create a new sequence.
3. Add an **Open Browser** activity to the **Designer** panel.
4. Select the activity and, in the **Url** field, write <https://academy.uipath.com>.
5. In the **Design** tab, in the **Wizards** group, select **Recording > Web**. The **Web Recording** toolbar is displayed and the main view is minimized.
6. Click **Record**. The automating recording process starts.
7. In Internet Explorer, click **Login/ Sign up**, and then select **Continue with Email**.
8. Enter your email address and password.
9. Click **Login** and press Esc two times. The recording is saved and displayed in the **Designer** panel.
10. Close Internet Explorer manually.
11. In Studio, add a **Close Tab** activity as the last activity in the **Attach Browser** container.
12. Press F5. The automation is executed as expected.

OUTPUT :



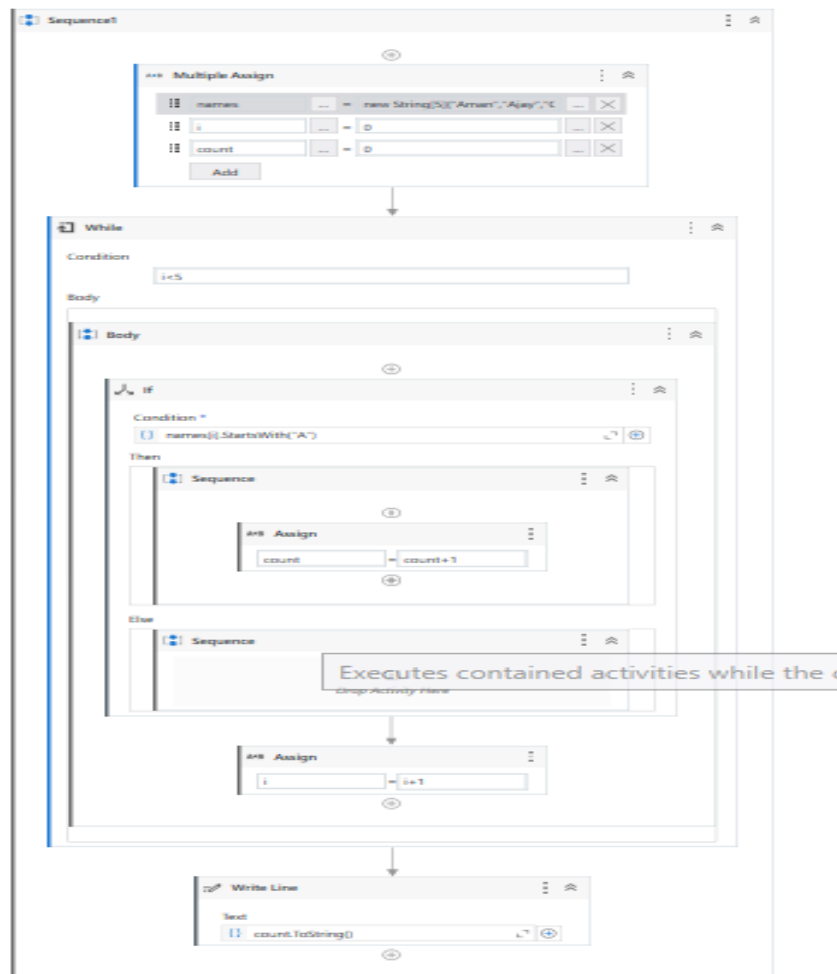
Practical 5A

AIM : Consider an array of names. We have to find out how many of them start with the letter "a". Create an automation where the number of names starting with "a" is counted and the result is displayed.

Steps:

1. Create a new sequence.
2. Drag and drop multiple assign activity
3. Create a string array and add the names you wish to add
`Names=New String[5]{“Aman”,”Ajay”,”Omkar”,”Ashish”,”Tejas”}`
4. Once the array is created, go to the variable tab and change the data type of the array to Array of [T] and select the datatype as string.
5. Once array is created, create a new variable i=0 for iteration and count=0 and select the datatype as Int32 for both the variables.
6. Once done, Drag and drop a while activity from activities panel.
7. Add condition as `i<5`
8. In the body panel of the while activity, drag and drop an if activity
9. Add the following condition in the if activity:
`Names[i].StartsWith(“A”)`
10. Drag and drop an assign activity in the “then” panel of the if activity and add
`Count=count+1`(This will increase the counter by 1 if the condition is true)
11. After the else panel in the if activity, drag and drop an assign activity and add
`i=i+1`
12. Outside the while loop in the main sequence, drag and drop writeline activity and add:
`Count.ToString()`.
13. Run the automation and check the output.

Output:



- ① Debug started for file: Sequence1
- ② record2 execution started
- ③ 3
- ④ record2 execution ended in: 00:00:00

Practical 6A

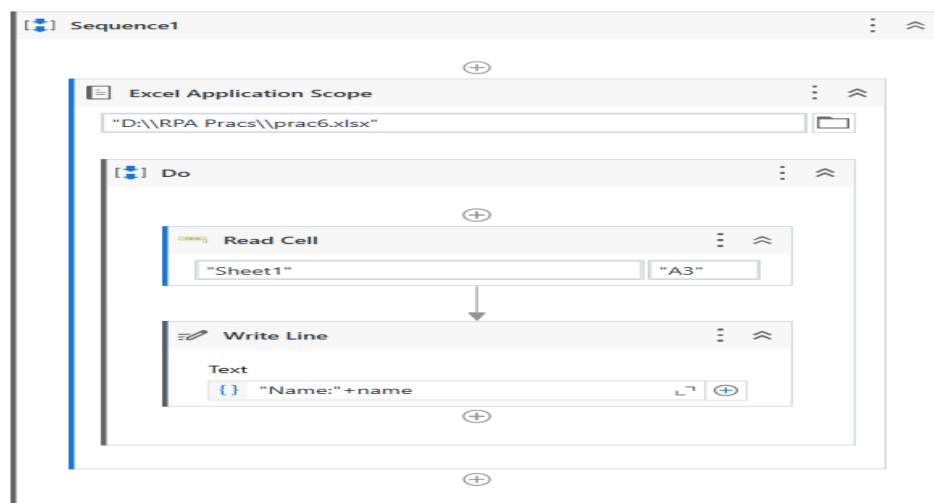
AIM : Create an application automating the read, write and append operation on excel file.

Steps:

1. Create a new sequence.
2. Create a excel file and add 2 columns of data

	A	B
1	fname	roll
2	abcd	1234
3	xyzw	5678
4	abcd	1234
5		

3. Drag and drop Excel application scope from the activities panel
4. In the application scope, browse the excel file in the system and add the path of the file.
5. In the DO panel of the excel application scope, Drag and drop read cell activity.
6. It will automatically take the sheet as “Sheet1” and cell as “A1” as the initial point .
7. Right click on the “Sheet1” and select create variable and add a variable as name and add any cell number you want to print.
8. Drag and drop a Writeline activity and add : “Name:”+name
9. Run the automation and check the output.



Output: The system will print the data available in cell A3

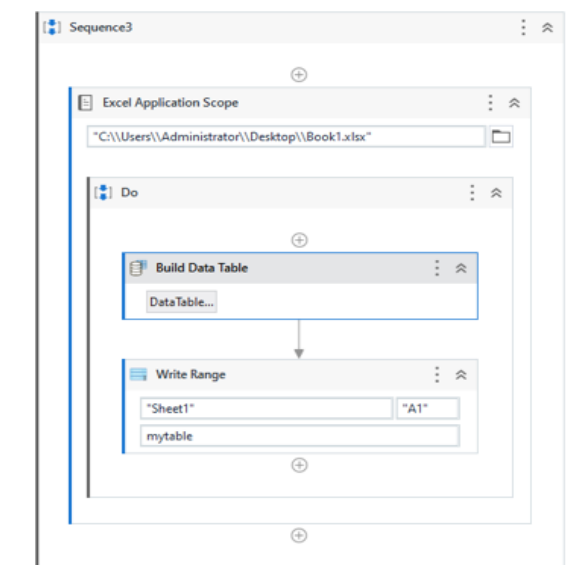
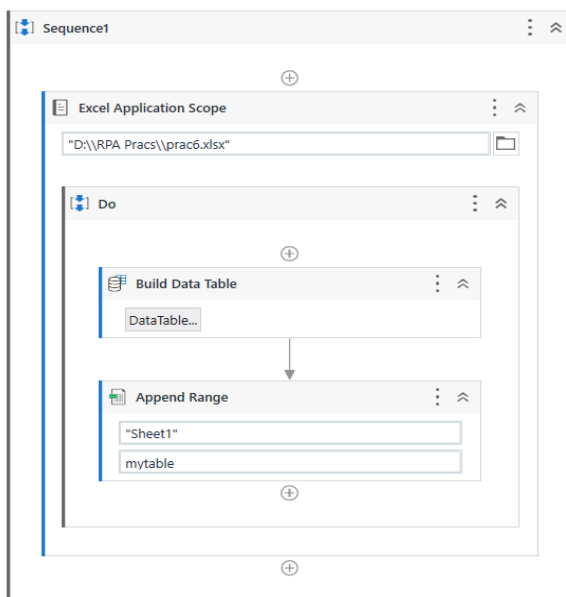
```
⊙ Debug started for file: Sequence1
⊙ record2 execution started
⊙ Name:xyzw
⊙ record2 execution ended in: 00:00:00
```

Practical 6B

AIM : Automate the process to extract data from an excel file into a data table and vice versa

Steps:

1. Create a new sequence.
2. Create a excel file and keep the file empty
3. Drag and drop Excel application scope from the activities panel
4. In the application scope, browse the excel file in the system and add the path of the file.
5. Drag and drop Build data table activity and add a datatable and click on create option
6. On the right side of the page, right click on the output option and create a variable “mytable”
7. In the DO panel of the excel application scope, Drag and drop append range / write range activity.
8. It will automatically take the sheet as “Sheet1” and cell as “A1” as the initial point if you use write range.
9. It will also give a text box to enter the data you wish to write in the excel file. Add the variable name as mytable.
10. Run the automation and check the output.



Output: This automation will write the data of the data table in the excel file and append the data the number of times you run the automation.

	A	B
1	abcd	1
2	xyzw	2
3	defg	2
4	abcd	1
5	xyzw	2
6		
7		

Practical 7A

AIM : Implement the attach window activity.

Attach Window

SUGGEST EDITS

UiPath.Core.Activities.WindowScope

A container that enables you to attach to an already opened window and perform multiple actions within it. This activity is also automatically generated when using the Desktop recorder.

Properties

Input

- **Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents. If both this property and the **Window** property are configured, the **Selector** property is used at design time, and the **Window** property is used at runtime.
- **Window** - The window to attach to. This field accepts only Window variables. If both this property and the **Selector** property are configured, the **Selector** property is used at design time, and the **Window** property is used at runtime.

Options

- **SearchScope** - The application window in which to search for the UI element defined by the Selector property.

Common

- **DisplayName** - The display name of the activity.
- **TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

Output

- **ApplicationWindow** - The found active window. This field supports only Window variables. When a Window variable is specified, SearchScope and Selector properties are ignored.

Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

Practical 7B

AIM : Find different controls using UiPath

STEPS :

1. Open Studio and create a new **Process**.
2. Drag a **Sequence** container in the **Workflow Designer**.
 - Create the following variable:

Variable Name	Variable Type
EditElement	UiPath.Core.UiElement
FormatElement	UiPath.Core.UiElement

3. Drag an **OpenApplication** activity inside the **Sequence** container.
4. Drag a **Find Element** activity below the **OpenApplication** activity.
 - In the **Properties** panel, select the **COMPLETE** option from the **WaitForReady** drop-down list.
 - Add the variable EditElement in the **FoundElement** field.
5. Place a **Find Relative** activity below the **Find Element** activity.
 - In the **Properties** panel, add the value 20 in the **OffsetX** field.
 - Select the option **TopRight** from the **Position** drop-down list.
 - Add the variable EditElement in the **Element** field.
 - Add the variable FormatElement in the **RelativeElement** field.
6. Place a **Click** activity below the **Find Relative** activity.
 - In the **Properties** panel, add the variable FormatElement in the **Element** field.
 - Select the **None** option from the **KeyModifiers** drop-down list.
7. Place another **Click** activity below the first **Click** activity.
 - Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:

Drag an **Activate** activity below the **Click** activity.

- Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:

Place a **Set Focus** activity below the **Activate** activity.

- Inside the activity, click the **Indicate on screen** option, and select the **Size** menu option. The GIF below shows all the steps you need to follow:

4. Drag a **Send Hotkey** activity below the **Set Focus** activity.

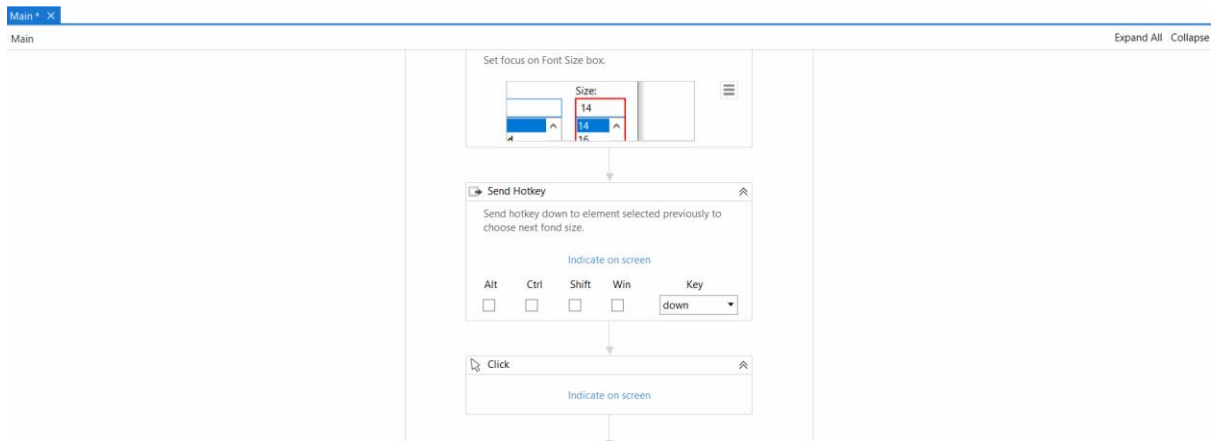
- In the **Key** field, type the value down.

- In the **Properties** panel, select the check box for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the **None** option from the **KeyModifiers** drop-down list.
- Select the check box for the **SpecialKey** option. This indicates that you are using a special key in the keyboard shortcut.

11. Place a **Click** activity below the **Send Hotkey** activity.

- Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:
 - Run the process. The automation opens a new Notepad file, navigates through the menu

OUTPUT :



Practical 7C

AIM : Demonstrate the following activities in UiPath:

i.MOUSE

UiPath Studio features activities that simulate any type of keyboard or mouse input that a human would use. Also, there are activities that can set focus to a certain window, minimize or maximize it, or perform any other kind of action on it. These activities are essential in creating an automation that simulates human behaviour. As explained [here](#), there are several technologies that can be used for these activities, each with their own advantages in certain situations.

Double Click, **Click**, **Hover** are activities that simulate the clicking or hovering of UI elements. These activities are very useful in situations when human behavior must be mimicked. As input, these activities receive a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required.

ii.Type Into

Type Into sends keystrokes to a UI element. Special keys are supported and can be selected from the drop-down list. This is a basic text input activity that is widely used in automations and is also generated by the automatic recording wizards. As input, this activity receives a string or a string variable that contains the text to be written, and a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required.

iii. Type Secure text

Type Secure Text sends a secure string to a UI element. As input, this activity receives a SecureString variable that contains the text to be written, and a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required. This activity is useful for secure automations, as it can use passwords that are stored in SecureString variables. Usually, the SecureString variable is supplied by a **Get Secure Credential** activity.

Practical 8A

AIM : Demonstrate the following events in UiPath:

- i. Element triggering event**
- ii. Image triggering event**
- iii. System Triggering Event**

STEPS :

1. Open Studio and create a new **Process** named by default **Main**.
2. Drag a **Sequence** container in the **Workflow Designer**.
3. Create the following variable:

Variable Name	Variable Type
ContinueMonitor	Boolean

4. Drag a **Log Message** activity inside the **Sequence** container.
 - In the **Properties** panel, select the **Level** option from the **Message** drop-down list.
 - Add the expression "Start monitoring..." in the **Message** field.
5. Add an **Assign** activity under the **Log Message** activity.
 - In the **Properties** panel, add the variable ContinueMonitor in the **To** field.
 - Add the condition True in the **Value** field.
6. Place a **Monitor Events** activity below the **Assign** activity.
 - In the **Properties** panel, add the value ContinueMonitor in the **RepeatForever** field.
7. Drag a **Hotkey Trigger** activity inside the **Monitor Events** activity. This activity opens the **Calculator** app from **Windows**.
 - Select the checkboxes for the Alt and Shift options.
 - In the **Key** field, type the letter c.
 - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
8. Drag another **Hotkey Trigger** activity and place it next to the previous **Hotkey Trigger** activity. This activity opens a new browser tab and searches on Google the text previously selected by the user.
 - Select the checkboxes for the **Alt** and **Shift** options.
 - In the **Key** field, type the letter g.
 - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
9. Drag another **Hotkey Trigger** activity and place it next to the previous **Hotkey Trigger** activity. This activity stops monitoring the events.
 - Select the check boxes for the **Alt** and **Shift** options.
 - In the **Key** field, type the letter s.
 - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
10. Add a new **Sequence** container and place it below the **Hotkey Trigger** activity.

- In the **Properties** panel, add the name Event Handler in the **DisplayName** field.
- Create the following variable:

Variable Name	Variable Type
TriggerHotkey	UiPath.Core.EventInfo
ContinueMonitor	Boolean

11. Drag a **Log Message** activity inside the **Event Handler**.

- In the **Properties** panel, select the **Info** option from the **Level** drop-down list.
- Add the expression "Event triggered" in the **Message** field.

12. Drag a **Get Event Info** activity below the **Log Message** activity.

- In the **Properties** panel, add the variable TriggerHotkey in the **Result** field.
- Select the **UiPath.Core.EventInfo** option from the **TypeArgument** drop-down list.

13. Place a **Switch** activity below the **Get Event Info** activity. All **Hotkey Triggers** are described inside this activity and treated as cases.

- In the **Properties** panel, add the value TriggerHotkey.KeyEventInfo.KeyName.ToLower in the **Expression** field.
- Select the **String** option from the **TypeArgument** drop-down list.

14. Click the **Add new case** button from the **Switch** activity.

- Add the value c in the **Case value** field.

15. Place an **Open Application** activity and place it inside the **Case c** container. This represents the first **Hotkey Trigger** case that opens the **Calculator** app.

- In the **Properties** panel, add the expression "calc.exe" in the **Arguments** field.
- Add the expression "<wnd app='applicationframehost.exe' title='Calculator' />" in the **Selector** field.

16. Click the **Add new case** button from the **Switch** activity.

- Add the value g in the **Case value** field.

17. Drag a **Sequence** container and place it inside the **Case g** container. This represents the second **Hotkey Trigger** case that initiates a Google search for the previously selected text.

- In the **Properties** panel, add the name Google selected text in the **DisplayName** field.
- Create the following variable:

Variable Name	Variable Type
TextToSearch	GenericValue

18. Drag a **Delay** activity and place it inside the **Google selected text** sequence.

- In the **Properties** panel, add the value 00:00:00.5000000 in the **Duration** field.

19. Add a **Copy Selected Text** activity below the **Delay** activity.

- In the **Properties** panel, add the value True in the **ContinueOnError** field.
- Add the variable TextToSearch in the **Result** field.
- Add the value 2000 in the **Timeout (milliseconds)** field.

20. Drag an **If** activity under the **Copy Selected Text** activity.

- In the **Properties** panel, add the expression TextToSearch IsNot Nothing in the **Condition** field.

21. Place an **Open Browser** activity inside the **Then** box.

- In the **Properties** panel, select the **IE** option from the **BrowserType** drop-down list.
- Add the expression "www.google.com" in the **Url** field.
- Select the checkbox for the **NewSession** option. This starts a new session in the selected browser.

22. Place a **Type Into** activity inside the **Do** sequence.

- In the **Properties** panel, select the **Target** option from the **Target** drop-down list.
- Add the expression "<webctrl tag='INPUT' aaname='Search' />" in the **Selector** field.
- Select the **INTERACTIVE** option from the **WaitForReady** drop-down list.
- Add the variable TextToSearch in the **Text** field.
- Select the checkbox for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the checkbox for the **SimulateType** option. This option simulates the type using the technology of the target application.

23. Drag a **Send Hotkey** application below the **Type Into** activity.

- In the **Properties** panel, add the expression "enter" in the **Key** field.
- Select the **Target** option from the **Target** drop-down list.
- Add the expression "<webctrl tag='INPUT' aaname='Search' />" in the **Selector** field.
- Select the **INTERACTIVE** option from the **WaitForReady** drop-down list.
- Select the checkbox for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the **None** option from the **KeyModifiers** drop-down list.
- Select the checkbox for the **SpecialKey** option. This option indicates if the use of a special key in the keyboard shortcut.

24. Drag a **Message Box** activity in the **Else** container.

- In the **Properties** panel, select the **Ok** button from the **Buttons** drop-down list.
- Add the expression "Text could not be copied. Please try again." in the **Text** field.
- Select the checkbox for the **TopMost** option. This option always brings the message box to the foreground.

25. Click the **Add new case** button from the **Switch** activity.

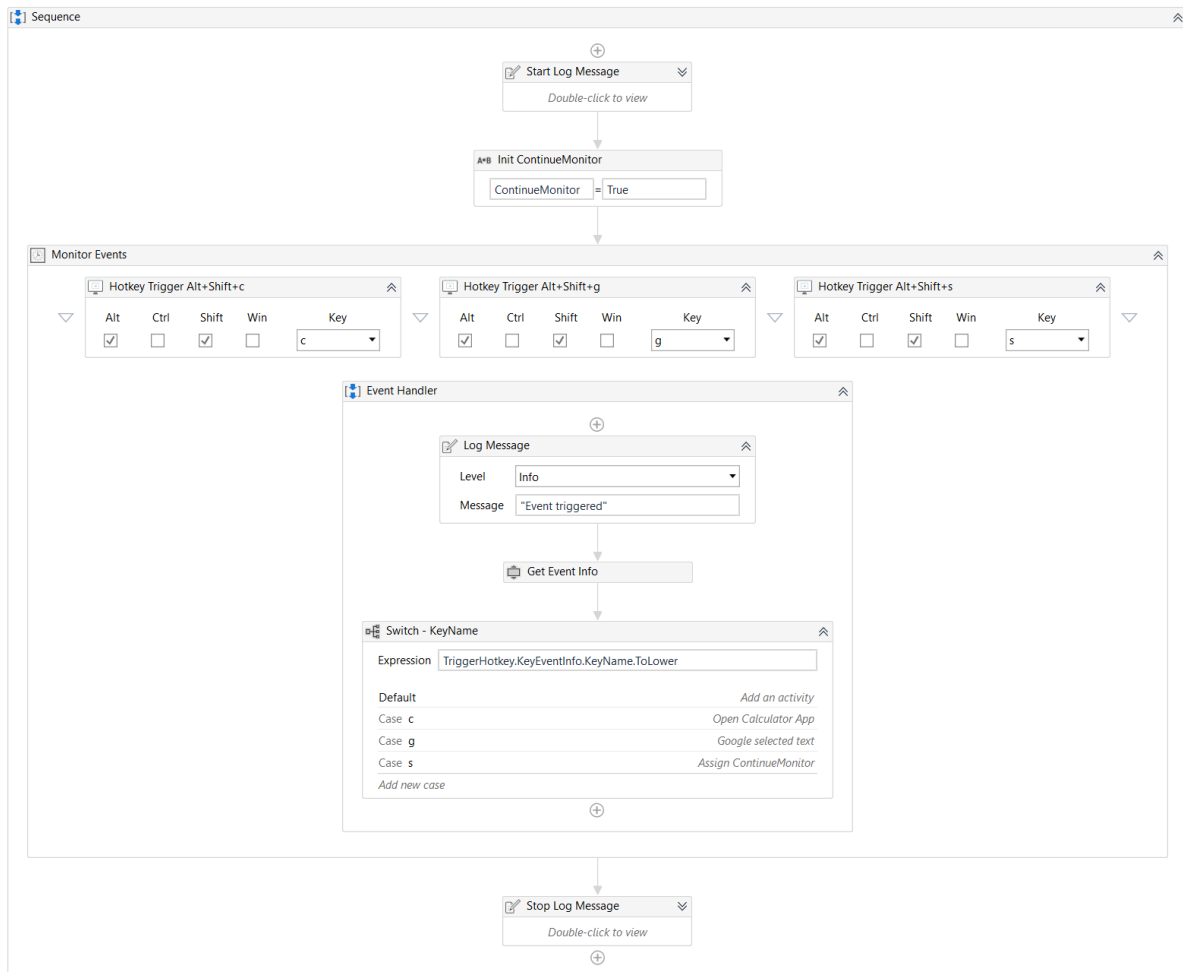
- Add the value s in the **Case value** field.

26. Drag an **Assign** activity inside the **Case s** container. This represents the third **Hotkey Trigger** case that stops monitoring the events.

- Add the variable ContinueMonitor in the **To** field.
- Add the condition False in the **Value** field.

27. Place a **Log Message** activity below the **Monitor Events** activity.

- In the **Properties** panel, select the **Info** option from the **Level** drop-down field.
- Add the expression "Stop monitoring.." in the **Message** field.



Practical 8B

AIM : Automate the following screen scraping methods using UiPath

- i. Full Test
- ii. Native
- iii. OCR

STEPS :

Output or screen scraping methods refer to those activities that enable you to extract data from a specified UI element or document, such as a .pdf file.

To understand which one is better for automating your business process, let's see the differences between them.

Capability Method	Speed	Accuracy	Background Execution	Extract Text Position
FullText	10/10	100%	Yes	no
Native	8/10	100%	No	yes
OCR	3/10	98%	No	yes

FullText is the default method, it is fast and accurate, yet unlike the **Native** method, it cannot extract the screen coordinates of the text.

Both these methods work only with desktop applications, but the **Native** method only works with apps that are built to render text with the Graphics Device Interface (GDI).

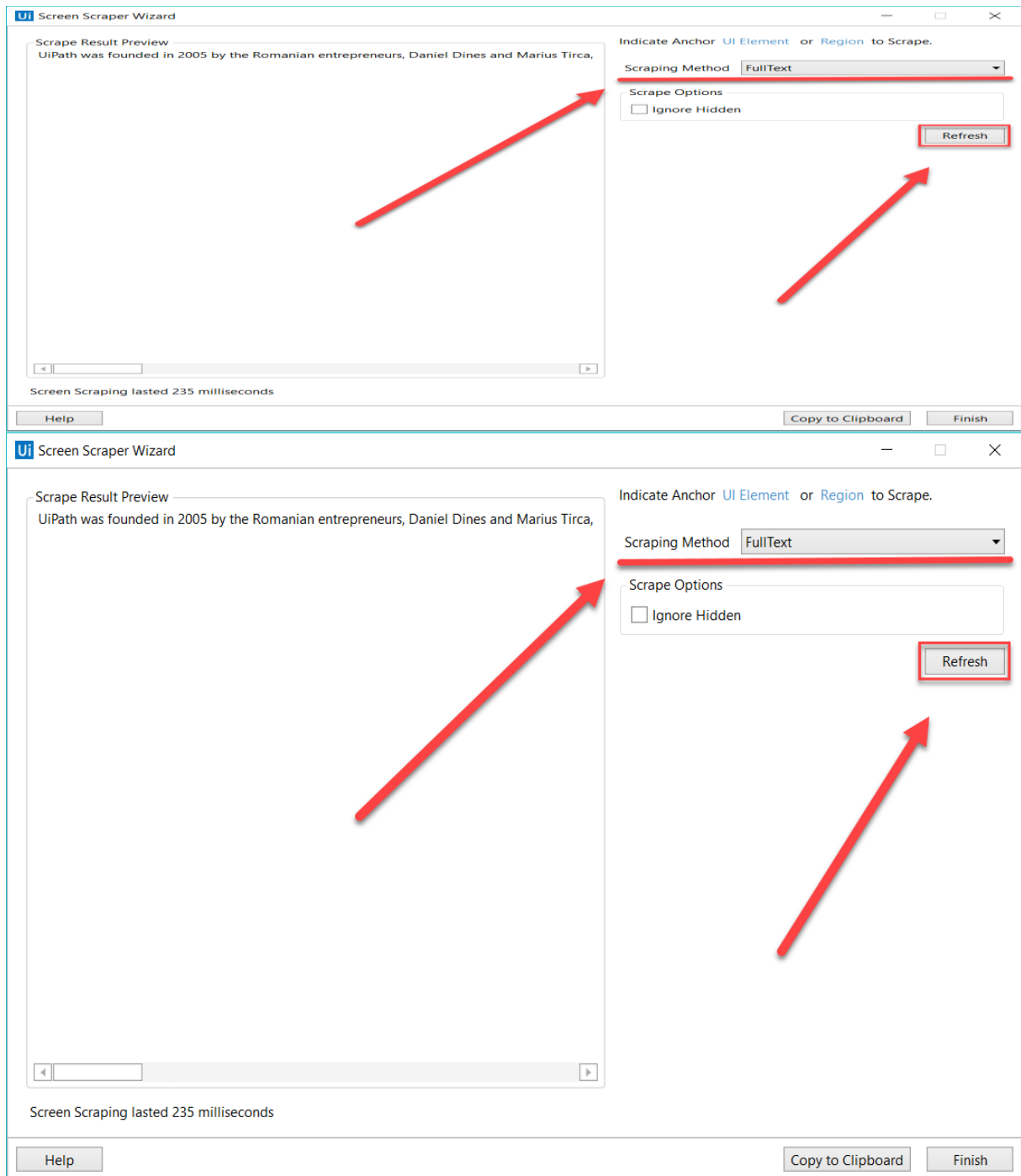
OCR is not 100% accurate but can be useful to extract text that the other two methods could not, as it works with all applications including Citrix. Studio uses two OCR engines, by default: Google Tesseract and Microsoft Modi.

Languages can be changed for OCR engines and you can find out how to [Install OCR Languages](#) here.

Capability Method	Multiple Languages Support	Preferred Area Size	Support for Color Inversion	Set Expected Text Format
Google Tesseract	Can be added	Small	yes	yes
Microsoft MODI	Supported by default	Large	no	no

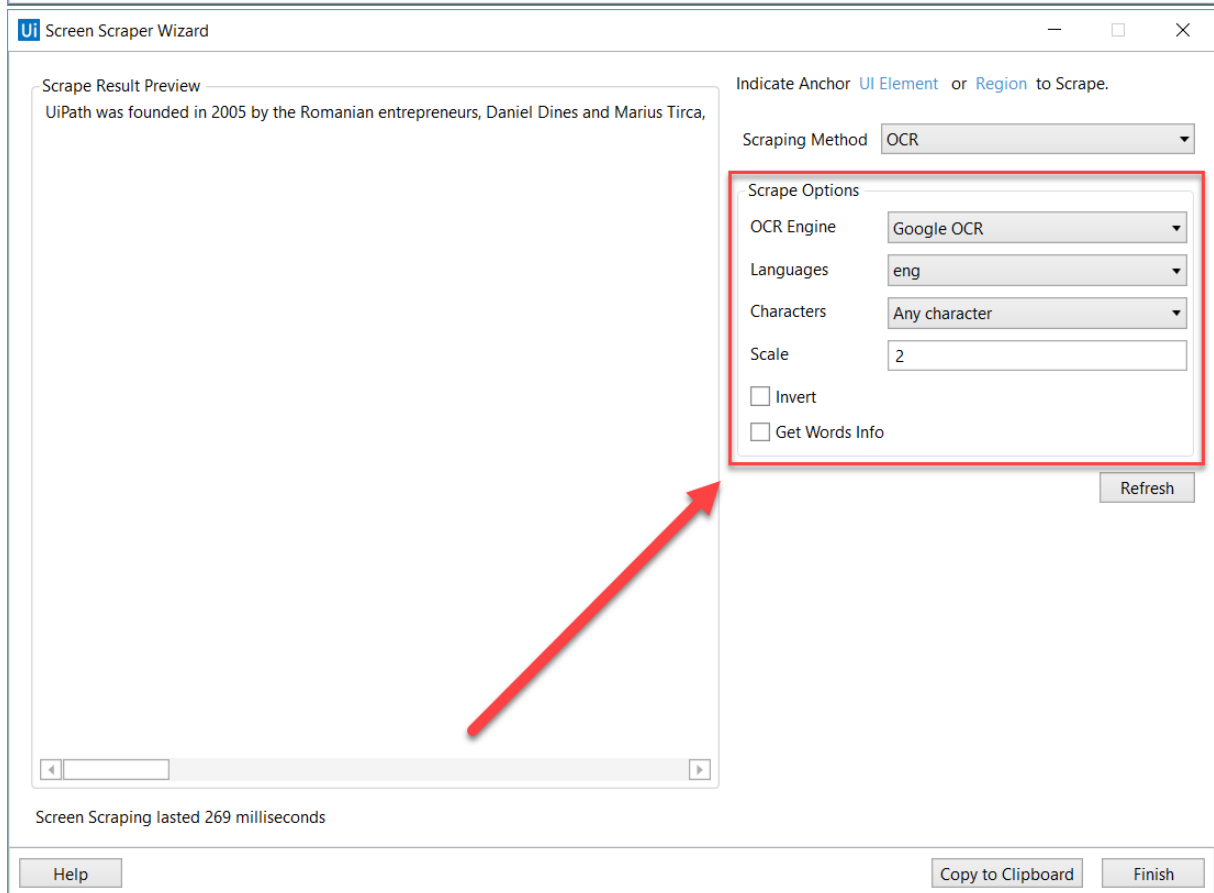
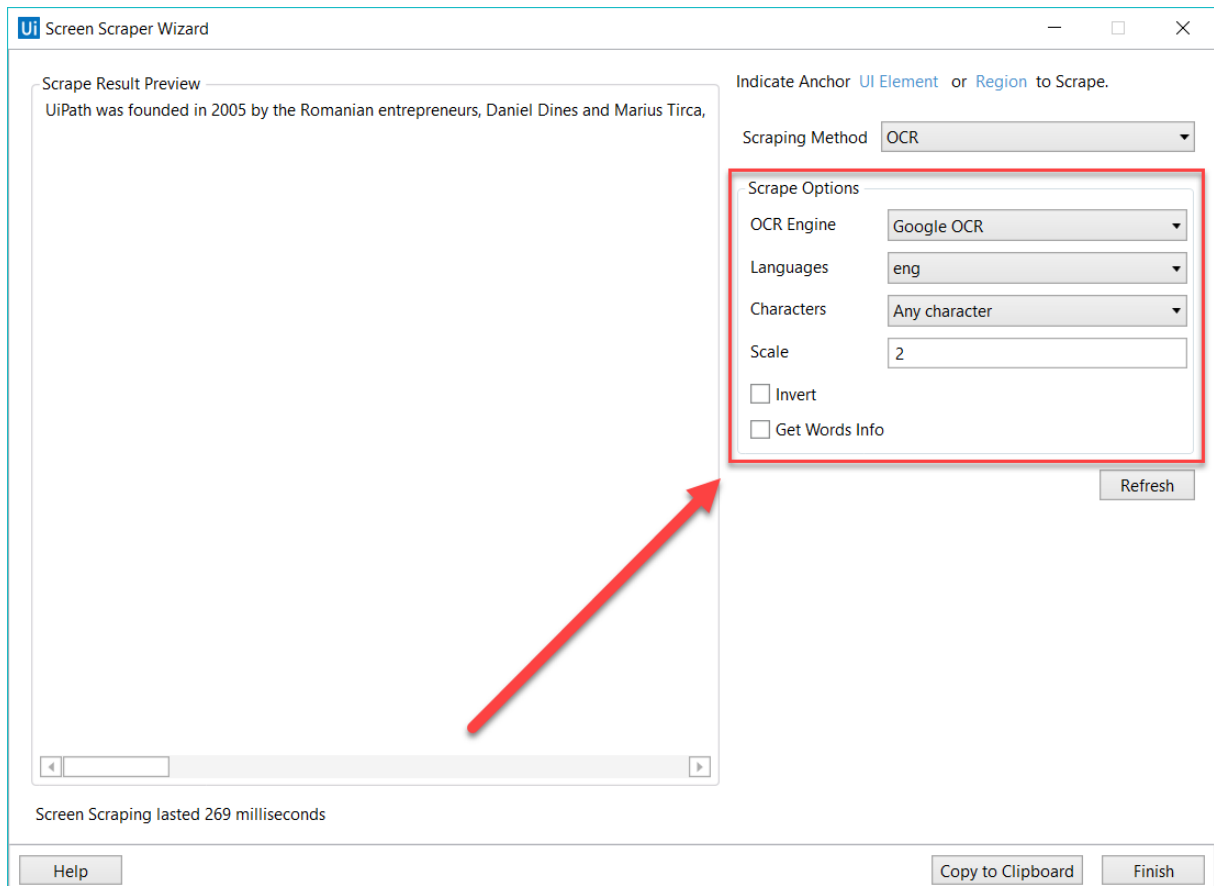
To start extracting text from various sources, click the **Screen Scraping** button, in the **Wizards** group, on the **Design** ribbon tab.

The screen scraping wizard enables you to point at a UI element and extract text from it, using one of the three output methods described above. Studio automatically chooses a screen scraping method for you, and displays it at the top of the **Screen Scraper Wizard** window.



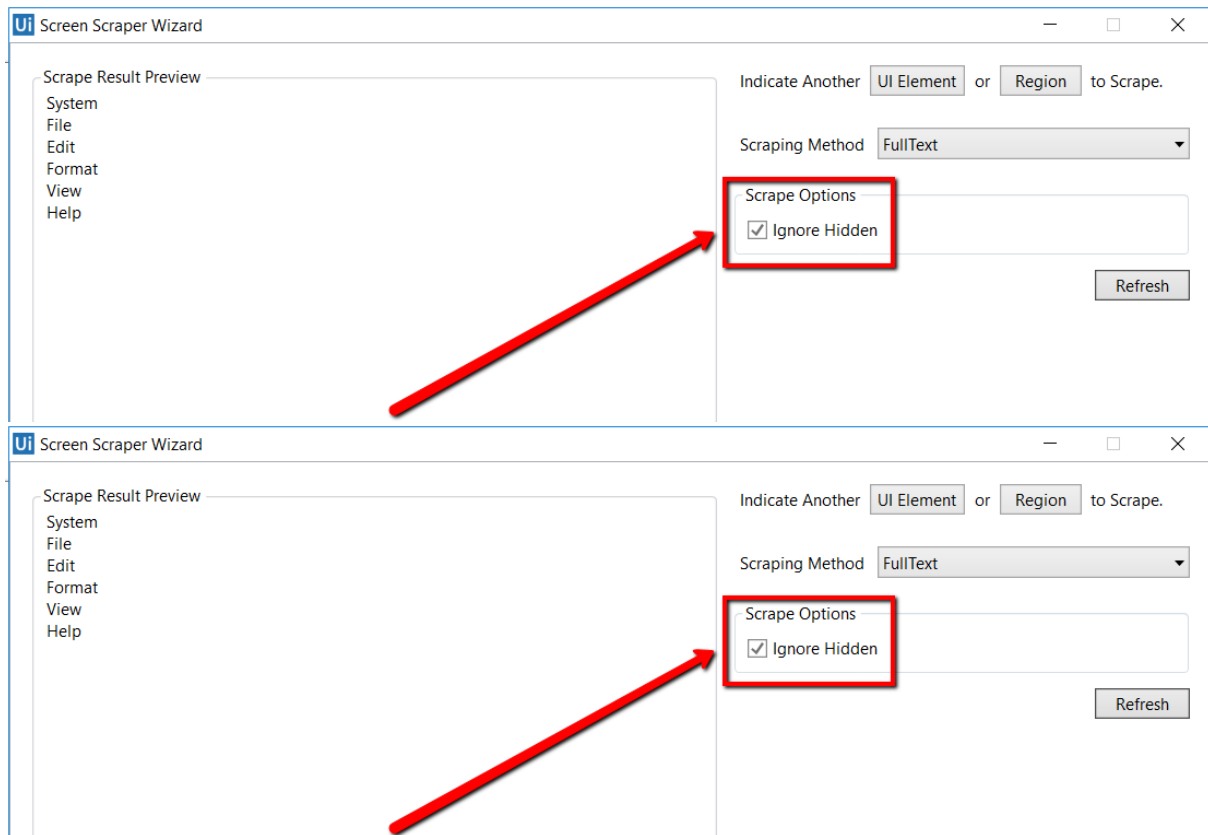
To change the method of screen scraping, select another one from the **Options** panel and then click **Refresh**.

When you are satisfied with the scraping results, click **Copy to Clipboard** and then **Finish**. The latter option copies the extracted text to the Clipboard, and it can be added to a **Generate Data Table** activity in the **Designer** panel. Just like desktop recording, screen scraping generates a container (with the selector of the top-level window) which contains activities, and partial selectors for each activity.



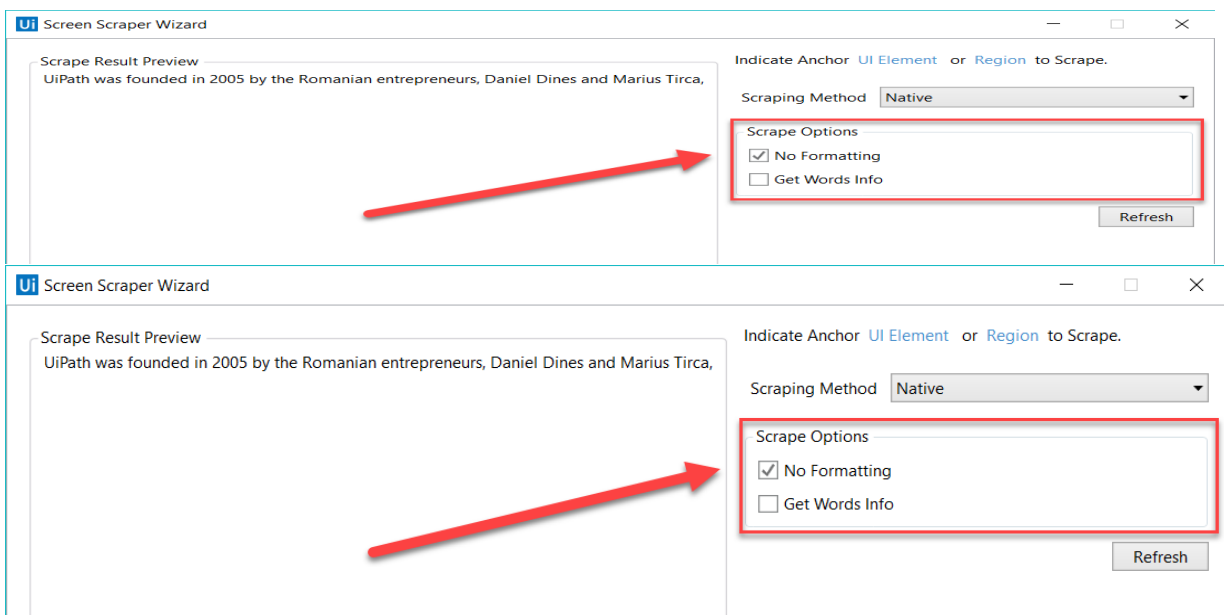
Each type of screen scraping comes with different features in the **Screen Scraper Wizard**, in the **Options** panel:

1. FullText



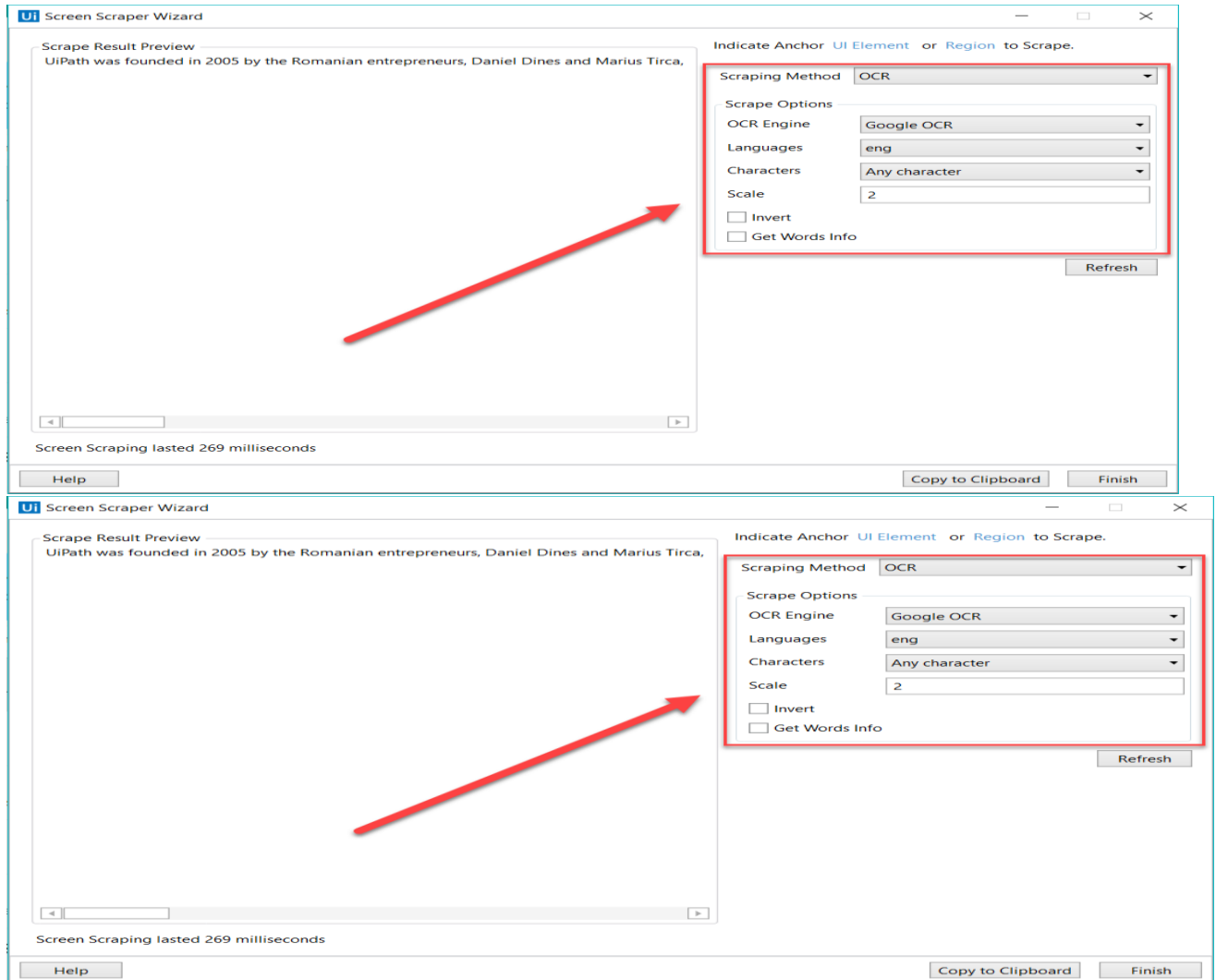
- **Ignore Hidden** – when this checkbox is selected, the hidden text from the selected UI element is not copied.

2. Native



- **No Formatting** – when this checkbox is selected, the copied text does not extract formatting information from the text. Otherwise, the extracted text's relative position is retained.
- **Get Words Info** – when this checkbox is selected, Studio also extracts the screen coordinates of each word. Additionally, the Custom Separators field is displayed, which enables you to specify the characters used as separators. If the field is empty, all known text separators are used.

3. Google OCR



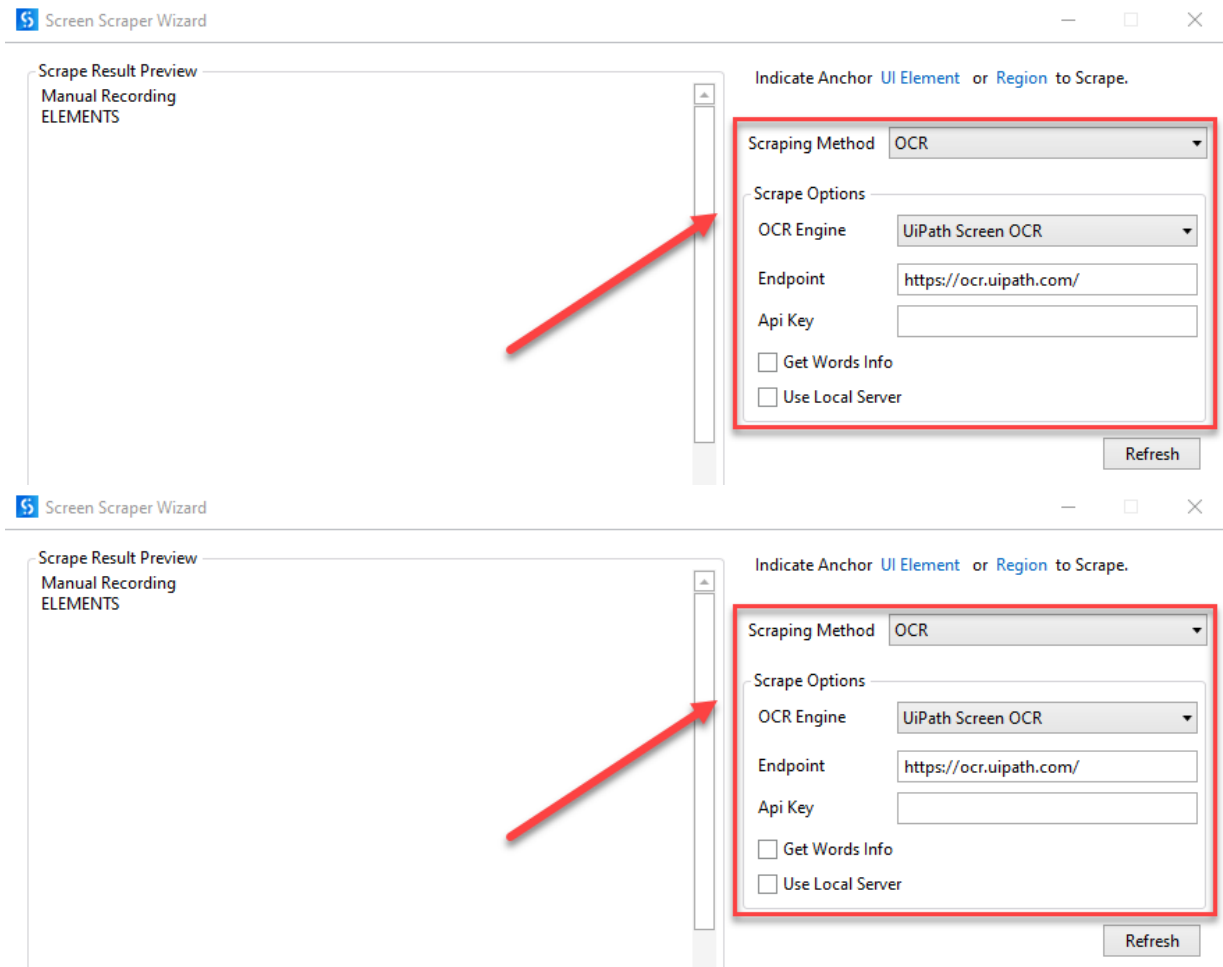
- **Languages** – only English is available by default.
- **Characters** – enables you to select which types of characters to be extracted. The following options are available: **Any character**, **Numbers only**, **Letters**, **Uppercase**, **Lowercase**, **Phone numbers**, **Currency**, **Date** and **Custom**. If you select **Custom**, two additional fields, **Allowed** and **Denied**, are displayed that enable you to create custom rules on which types of characters to scrape and which to avoid.
- **Invert** – when this checkbox is selected, the colors of the UI element are inverted before scraping. This is useful when the background is darker than the text color.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** – gets the on-screen position of each scraped word.



Note:

In some instances of UiPath Studio, the Google Tesseract engine may have training files (about training files: [Wikipedia](#), [GitHub](#)) that do not work for certain non-English languages. Running a project with these corrupted training files may lead to an exception being thrown. To fix this issue, download the training file for the language you wish to use from [here](#) and copy it into the tessdata folder from the UiPath installation directory. To check if the training files you downloaded work, you can download this [test project](#).

4. UiPath Screen OCR



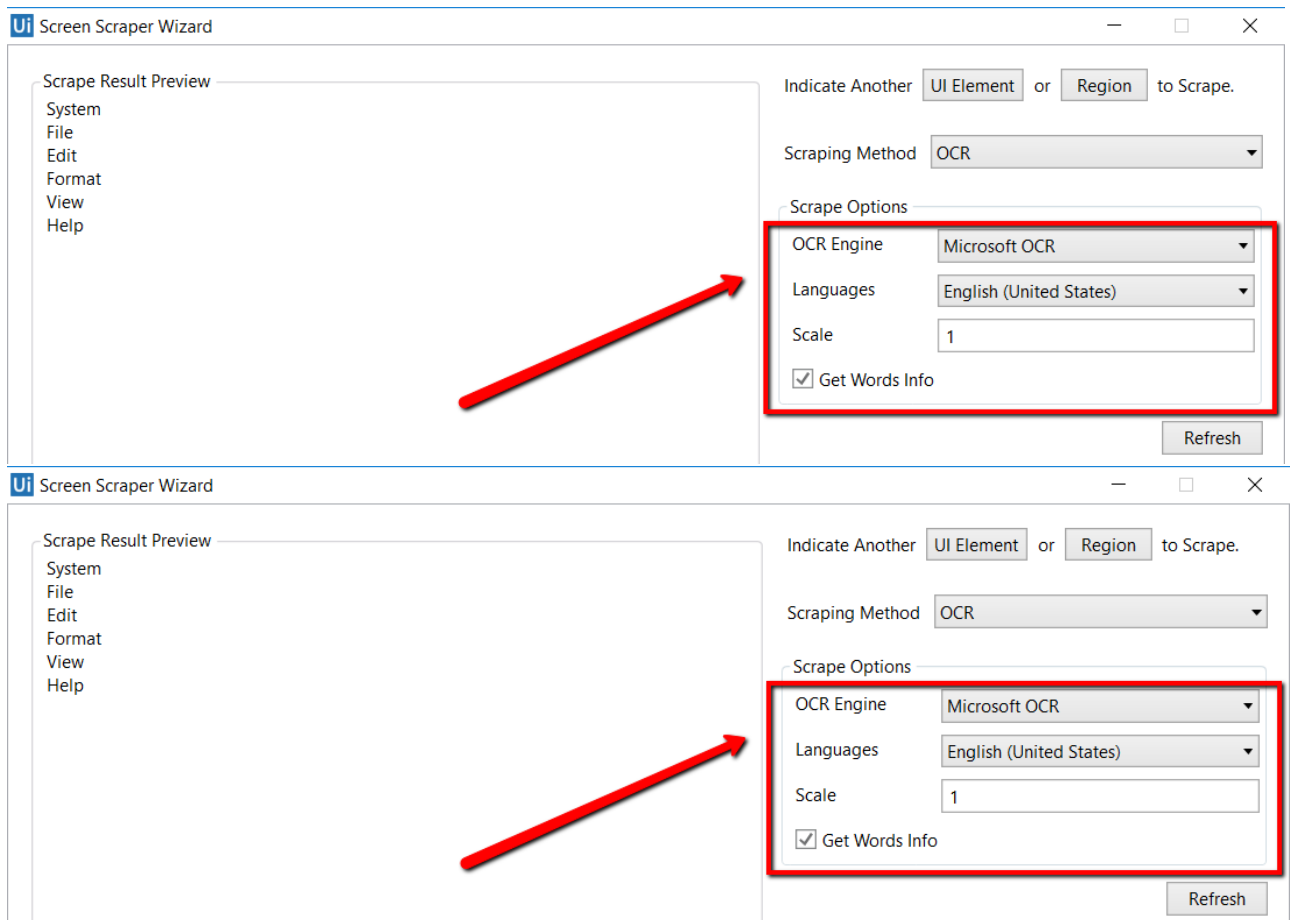
- **Endpoint** – the endpoint where the OCR model is hosted, either publicly or through an [ML Skill in AI Center](#).
- **API Key** – the endpoint API key.
- **Get Words Info** – gets the on-screen position of each scraped word.
- **Use Local Server** – select this option if you want to run the OCR locally (requires [Computer Vision Local Server Pack](#))

5. Microsoft OCR



Important

Microsoft OCR scraping engine does not support .NET 5 workflows.



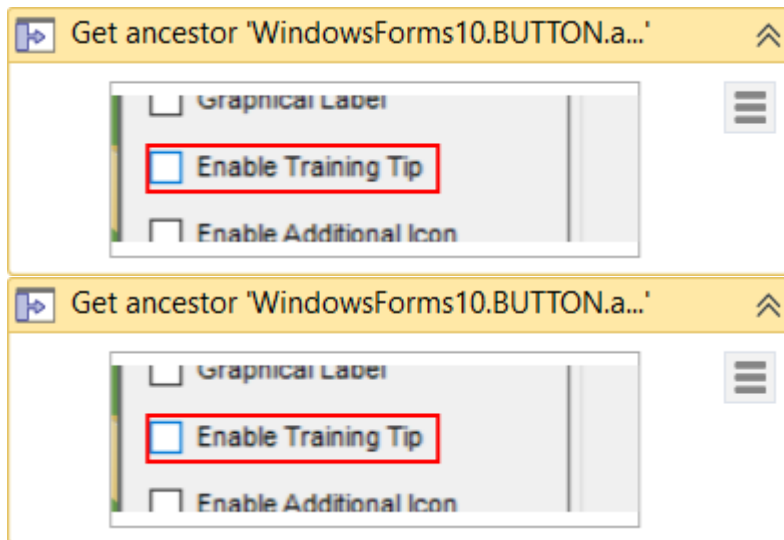
- **Languages** – enables you to change the language of the scraped text. By default, English is selected.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** - gets the on-screen position of each scraped word.

Besides getting text out of an indicated UI element, you can also extract the value of multiple types of attributes, its exact screen position, and its ancestor.

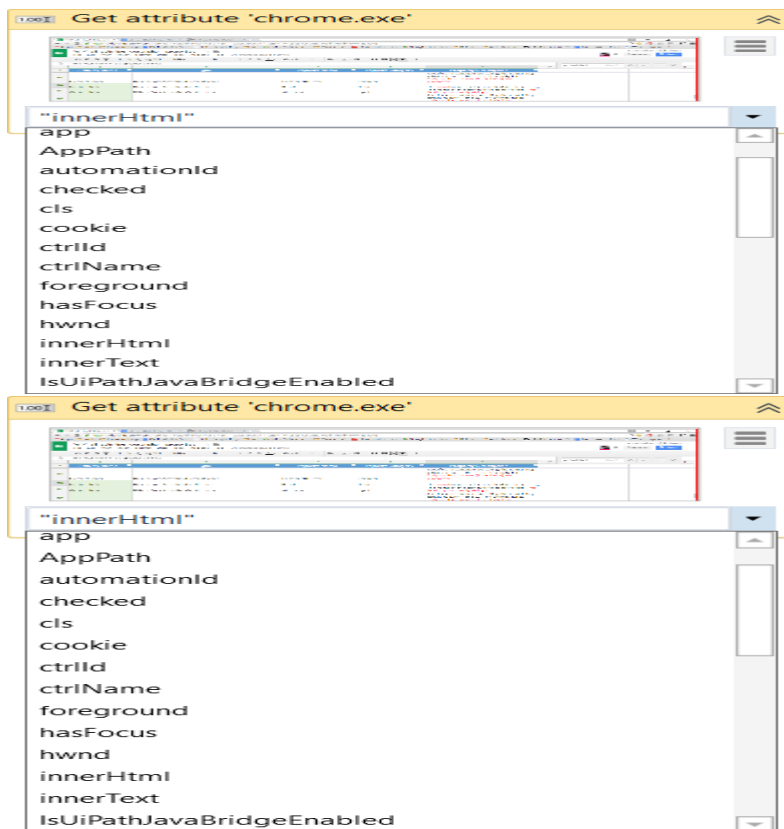
This type of information can be extracted through dedicated activities that are found in the **Activities** panel, under **UI Automation > Element > Find** and **UI Automation > Element > Attribute**.

These activities are:

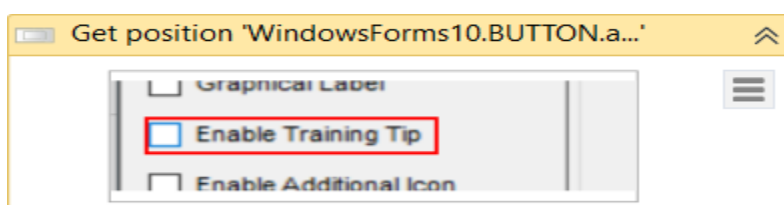
- **Get Ancestor** – enables you to retrieve an ancestor from a specified UI element. You can indicate at which level of the UI hierarchy to find the ancestor, and store the results in a UiElement variable.



- **Get Attribute** – retrieves the value of a specified UI element attribute. Once you indicate the UI element on screen, a drop-down list with all available attributes is displayed.



- **Get Position** – retrieves the bounding rectangle of the specified UiElement, and supports only Rectangle variables.



Practical 8C

AIM : Install and automate any process using UiPath with the following plug-ins:

- i. Java Plugin**
- ii. Mail Plugin**
- iii. PDF Plugin**
- iv. Web Integration**
- v. Excel Plugin**
- vi. Word Plugin**
- vii. Credential Management**

STEPS :

UiPath Run Job

The UiPath Run Job post-build step starts an already deployed process on an Orchestrator instance. The processes this task refers to are found in Automations->Processes on newer versions of Orchestrator and directly on the Processes tab on older versions of Orchestrator.

UiPath Run Job

UiPath

Process: ?

Input Parameters: ?

Priority: Low

Strategy: ☐ Allocate dynamically ☐ Specific robots

JobType: ☐ Unattended ☐ NonProduction ?

Orchestrator address: ?

Orchestrator tenant: ?

Orchestrator folder: ?

Authentication: ☐ Authenticate to an On-Premise Orchestrator using a username and a password ☐ Authenticate to a Cloud Orchestrator using a refresh token ☐ Authenticate to a Cloud Orchestrator using an external application

Job results output path: ?

Timeout (seconds): ?

Fail when job fails: ☒ ?

Wait for job completion: ☒ ?

Trace logging level: None

Errors: Process name is mandatory, Orchestrator address is mandatory, The folder is mandatory

⚙️ Configuration

Argument	Description
Process	(Required) Process name. You can take the process name from the Orchestrator UI. If the process is deployed in a Modern Folder then this argument should be the NAME of the process in the Processes tab. If the process is deployed in a Classic Folder, then the argument must be formed by the NAME of the process and the ENVIRONMENT (eg: NAME: ProcessA ENVIRONMENT: Production ProcessName: ProcessA_Production).
Parameters	The full path to a json input file. This is used when the Process requires input.
Priority	The job run priority.
Strategy	Specify the job run strategy, dynamically allocated job(s) or robot specific job(s). Options: Allocate dynamically, Specific robots
Job Type	This feature is available only on modern folders! Choose the license model of the runtime under which the job is to be executed.
Orchestrator address	The address of the Orchestrator instance where we'll run the process.
Orchestrator tenant	Specify the Orchestrator tenant.
Orchestrator folder	Specify the folder where the specified process was deployed.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: (1) Authenticate to an On-Premise Orchestrator using username and password (2) Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) (3) Authenticate to a Cloud Orchestrator using external app authentication.
Job results output path	Specify the output full path of the job results, e.g. testResults.json. The results are outputted in json format. If not specified, the results are outputted to the artifact staging directory as UiPathResults.json. The output is in json format.
Timeout	Specify the job run(s) timeout in seconds.
Fail when job fails	The task fails when at least one job fails. (default true)
Wait for job completion	Wait for job run(s) completion. (default true)
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.
	Parameters used for strategy Allocate dynamically

Argument	Description
No. of jobs	The number of job runs. (default 1)
User	The name of the user. This feature is available only on modern folders! This should be a machine user, not an orchestrator user. For local users, the format should be MachineName\UserName
Machine	The name of the machine. This feature is available only on modern folders!
	Parameters used for strategy Specific robots
Robot names	Comma-separated list of specific robot names.

❏ Pipeline example:

```

pipeline {
  agent any
  environment {
    MAJOR = '1'
    MINOR = '0'
  }
  stages {
    stage ('Build') {
      UiPathRunJob(
        credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
        failWhenJobFails: true,
        folderName: 'A_Classic',
        orchestratorAddress: 'https://testorchestrator.some-domain.com',
        orchestratorTenant: 'Default',
        parametersFilePath: '',
        priority: 'Low',
        processName: 'ProcessA_EnvB',
        resultFilePath: 'output.json',
        strategy: Dynamically(jobsCount: 1, machine: 'TestMachine', user: 'TestUser'), timeout: 3600,
        waitForJobCompletion: true, traceLoggingLevel: 'None'
      )
      UiPathRunJob(
        credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
        failWhenJobFails: true,
        folderName: 'A_Classic',
        orchestratorAddress: 'https://testorchestrator.some-domain.com',
        orchestratorTenant: 'Default',
        parametersFilePath: '',
        priority: 'Low',
        processName: 'ProcessA_EnvB',
        resultFilePath: 'output.json',
        strategy: Robot('robot1,robot2'),
        timeout: 1800,
        waitForJobCompletion: false,
        traceLoggingLevel: 'None'
      )
    }
  }
}

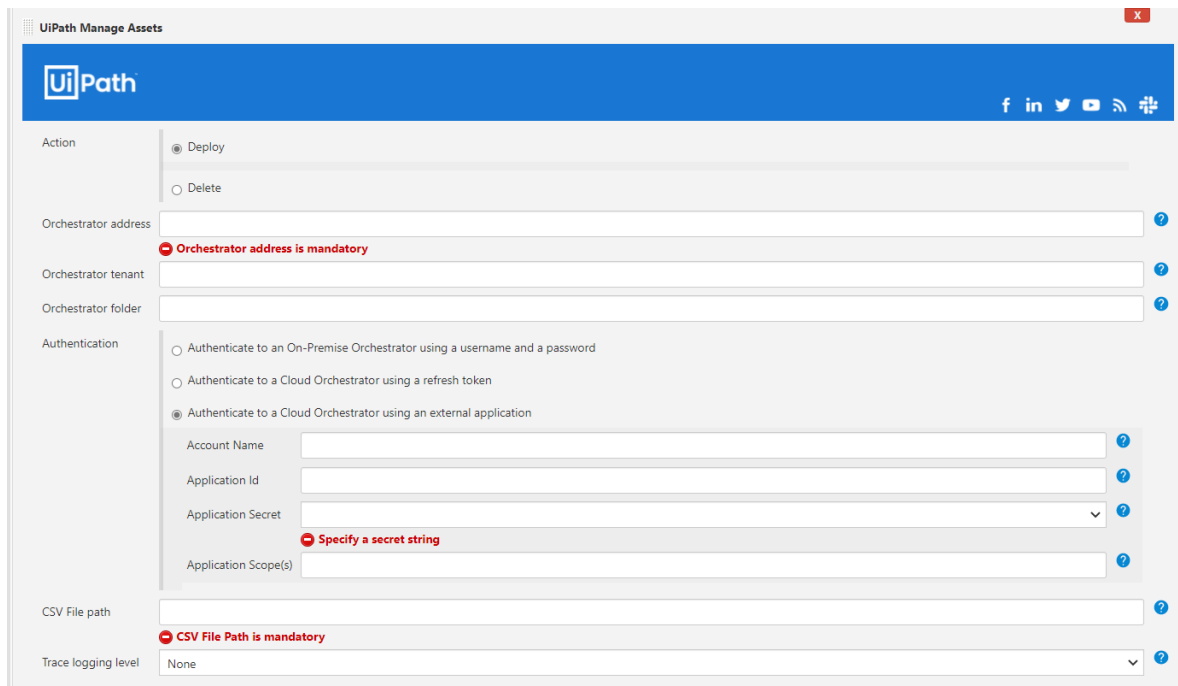
```

UiPath Manage Assets

The UiPathManageAssets step enables you to deploy, update or delete assets on an Orchestrator instance. In order to deploy assets you must describe them in a CSV file like the one in the example below encoded in uft-8.

```
name,type,value,description
asset_1_name,text,asset_value,this is an test description # we can have comments
asset_2_name,integer,123
asset_3_name,boolean,false
asset_4_name,credential,"username::password"
```

There are 4 types of assets text, integer, boolean and credential. For the credential you must encode the username and password by using :: to separate the two fields.



Configuration

Argument	Description
Action	What to do with the provided assets: deploy or delete. If a deployed asset exists then it will be updated instead.
Orchestrator address	The address of the Orchestrator instance where we'll deploy or update assets.
Orchestrator tenant	Specify the Orchestrator tenant onto which the assets will be deployed or updated.
Orchestrator folder	Specify the folder where assets will be deployed or updated.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: (1) Authenticate to an On-Premise

Argument	Description
	Orchestrator using username and password (2) Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) (3) Authenticate to a Cloud Orchestrator using external app authentication.
CSV File Path	The path to the csv file containing the asset details. The same file can be used to deploy or update the assets although the type isn't required for update. The type field can also be empty but the column must be present. For delete, only the name column is used, so the other columns can be empty but they must be present. You can set an optional description for each asset (e.g. type, value, description). Make sure to remove any line breaks as each line is interpreted as a new asset.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

Pipeline example:

```

pipeline {
  agent any
  environment {
    MAJOR = '1'
    MINOR = '0'
  }
  stages {
    stage ('Build') {
      UiPathAssets (
        assetsAction: DeployAssets(),
        credentials: Token(accountName: "", credentialsId: ""),
        filePath: '${WORKSPACE}/test.csv',
        folderName: 'Default',
        orchestratorAddress: 'https://test-orchestrator.somedomain.com',
        orchestratorTenant: 'Default',
        traceLoggingLevel: 'None'
      )
      UiPathAssets(
        assetsAction: DeleteAssets(),
        credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
        filePath: '${WORKSPACE}/test.csv',
        folderName: 'Default',
        orchestratorAddress: 'https://test-orchestrator.somedomain.com',
        orchestratorTenant: 'Default',
        traceLoggingLevel: 'None'
      )
    }
  }
}

```

UiPath Pack

Application: RPA

Type: Build task

UiPath Pack is available in standard jobs and pipelines, and lets you package an existing UiPath project into a NuGet package.

UiPath Pack

Choose versioning method

- ☒ Auto generate the package version
- ☐ Use custom package versioning
- ☐ Use the version specified in the project file

Project(s) path: \$(WORKSPACE)\LibraryInLibrary

Output folder: \$(WORKSPACE)\Output

Output type: Pack a library project

Run workflow analysis: ☐

☒ Use Orchestrator feed when packaging libraries

Orchestrator address:

Orchestrator tenant:

Authentication

- ☐ Authenticate to an On-Premise Orchestrator using a username and a password
- ☐ Authenticate to a Cloud Orchestrator using a refresh token
- ☒ Authenticate to a Cloud Orchestrator using an external application

Account Name:

Application Id:

Application Secret: Specify a secret string

Application Scope(s):

Trace logging level: None

⚙️ Configuration

Job parameter	Description
Choose versioning method	UiPath packages are versioned. With UiPath pack you can choose between 3 different options: (1) Auto generate (2) Define custom version (3) Use the current version set in the project.
Project(s) path	The location of the project(s) to be packaged. It can be a direct path to a project.json file or a directory with one or multiple projects. In the latter case, each level one project is packaged individually.
Output folder	Path to a folder, where the created package should be placed.
Output type	The output type of the project(s). There are 5 options for the project(s) type: (1) Output type of the project (2) Pack a process project (3) Pack a library project (4) Pack a tests project (5) Pack an objects project.

Job parameter	Description
Run workflow analysis	Run the workflow analysis before packing, checking the project via predefined rules for violations. Fail the job in case of errors. By default the analysis is not run.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.
Use orchestrator	Use Orchestrator feed when packaging libraries. The Orchestrator must be 20.4 or higher. The library feed needs to allow API Key authentication in Tenant -> Setting -> Deployment.
Orchestrator address	The address of the Orchestrator instance from which library dependencies should be restored.
Orchestrator tenant	The Orchestrator tenant from which library dependencies should be restored.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: (1) Authenticate to an On-Premise Orchestrator using username and password (2) Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) (3) Authenticate to a Cloud Orchestrator using external app authentication.

Pipeline Example:

```

pipeline {
    agent any
    environment {
        MAJOR = '1'
        MINOR = '0'
    }
    stages {
        stage ('Build') {
            steps {
                UiPathPack (
                    outputPath: "Output\\${env.BUILD_NUMBER}",
                    projectJsonPath: "UiBank\\project.json",
                    version: [${class: 'ManualVersionEntry', version:
"${MAJOR}.${MINOR}.${env.BUILD_NUMBER}"]
                    useOrchestrator: true,
                    traceLoggingLevel: "None",
                    orchestratorAddress: "OrchestratorUrl",
                    orchestratorTenant: "tenant name",
                    credentials: [${class: 'UserPassAuthenticationEntry', credentialsId: "credentialsId"}]
                )
            }
        }
    }
}

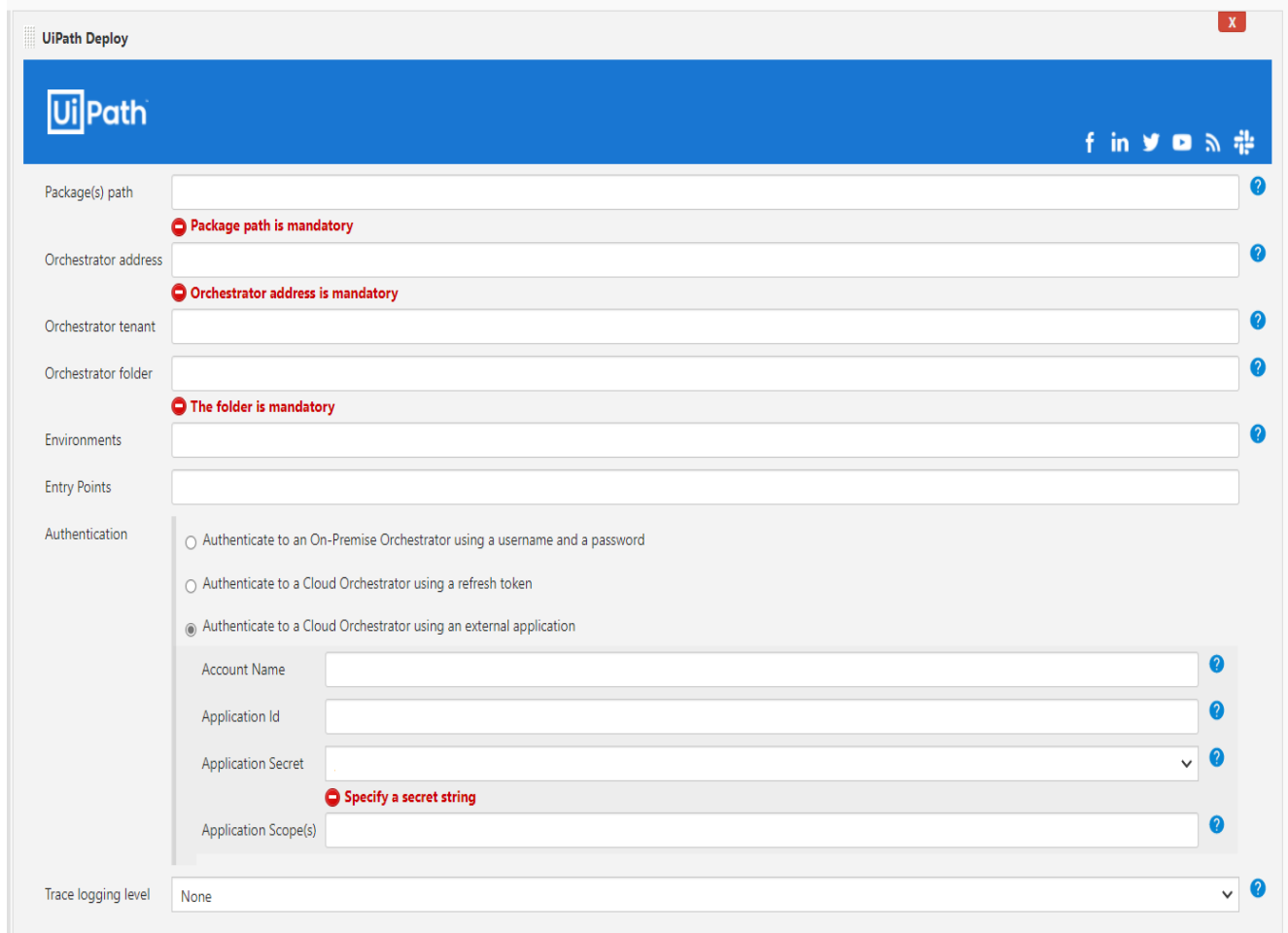
```

UiPath Deploy

Application: RPA

Type: Post-Build task

UiPath Deploy is available in standard jobs and pipelines, and lets you deploy a UiPath NuGet package onto UiPath Orchestrator.



UiPath Deploy

Package(s) path ?

Package path is mandatory

Orchestrator address ?

Orchestrator address is mandatory

Orchestrator tenant ?

Orchestrator folder ?

The folder is mandatory

Environments ?

Entry Points

Authentication

☐ Authenticate to an On-Premise Orchestrator using a username and a password

☐ Authenticate to a Cloud Orchestrator using a refresh token

☒ Authenticate to a Cloud Orchestrator using an external application

Account Name ?

Application Id ?

Application Secret ?

Specify a secret string

Application Scope(s) ?

Trace logging level ?

None

Configuration

Job parameter	Description
Package(s) path	The folder that holds your UiPath nuget package(s).
Orchestrator address	The address of the Orchestrator instance onto which the package(s) will be deployed.
Orchestrator tenant	The Orchestrator tenant onto which the package(s) will be deployed.
Orchestrator folder	The folder to deploy to. If the folder is a classic folder, you will also need to set the environments field. For modern folders, setting the environments is not required.

Job parameter	Description
Environments	The environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Specify the environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Required when using a classic folder, otherwise not applicable.
Entry Points	<p>Specify entry points to create or update a process. The entry point specifies the filePath starting from the root of the project.</p> <p>Conditions:</p> <ul style="list-style-type: none"> • Entry points are available for Orchestrator version 21.4 or higher (e.g. 21.4.UiPathDeploy.entryPoints). For Orchestrator versions lower than 21.4, you need to enter any value, as the field must not remain empty. • Default entry point set to Main.xaml. • For classic folders (deprecated) you can specify only one entry point for each environment • Multiple Entrypoints can be specified as ' Main.xaml, EntryPoint2.xaml ' <p>For more information, see Orchestrator Entry Points</p>
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: (1) Authenticate to an On-Premise Orchestrator using username and password (2) Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) (3) Authenticate to a Cloud Orchestrator using external app authentication.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

Note :

Make sure that your network allows access to the following NuGet package feed:

- <https://api.nuget.org/v3/index.json>
- https://uipath.pkgs.visualstudio.com/_packaging/nuget-packages/nuget/v3/index.json
- https://uipath.pkgs.visualstudio.com/Public.Feeds/_packaging/UiPath-Internal/nuget/v3/index.json
- <https://www.myget.org/F/workflow>
- <http://www.myget.org/F/uipath>
- <https://www.myget.org/F/uipath-dev/api/v3/index.json>

❏ Pipeline Example:

```
pipeline {
  agent any
  environment {
    MAJOR = '1'
    MINOR = '0'
  }
  stages {
    stage ('PostBuild') {
      steps {
        UiPathDeploy (
          packagePath: "path\\to\\NuGetpackage",
          orchestratorAddress: "OrchestratorUrl",
          orchestratorTenant: "tenant name",
          folderName: "folder name",
          environments: "environment",
          credentials: [$class: 'UserPassAuthenticationEntry', credentialsId: "credentialsId"],
          traceLoggingLevel: 'None'
        )
      }
    }
  }
}
```

► UiPath Run tests

Application: Testing

Type: Post-Build task

UiPath Run tests is available in standard jobs and pipelines, and lets you (1) run an existing Test Set on Orchestrator, or (2) package, deploy and run test cases, by specifying the path to a UiPath test project.

After the test run has finished, the Test Result tab will be published to the Jenkins build, showing the detailed results. Additionally, a JUnit test results file will be output to the test result output path, when specified, or to the workspace root if not specified.

Depending on the result, the build will be either marked as successful (all test cases are passed), or unstable (at least one test case failed).

UiPath Run tests

UiPath

[f](#) [in](#) [t](#) [y](#) [r](#) [s](#)

Target

☐ Execute test set

☐ Execute test project

Orchestrator address

Orchestrator address is mandatory

Orchestrator tenant

Orchestrator folder

The folder is mandatory

Authentication

☐ Authenticate to an On-Premise Orchestrator using a username and a password

☐ Authenticate to a Cloud Orchestrator using a refresh token

☐ Authenticate to a Cloud Orchestrator using an external application

Test results output path

Input Parameters

Timeout (seconds)

Trace logging level

None

Configuration

Job parameter	Description
Target	(1) Execute test set (specify an existing test set on UiPath Orchestrator) or (2) Execute test project (provide the project.json path of a UiPath Testing project)
Test result output path	The executed test set returns the test result as junit.xml. Specify the path where the result should be stored, relative to the Jenkins workspace directory (e.g. result.xml). <i>Optional</i>
Input Parameters	Define custom Arguments for your test cases to override default values at the test set level. Through the arguments, you can parameterize the test cases at runtime. To define arguments, you need to have published a package with arguments. <i>Optional</i>
Orchestrator address	The address of the Orchestrator instance onto which the package(s) will be deployed.
Orchestrator tenant	The Orchestrator tenant onto which the package(s) will be deployed.
Orchestrator folder	The folder to deploy to. If the folder is a classic folder, you will also need to set the environments field. For modern folders, setting the environments is not required.
Environments	The environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to

Job parameter	Description
	use the latest project version. Specify the environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Required when using a classic folder, otherwise not applicable.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: (1) Authenticate to an On-Premise Orchestrator using username and password (2) Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) (3) Authenticate to a Cloud Orchestrator using external app authentication.
Timeout (seconds)	The execution timeout for the test run. The default value is 7200 seconds. If the timeout exceeds before the execution on Orchestrator is finished and returned the final result, the built will cancel and be marked as failed.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

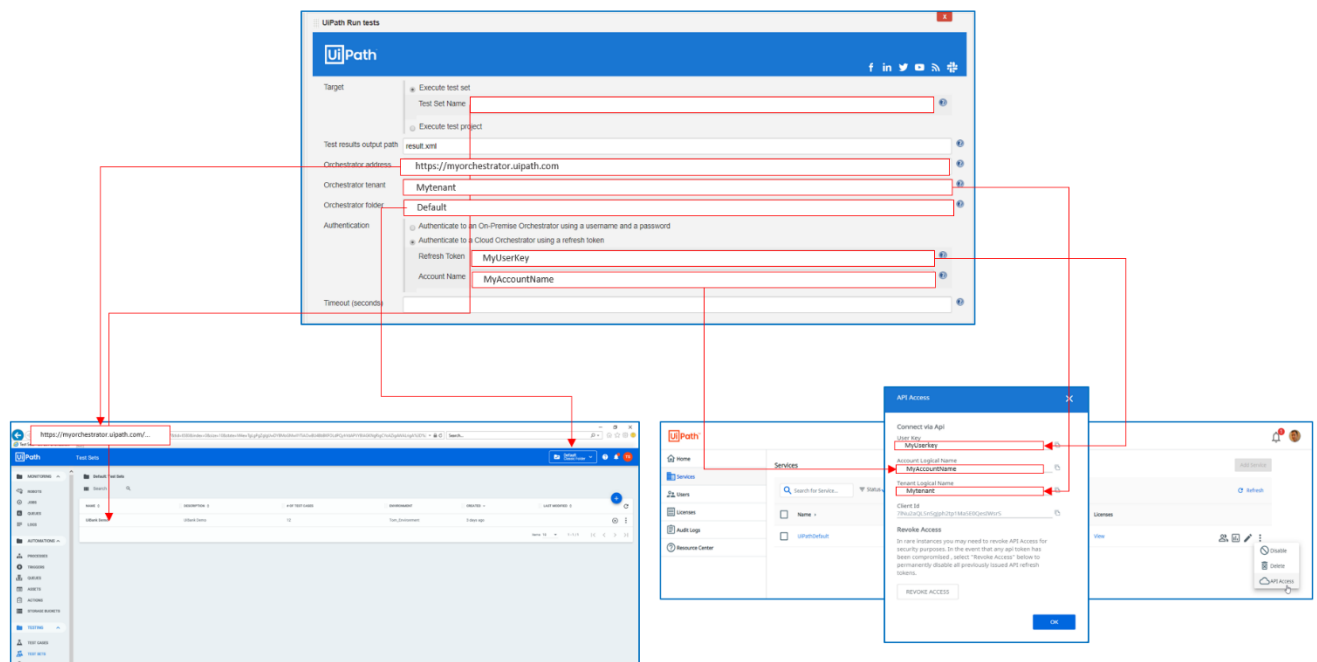
❏ Pipeline Example:

```

pipeline {
  agent any
  environment {
    MAJOR = '1'
    MINOR = '0'
  }
  stages {
    stage ('PostBuild') {
      steps {
        UiPathTest (
          testTarget: [$class: 'TestSetEntry', testSet: "My Test Set"],
          orchestratorAddress: "OrchestratorUrl",
          orchestratorTenant: "tenant name",
          folderName: "folder name",
          timeout: "10000",
          traceLoggingLevel: 'None',
          testResultsOutputPath: "result.xml",
          credentials: [$class: 'UserPassAuthenticationEntry', credentialsId: "credentialsId"]
        )
      }
    }
  }
}

```

Obtaining the Cloud Orchestrator API Key



Configure service connection for external apps

Step 1: Configure your external application and scopes in Automation Cloud. After adding the application, keep the App ID, Secret and Application Scopes at hand, to be used for the next step.

Note: If you generate and use a new Secret, the old one is going to be invalidated.

Step 2: Configure application credentials as secret text in Jenkins. For this step you need the Secret generated in Automation Cloud.

Step 3: Configure the Authentication for each task under Post-Build Actions, by adding the Account Name, followed by the App ID, Secret and Application Scopes generated through Automation Cloud.

Note: Consider using the external app in individual pipelines to avoid invalidation errors.

Additional information

All paths specified should be local to the current workspace. You can use environment variables in paths, though you should ensure that they result in paths that are local to the workspace. All paths

In order to deploy packages or run tests, ensure that the authenticated user has the Folders View (or OrganizationUnits View) and (20.4+ only) Background Tasks View permissions.

In order to package libraries when connected to an Orchestrator instance, ensure that the authenticated user has the Libraries View permission.

Authentication on Cloud Orchestrator using External Apps.

For further details on managing external apps on Orchestrator, pls refer to the [official documentation](#).

Questions

Do you have any questions regarding the plugin? Ask them [here](#).

Troubleshooting

Unauthorized error

Forbidden error

Folder/environment not found

Package already exists (Conflict)

Failed to run the command (Generic error))

Jenkins fails to process paths containing non-Latin characters

Unauthorized error

If using basic authentication:

- ensure the correctness of the username-password combination on the web login
- if federated authentication is enabled, make sure you write the username in the task as "DOMAIN\user"

If using token authentication:

- Revoke the token from the API access panel and generate a new one
- Ensure that the user that generated the key has can access the Orchestrator and has a user on the Orchestrator instance

If authenticating against on an on-premise Orchestrator you might receive this error as a result of the certificate used for the Orchestrator not being valid. This might mean that it has the wrong CN or other validation issues. Ensure that the Orchestrator certificate is valid and that the machine running the job trusts the Orchestrator certificate in case you are using a self-signed certificate.

Forbidden error

Likely, the user does not have the permission to perform the action.

Ensure that the user has permissions to read folders, upload packages, create and update processes, read test sets and test cases, create and run test sets, read background tasks.

Folder/environment not found

Ensure that the authenticated user used by CI/CD plugins has the Folders.View and (20.4 only) BackgroundTask.View permissions.

Package already exists (Conflict)

Ensure that the package that you are trying to deploy does not exist with the same version already. If it does, consider using automatic package versioning, so that the new version is bumped up every time we deploy.

Failed to run the command (Generic error)

If the Jenkins workspace is inside a location on disk (like C:\Windows or C:\Program Files) to which the user does not have permissions, ensure that the workspace is placed on a path that can be accessed smoothly by the user

Jenkins fails to process paths containing non-Latin characters

Jenkins is not able to pass correctly non-standard encoded characters when invoking the UiPath Plugin. The unknown characters will be replaced by ???.

The solution depends on how Jenkins is deployed on both the server and the agent host machines, but involves setting "file.encoding" to UTF-8 in Java options:

- Windows
 - Running Jenkins in Windows as a Service In the service configuration file add the arguments into the tag. It should look like this:

```
<arguments>-Xrs -Xmx512m -Dhudson.lifecycle=udson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 -jar "%BASE%\jenkins.war" --httpPort=8080 --webroot="%BASE%\war"</arguments>
```
 - Running Jenkins inside Docker The JAVA_OPTS should be passed to the container via --env JAVA_OPTS="..." like the following:

```
docker run --name myjenkins -p 8080:8080 -p 50000:50000 --env JAVA_OPTS=-Dhudson.lifecycle=udson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 jenkins/jenkins:its
```
 - Running Jenkins inside Tomcat Use environment variable CATALINA_OPTS:

```
export CATALINA_OPTS="-DJENKINS_HOME=/path/to/jenkins_home/ -Dhudson.lifecycle=udson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 -Xmx512m"
```
- Linux
 - Debian / Ubuntu based Linux distributions In the configuration file search for the argument JAVA_ARGS and add the file encoding. It might look like this:

```
JAVA_ARGS="-Dfile.encoding=UTF-8 -Xmx512m"
```
 - RedHat Linux based distributions In the configuration file search for the argument JENKINS_JAVA_OPTIONS and add the file encoding. It might look like this:

```
JENKINS_JAVA_OPTIONS="-Dfile.encoding=UTF-8 -Xmx512m"
```

License

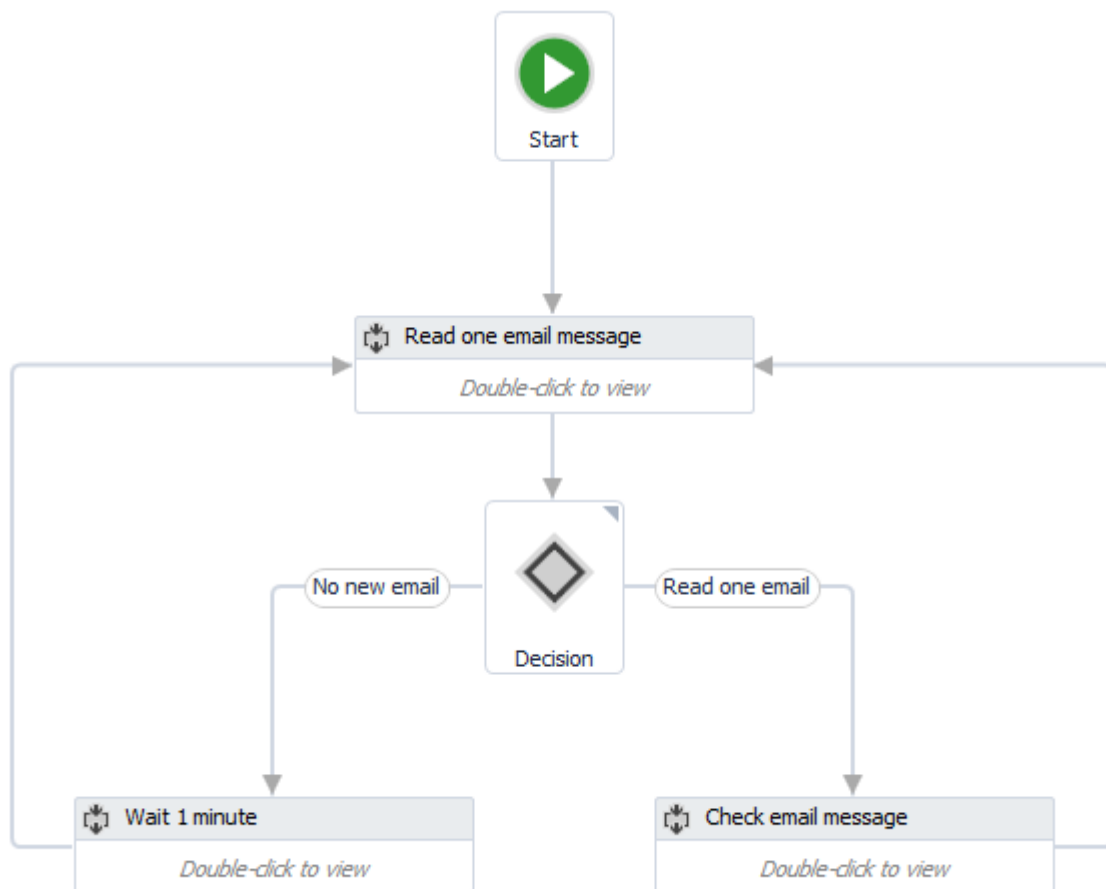
UiPath Open Platform License Agreement – V.20190913

Practical 9A

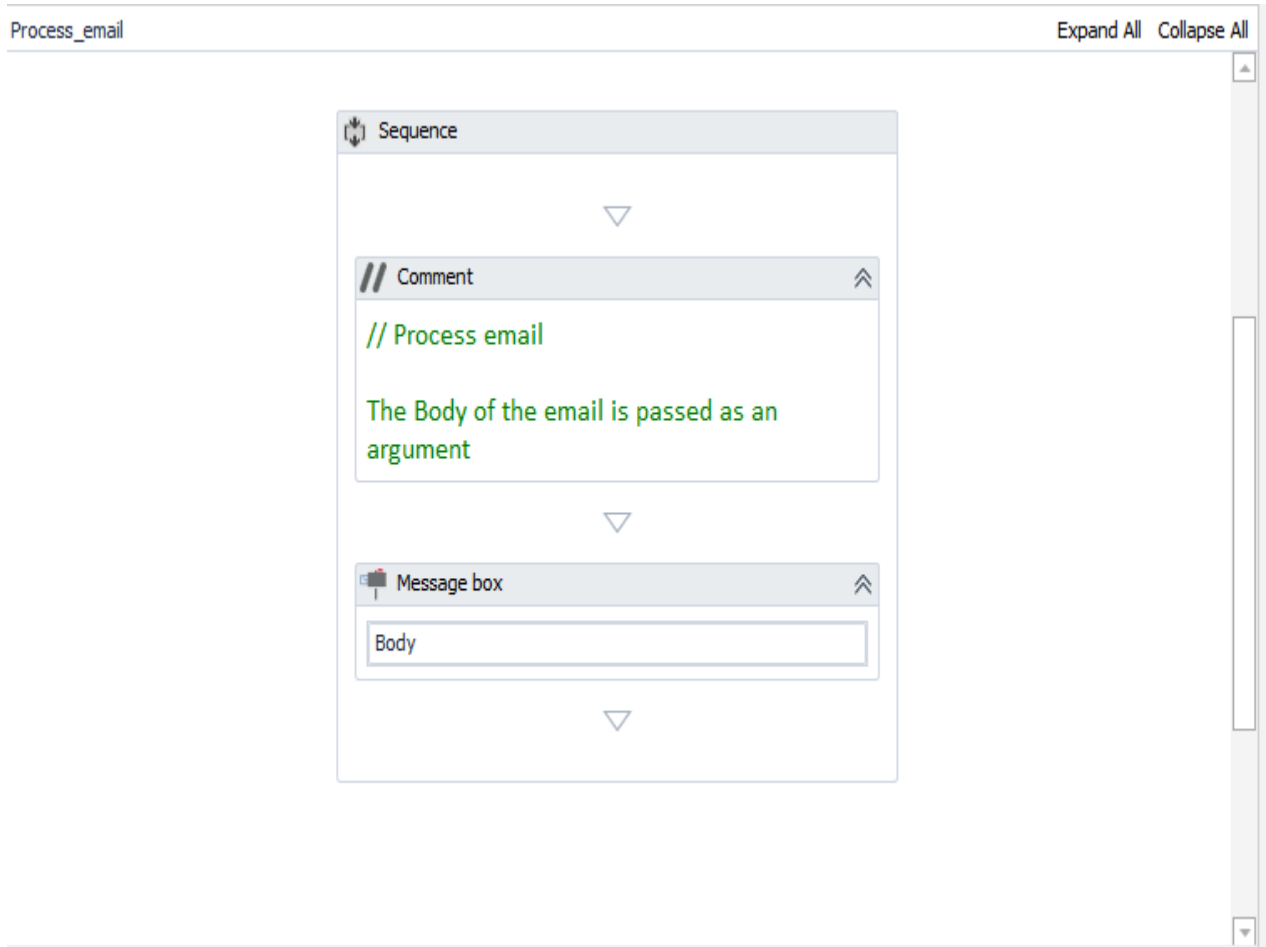
AIM : Automate the process of send mail event (on any email).

STEPS :

Email triggers are the foundation of any email automation process. In the attached *Mail_Trigger_Sample Workflow*, the Inbox is checked every 60 seconds for fresh emails.



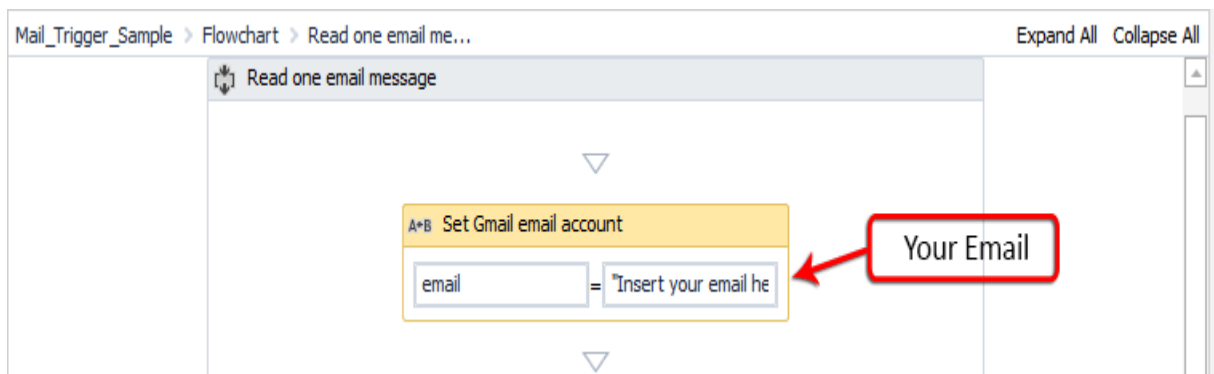
If a new email is detected and a certain condition is met (a specific Keyword is found in the Subject), then the message Body is passed as an argument to the *Process_Email* Workflow.



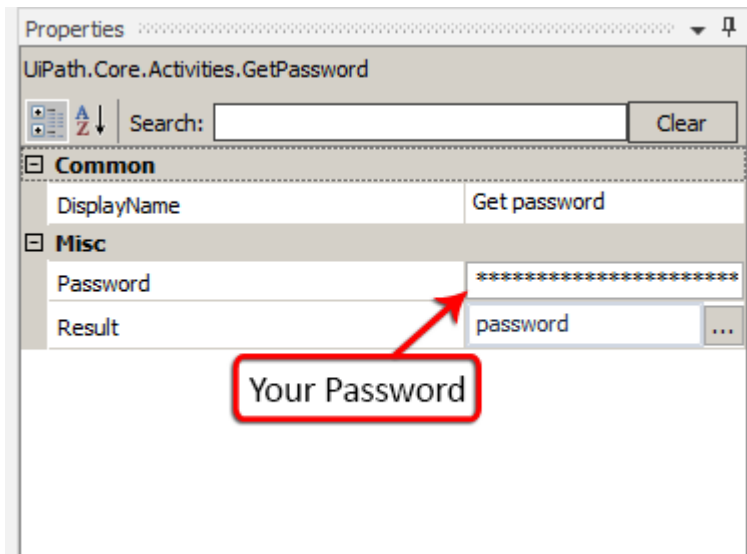
Workflow Steps:

1. Before running the *Mail_Trigger_Sample* workflow, you will need to set:

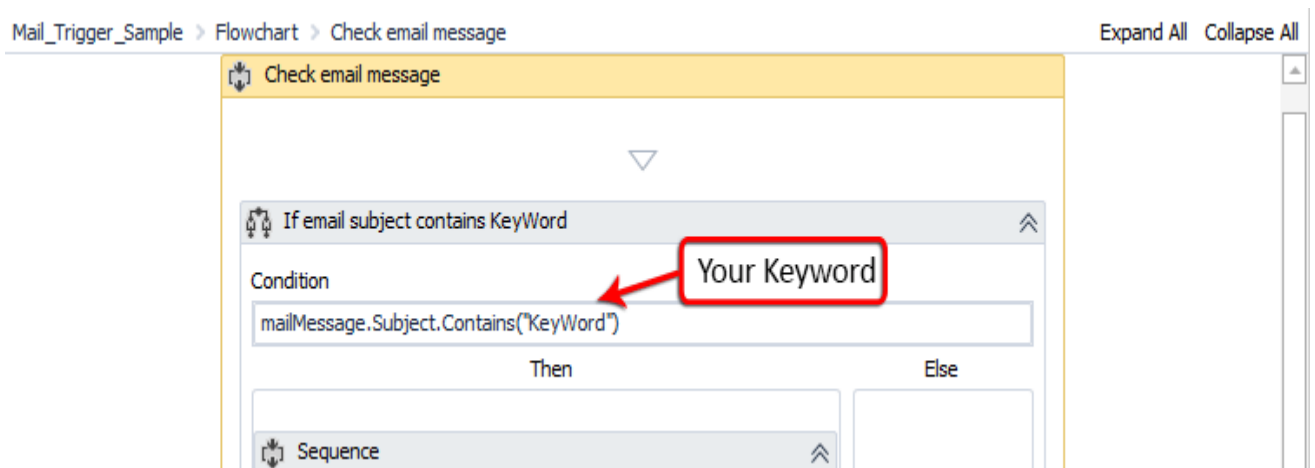
Email address:



Password:



Keyword:



Also, please make sure you set your *MailFolder*, *Port*, and *Server* in the **GetIMAPMailMessages** activity properties. By default, the **Workflow** is configured to work with the Gmail settings.

Properties

UiPath.Mail.IMAP.Activities.GetIMAPMailMessages

Search: Clear

Common

DisplayName: Get IMAP mail messages

TimeoutMS: ...

Host

EnableSSL: ☒

MailFolder: ...

Port: ...

Server: ...

Input

Top: ...

Logon

Email: ...

Password: ...

Options

DeleteMessages: ☐

MarkAsRead: ☒

OnlyUnreadMessages: ☒

Output

Messages: ...

2. When the **Workflow** is running, it checks your email for new messages (using the **GetIMAPMailMessages** activity). If no new email is found, it waits for 60 seconds and then it tries again. You can change the waiting time to other value, within the **Delay** activity.

Mail Trigger Sample * X Process Email

Mail_Trigger_Sample > Flowchart > Wait 1 minute Expand All Collapse All

Wait 1 minute

Delay 1 minute

Properties

System.Activities.Statements.Delay

Search: Clear

Common

DisplayName: Delay 1 minute

Misc

Duration: ...

Set time between trials

3. If a new email is found, it is marked as read (you can change that by un-checking the *MarkAsRead* property in the **GetIMAPMailMessages** activity)

4. The **Workflow** then checks if the email's subject contains the keyword. If there's a match, the **Process_Email Workflow** is invoked and the email body is passed as an argument (**Body**)

5. The **Process_Email Workflow** will pop up a window with the message body.

Note: The workflows are created & tested with UiPath Studio 8.

Attachments:

Mail_Trigger_Sample.xaml

Process_Email.xaml

Practical 9B

AIM : Automate the process of launching an assistant bot on a keyboard event.

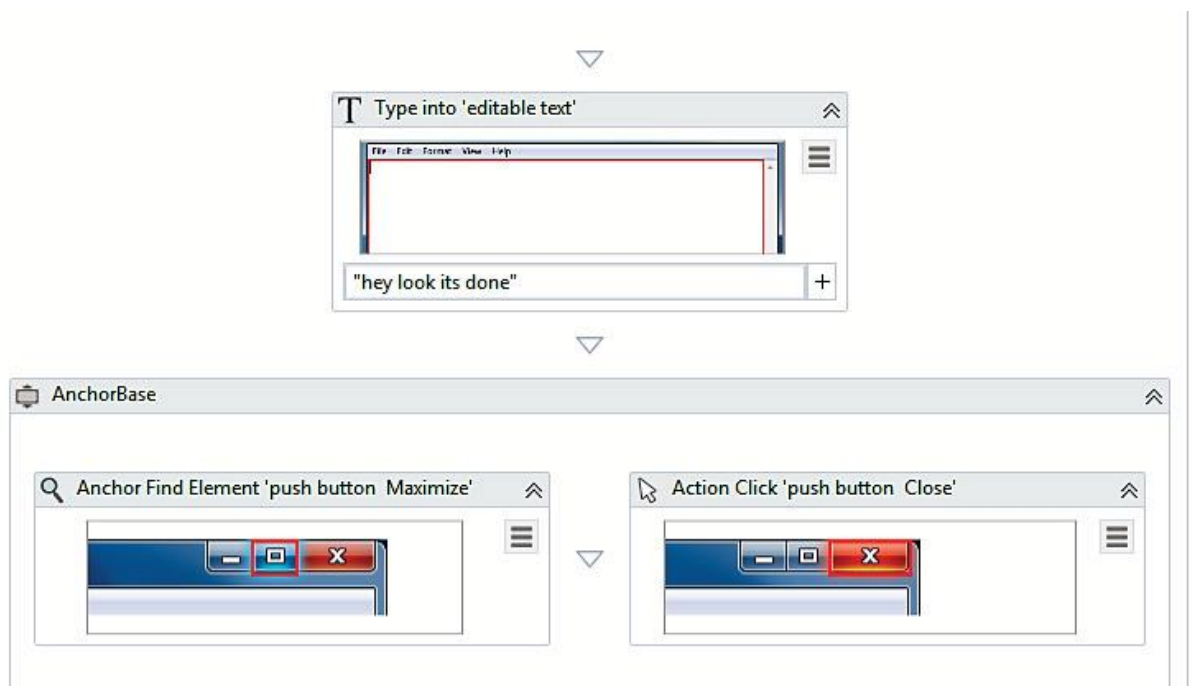
STEPS :

1. Drag and drop the Monitor events activity: In this step, we will just drag and drop the **Monitor events** activity into the workflow

2. Drag the Hotkey trigger activity: In the next step, we will use the **Hotkey trigger** activity for the user to start the automation process. Assign *Alt + W* to the hotkey so that, when the user presses this hotkey, the event will be executed:

3. Open Notepad and type into it: Our final step is to record the sequence of the steps to be performed. In this case, this is to open Notepad and then type into it. For that just use the help of the **Desktop** recorder. First, we double-click on the Notepad application in the window as shown in the screenshot. Select the **ClickType** as **CLICK_DOUBLE** from the **Properties** panel: After that, we record the typing action and close the Notepad window. Then click on **Do not Save** because you do not want to save your file. The sequence is shown in the following screenshot:

OUTPUT :



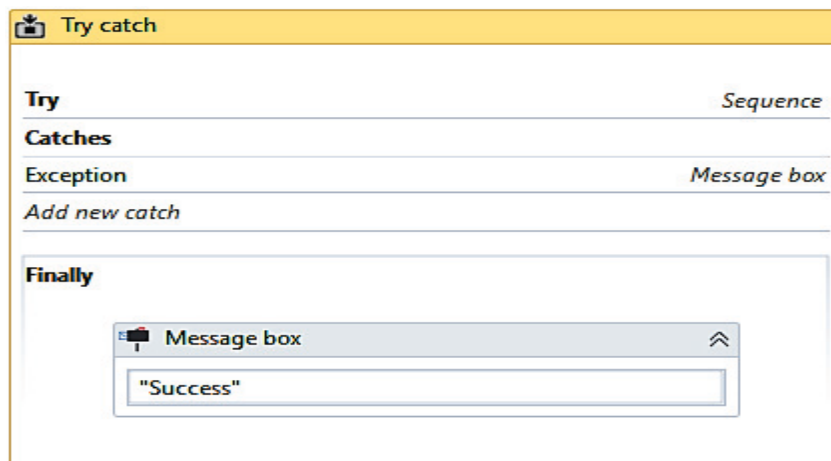
Practical 9C

AIM : Demonstrate the Exception handling in UiPath

STEPS :

1. Drag and drop the **Try catch** activity: Create a blank project. Drag and drop the **Flowchart** activity into the Designer panel. Search for the **Try catch** activity in the **Activities** panel and drag it into the **Flowchart**. Set it as the **Start** node:
2. **Try:** When we double-click on the **Try catch** activity, dragged and dropped inside the workspace, space for the **Try** activity appears
3. **Catches:** Inside the **Catches** activity, first we have to click on **Add new Catch** and then click on **Add Exception** option, from which we have to select the type of exception. In most cases, **System.Exception** is preferred. The following screenshot shows the types of exception. There are many more exceptions which can be viewed by clicking on the **Browse for Types** option:
4. Say the execution fails: for example, the **Click** activity is unable to be executed because of the unavailability of a UI element. In such a case, we can use the **Catches** block in order to either view the error that has occurred or for an alternative method to be used if that particular error occurs. As shown in the following screenshot, we will drop the activity in the **Catches** block. To print a message, we use a **Message box**:
5. When we click on **Add new catch**, we are asked to select the type of exception. We have selected **System.Exception**. Now inside the exception block, we have dropped a **Message box** activity. Entering `□□□□□□□□□□□□□□□□□□` will display the error that occurred during execution
6. **Finally:** When we have defined the exception for our sequence, the **Finally** block will always work, regardless of whether the execution was successful or not. Suppose we want to display a message to the user notifying that the process is complete. To make sure that the whole **Try catch** activity is executed, we will just drop a **Message box** activity in the area provided in the **Finally** block

OUTPUT :



Practical 9D

AIM : Demonstrate the use of config files in UiPath

THEORY :

When it comes to configuration, UiPath does not have any pre-built configuration file such as Visual Studio, but we can create one. It is considered to be one of the best practices to keep environment settings in a config file so that they can be easily changed by the user when required. Thus, when we create a project, the `UiPath.config` file that holds all the activities is created automatically. `UiPath.config` can be found in the folder where the project is saved. To access the folder, we can just open the Project, then copy the path (as shown in the following screenshot), and paste it into File Explorer:

STEPS :

You can also store your settings with the help of a spreadsheet or credentials. There are various parameters contained in the `UiPath.config` file. They are:

Name: This is the title of the project that is provided when creating a project in the create New Project window:

Description: When creating a project, a description is also required. You can add the description in the Create New Project window, as shown in the preceding screenshot.

Managing and Maintaining the Code Chapter 9

Main: This is the entry point for the project. It is saved as `Main.xaml` by default, but you can change its name from the **Project** panel. Also, you have multiple workflows for a project, it is necessary to attach all these files to the main file with the **Invoke Workflow File** activity. Otherwise, those files will not be executed:

Dependencies: These are the activities packages that are used in a project and their versions.

Excluded data: Contains keyword that can be added to the name of an activity to prevent the variable and argument values from being logged at the verbose level.

Tool version: The version of Studio used to create a project.

Managing and Maintaining the Code Chapter 9

Adding Credential: We can add particular settings that can be used further. For example, we can save the username and password to be used further, so this can be done with the help of the **Add credential** activity that can be found in the **Activities** panel, as shown in the following screenshot:

After adding credentials, type the required values in the **Properties** panel, as shown in the following screenshot:

Managing and Maintaining the Code Chapter 9

So, when the credentials are set, we can delete, secure, or request credentials, as shown in the following steps:

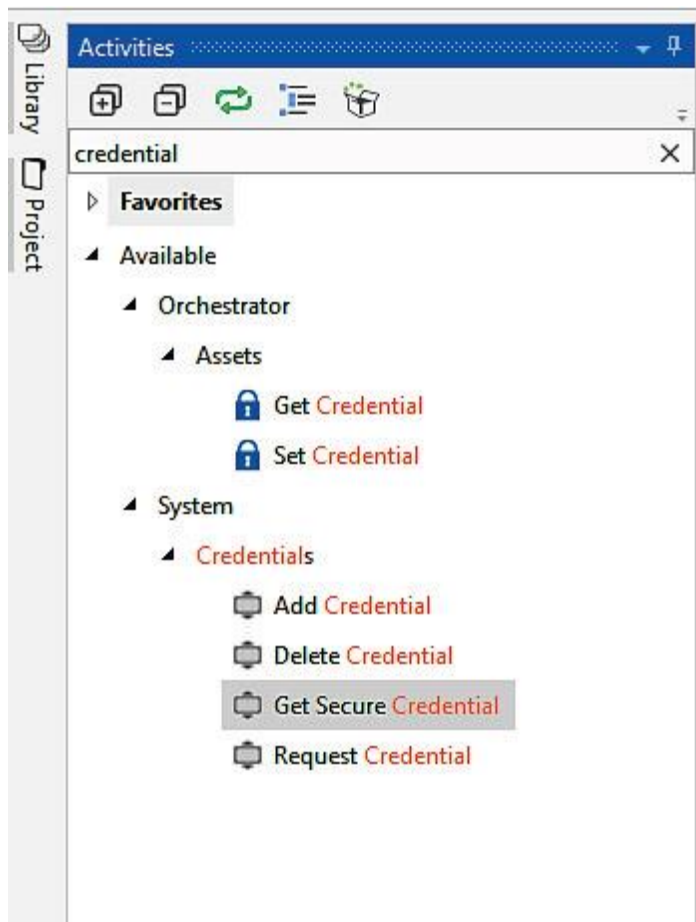
1. **Delete Credentials:** If we want to delete a credential then we can just drag and drop the **Delete Credentials** activity and then define the target for the credential:

Managing and Maintaining the Code Chapter 9

2. **Get Secure Credential:** This is used to get the values, that is, the username and password, that were set during the addition of a credential. We have to set the target the same as before; the output will be the username and password:

3. **Request Credential:** This is a property in which the robot displays a message dialog asking the user for username and password and stores this information as a string. This can then be used in further processes. The user can select OK to provide credentials or even cancel it if they do not want to provide credentials.

OUTPUT :



Practical 10A

AIM : Automate the process of logging and taking screenshots in UiPath

THEORY :

UiPath has a multi-process architecture that offers to execute each workflow separately in the executor. Executors are managed by UI robots. So, if any executor stops working, then the entire process will not be affected.

STEPS :

Common

- **Continue on error** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.
- **Delay before** - Delay (in seconds) between the time the previous activity is completed and the time this activity begins performing any operations. The default value is 0.2 seconds. Adding a delay between activities ensures that one activity has enough time to complete before the next activity begins.
- **Delay before screenshot** - Delay (in seconds) between the time the element is brought into the foreground and the time the screenshot is taken. The default value is 0.2 seconds.
- **DisplayName** - The name displayed for the activity in the Designer panel. A display name is automatically generated when you indicate a target.
- **Timeout** - Specify a number of seconds for which to wait for the activity to be executed before throwing an error. The default value is 30 seconds.

Input

- **Auto increment** - Select what to append to the filename when saving the screenshot in case of filename conflicts. For example, if a file with the same name as the one you save already exists, choosing **Index** keeps the file name, creating a new file for each screenshot and adding an index number to each, consecutively. The options are **None** (if a file with the same name already exists, replace it), **Index** (add a number to the filename, for example screenshot (1).png), or **DateTime** (add the date and time when the screenshot is taken to the filename in the format YYYY.MM.DD at HH.MM.SS).
- **File name** - The name of the file where the screenshot of the specified UI element will be saved.
- **Target** - Before indicating on screen the application you want to automate, this field is set to (null). Once the target is indicated, all properties regarding the element that was indicated are displayed.
 - **Target.Click Offset** - Specifies an offset for the click activity, which can be further configured.
 - **Target.Click Offset.Anchoring Point** - Describes the starting point of the cursor to which offsets from OffsetX and OffsetY properties are added. The following options are available: TopLeft, TopRight, BottomLeft, BottomRight, and Center. By default, Center is selected.
 - **Target.Click Offset.OffsetX** - Horizontal displacement of the cursor position according to the option selected in the Position field. This field supports only Int32 variables.
 - **Target.Click Offset.OffsetY** - Vertical displacement of the cursor position according to the option selected in the Position field. This field supports only Int32 variables.
 - **Target.Fuzzy selector** - The parameters for the fuzzy selector.
 - **Target.Native text** - The text that is used to identify the target element.

- **Target.Selector** - The selector that is generated for the indicated element.
- **Target.Targeting methods** - The selector types that you want to use for identifying the element. This property can be set to any combination of Selector, Fuzzy selector, or Image.
- **Target.Visibility check** - Checks whether the UI element is visible or not. There are three options available:
 - None - Does not check for visibility.
 - Interactive (for Fuzzy Selector) - Checks if the element is potentially visible, ignoring page scroll and obstructions by other apps, or the fact that the application is minimized. This check is useful when trying to ensure that you are not targeting invisible elements that exist in the DOM but are hidden.
 - Fully visible - Checks if the UI element is visible or not.
- **Target.Wait for page load** - Before performing the action, wait for the application to become ready to accept input. The following options are available:
 - None - Does not wait for the target to be ready.
 - Interactive - Waits until only a part of the app is loaded.
 - Complete - waits for the entire app to be loaded.
- **Target.Window selector (Application instance)** - The selector that is used for the application window. Only applicable when the window attach mode is set to Application instance.

Input/Output Element

- **Input Element** - The Input UI Element that defines the screen element that the activity will be executed on.
- **Output Element** - Output a UI Element to use in other activities as an Input UI Element.

Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

Output:

- **Saved file path** - The full path to the file where to save the screenshot, including the appended suffix, if applicable. This also dictates where the **Auto increment** property saves the indexed screenshot files.
- **Saved image** - The screenshot saved as Image; used when Output is set to image.

Practical 10B

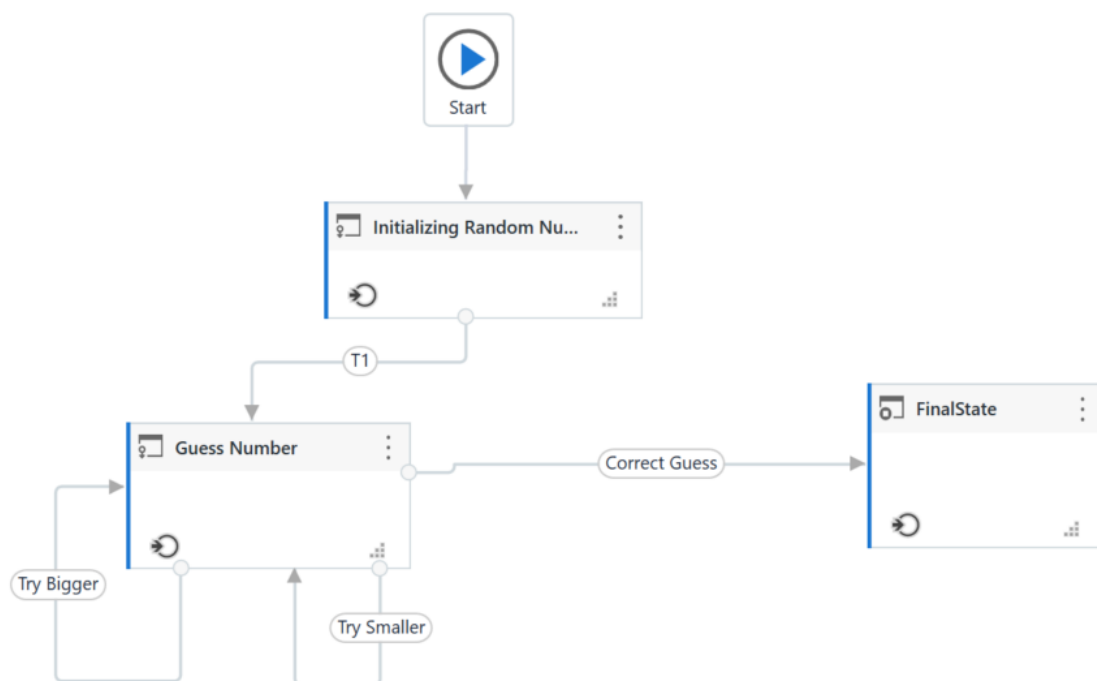
AIM : Automate any process using State Machine in UiPath

STEPS :

1. Create a blank process and, on the **Design** tab, in the **File** group, select **New > State Machine**. The **New State Machine** window is displayed.
2. In the **Name** field type a name for the automation, such as "First State Machine", and leave the default project location or add a subfolder. Click **Create**. The **Designer** panel is updated accordingly.
3. Create two integer variables, InitialGuess and RandomNumber. The first variable stores your guess, while the second stores the random number.
4. Add a **State** activity to the **Designer** panel and connect it to the **Start** node. This is the initial state, and it is used to generate a random number.
5. Double-click the activity. This **State** activity is displayed expanded in the **Designer** panel.
6. In the **Properties** panel, in the **DisplayName** field, type Initializing Random Number. This enables you to easily tell states apart.
7. In the **Entry** section, add an **Assign** activity.
8. In the **To** field, add the RandomNumber variable.
9. In the **Value** field, type new Random().Next(1,100). This expression generates a random number.
10. Return to the main project view and add a new **State** activity.
11. Connect it to the previously added activity.
12. Double-click the last added **State** activity. This activity is displayed expanded in the **Designer** panel.
13. In the **Properties** panel, in the **DisplayName** field, type Guess Number. This state is used to prompt the user to guess a number.
14. In the **Entry** section, add an **Input Dialog** activity.
15. Select the **Input Dialog**, and in the **Properties** panel, add an appropriate **Label** and **Title** to prompt the user to guess a number between 1 and 100.
16. In the **Result** field, add the InitialGuess variable. This variable stores the user's guess.
17. Return to the main project view and create a transition that points from the Guess Number state to itself.
18. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
19. In the **Properties** panel, in the **DisplayName** field, type Try Smaller. This message is displayed on the arrow, enabling you to run through your automation easier.
20. In the **Condition** section, type InitialGuess > RandomNumber. This verifies if the user's guess is bigger than the random number.
21. In the **Action** section, add a **Message Box** activity.
22. In the **Text** field, type something similar to "Your guess is too big. Try a smaller number." This message is displayed when the user's guess is bigger than the random number.
23. Return to the main project view and create a new transition that points from the **Guess Number** state to itself.
24. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
25. In the **Properties** panel, in the **DisplayName** field, type "Try Bigger". This message is displayed on the arrow, enabling you to run through your automation easier.
26. In the **Condition** section, type InitialGuess < RandomNumber. This verifies if the guess is smaller than the random number.
27. In the **Action** section, add a **Message Box** activity.

28. In the **Text** field, type something similar to "Your guess is too small. Try a bigger number." This message is displayed when the users guess is smaller than the random number.
29. Return to main project view and add a **Final State** activity to the **Designer** panel.
30. Connect a transition from the **Guess Number** activity to the **Final State**.
31. In the **Properties** panel, in the **DisplayName** field, type "Correct Guess".
32. In the **Condition** field, type `InitialGuess = RandomNumber`. This is the condition on which this automation steps to the final state and end.
33. Double-click the **Final State** activity. It is displayed expanded in the **Designer** panel.
34. In the **Entry** section, add a **Message Box** activity.
35. In the **Text** field, type something similar to "Congratulations. You guessed correctly! The number was " + `RandomNumber.ToString` + "." This is the final message that is to be displayed, when the user correctly guesses the number.


OUTPUT :




Practical 10C

AIM : Demonstrate the use of publish utility.

STEPS :

1. In Studio, create a new project.
2. In the **Design** ribbon tab, click **Publish**. The **Publish** window opens. Notice that the window's title bar changes depending on the context:
 - **Publish Process** when publishing a process;
 - **Publish Library** when publishing a library project;
 - **Publish UI Library** when publishing a UI library project;
 - **Publish Test Cases** when publishing test cases.
 - **Publish Templates** when publishing templates.
3. In the **Package Properties** tab:
 - Enter a name for the package. The drop-down list contains up to 5 of the most recent names of packages that you previously published.
 - In the **Version** section, review the **Current Version** of your project, and type a **New Version** if needed. Check the **Is Prerelease** box to mark the version as alpha. Please note that this automatically changes the project's version schema to semantic. When publishing a new version of the file locally, make sure that the custom location does not already include a file with the same proposed version number. For more details about project versioning, check the **About Automation Projects** page.
 - Optionally, use the **Project Icon** option to define a custom icon for the project. You can browse to and select a file, or enter a path or public URL to a jpeg, jpg, or png file up to 1MB in size. After the project is published, the icon is displayed as follows:
 - For processes, in the Assistant next to the process name, making it easier to identify it in the list of processes.
 - For templates, next to the template in **Home** (Studio Backstage View) > **Templates**.
 - For libraries, next to the package in the **Manage Packages** window in Studio.
 -  The icon is not visible in Manage Packages if a local file is used for a library published to Orchestrator or to a feed that does not support embedded icons. In this case, specify the icon using a URL.
 - In the **Project tags** box, you can add one or more tags to the project, either by creating new ones or by reusing tags already defined in Orchestrator. There are two types of tags: **labels** and **properties** (key-value pairs). Tags are included in the published package and they help describe and categorize projects. For example, they can refer to the automated application (an Excel label) or the department (a department:accounting key-value property). When you start typing, possible matches are suggested from already defined tags, and you can reuse one by selecting it from the list of matches. For a property match, the key followed by the : (colon) character is displayed first, and the associated values are displayed after you select the key, To add a new tag, after you enter the name, click the entry with the plus sign next to the name. Separating strings with the : (colon) character enables you to add properties, while entries that don't contain a : add labels. Labels and key-value properties are limited to 256 characters. Tag names can't contain these characters: <, >, %, &, \, ?, /, :. Project tags can be automatically applied to processes in Orchestrator. For more information about using tags, see Organizing resources with tags in the Orchestrator guide.

- In the **Release Notes** text box, enter details about the version and other relevant information. Release notes for published projects are visible in the **Packages** section in Orchestrator. Please note that the Release Notes field accepts a maximum of 10,000 characters.
- 4. Click **Next**.
If you are publishing a template, the **Template info** tab opens next (step 5). Otherwise, proceed to step 6.
- 5. (For templates only) In the **Template info** tab, provide the following information, and then click **Next**:
 - **Name** - The name of the template.
 - **Description** - The template description in the **Templates** tab.
 - **Default Project Name** - The default project name when creating a new project using this template.
 -  **Note:** Avoid using punctuation marks, separator characters, and characters that are not allowed in file names. These characters may be removed from the default name when the template is used.
 - **Default Project Description** - The default description when creating a new project using this template.

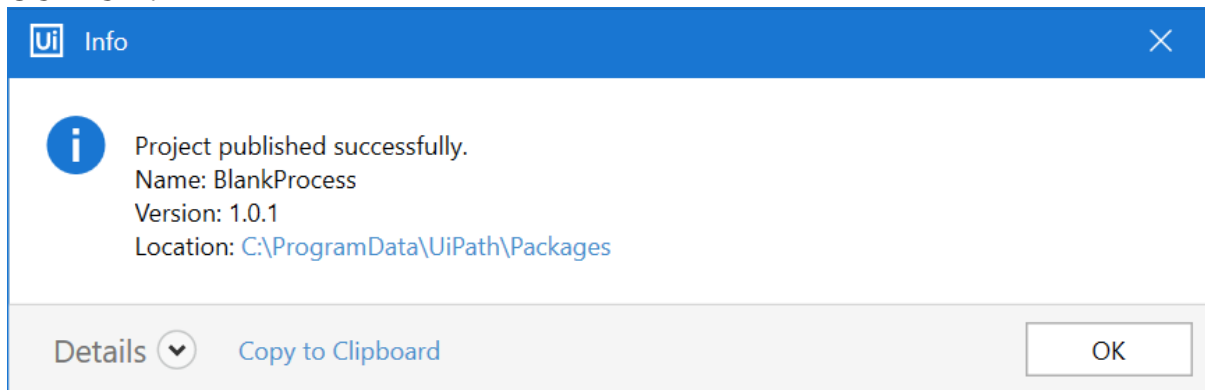
In the **Publish options** tab, select where to publish the project. The available options depend on the type of project you are publishing:

- For **processes** (including StudioX projects):
 - **Orchestrator Tenant Processes Feed, Orchestrator Personal Workspace Feed** and, if a first-level folder with a separate package feed or one if its subfolders is selected from the folders menu in the Studio status bar, the feed for that folder hierarchy. These options are available if Studio is connected to Orchestrator. The Orchestrator Personal Workspace Feed is available only if your user has the Personal Workspace feature enabled in Orchestrator.
If the Personal Workspace or a folder from a hierarchy with a separate package feed is selected in Studio, the feed for that folder is the default option. Otherwise, the tenant feed is the default option. If you already published a project in the current session, the last publish location you used is the default selection until you close Studio or change the Orchestrator folder from the Studio status bar.
 - **Assistant (Robot Defaults)** - the default package location for the Robot and Assistant, C:\ProgramData\UiPath\Packages. Projects published here automatically appear in the Assistant. The option is not available if Studio is connected to Orchestrator.
 - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.
- For **test cases**:
 - The same options that are available for processes, with the exception of Orchestrator Personal Workspace Feed.
- For **libraries** and **UI libraries**:
 - **Orchestrator Tenant Libraries Feed or Orchestrator Shared Libraries Feed** - available if Studio is connected to Orchestrator. The available option depends on whether the tenant libraries feed is enabled in Orchestrator.
 - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.
- For **templates**:
 - **Orchestrator Tenant Libraries Feed or Orchestrator Shared Libraries Feed** - available if Studio is connected to Orchestrator. The available option depends on whether the tenant libraries feed is enabled in Orchestrator.
 - **Local** - the location for publishing templates locally, by default: C:\Users\User\Documents\UiPath\templates.
 - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.

If you are publishing a library or any project with the Windows or cross-platform compatibility except for templates, additional settings are available in the **Publish options** tab under **Compilation Settings**:

- *(For libraries only)* **Activities Root Category** - enter a name for the category under which the reusable component will be listed in the **Activities** panel.
- **Include Sources** - select this option to package all .xaml sources within the published package, including workflows that were previously made private. For Windows - Legacy libraries, the files are saved in the generated assembly file and in the lib\net45 folder in the .nupkg file. For Windows and cross-platform libraries and processes, the files are saved in the content folder in the .nupkg file.
- **Remove Unused Dependencies** - select this option to remove all installed packages that are not referenced in the project. This option is not available for Windows-legacy processes.
- *(For Windows-legacy libraries only)* **Compile activities expressions** - select this option to compile and package all activities expressions. This results in an improved execution time.
- *(For Windows - legacy libraries only)* **Ready to Run** - select this option to optimize the generated assemblies for faster JIT compilation at runtime.
 - Click **Next** to advance to the **Certificate signing** tab, or **Publish** to publish your project.
 - *(Optional)* In the **Certificate Signing** tab, add a local **Certificate Path** next to the **Certificate** box. Furthermore, add the **Certificate Password** and an optional certificate **Timestamp** if needed. For more details, check out the [Signing](#)
 - Click **Publish**. A NUPKG file is created and uploaded to Orchestrator, the custom NuGet feed, or saved in the local directory. Depending on the project, the package contains:
 - For template projects and processes with the Windows - Legacy compatibility, the project source files.
 - For libraries and projects with the Windows or Cross-platform compatibility, compiled DLL files.
 - If the project is published successfully, the **Info** dialog box is displayed and the project is copied to the NuGet location set in the NuGetServerUrl parameter, in the UiPath.settings file.

OUTPUT :



Practical 10 D

AIM : Create and provision Robot using Orchestrator

THEORY :

When deploying multiple Standard Robots from the same machine on Orchestrator, you need to have the same **Machine Name** and **Machine Key** for each. To retain the values in the fields, you can click **Create Another** in the **Add Robot** window. Alternatively, you can copy the **Machine Name** and **Machine Key** from an already deployed Robot by clicking **More Actions > Duplicate**.

STEPS :

1. In Orchestrator, on the **Robots** page, click **Add**. The options to add a **Standard Robot** or a **Floating Robot** are displayed.
2. Click the **Standard Robot** button. The **Create a New Standard Robot** window is displayed.
3. Select the **Standard Machine** you want to register your Robot on. There are two possibilities:
 - a. You had already created the machine on the **Machines** page beforehand. In this case, you can select it from the **Machine** drop-down list.
 - b. You had not created the machine. In this case, simply type the name of a new one on the **Machines** field and click the **Provision Machine** button. Note that this step also adds the machine in the **Machines** page.
4. In the **Name** field, type any name for the Robot.
5. In the **Domain\Username** field, type the username that is used to login to the specified machine. If the user is in a domain, you are required to specify it in a DOMAIN\username format. You must use short domain names, such as desktop\administrator and NOT desktop.local/administrator.
6. **(Optional)** Add the Windows password for the specified username.
7. Select the desired robot type from the **Type** drop-down list. For more information, see the [**About Robots**](#) page.
8. **(Optional)** Add a description for the Robot. We recommend populating this field, especially when dealing with an environment with many robots.
9. Click **Create**. The Robot is now displayed on the **Robots** page and provisioned to Orchestrator, but it is offline.