LAB EXERCISE MANUAL

List of Exercises

Serial Number	Exercise Name
1	Specific Properties of Click, Type Into, and Send Hotkey
2	Find and Exists Activities
3	Use of Excel Activities to Read and Write
4	Advanced Activities of Excel
5	PDF Extraction
6	Email Automation
7	Extraction Wizards

Table of Contents

Setup UiPath Academic Alliance Edition

UiPath Learning offers its partner organizations the opportunity to use the UiPath software at no charge through a dedicated Learning license, which is free to students and educators alike, for teaching, learning and research.

- The license entitles the user to access UiPath Studio, the automation development software application, along with the UiPath Development Robot to allow for testing and execution of the automations.
- The Academic Alliance Edition of UiPath Studio is a locked version of the software aligned to the current Studio course resources provided by the UiPath Academic Alliance Partners and UiPath Learning Partners. Moreover, the current UiPath Certified Professional exams are based on this version.
- One license key allows for the activation of the UiPath Studio software on 2 different machines.
- The license type granted is Named User, to be applied solely to User Mode installs. Instructions for activating the Academic Alliance Edition license will be provided accordingly.
- Each Academic Alliance Edition license expires after 365 days upon its activation, and it is subject to being extended as long as the Academic Alliance partnership is maintained.

Using the Academic Alliance Edition ensures that you have the appropriate software version installed which aligns with the curricula and hands-on lab materials. This will avoid any disruptions to your learning experience. This will also ensure that projects can be shared between students and educators without software version differences.

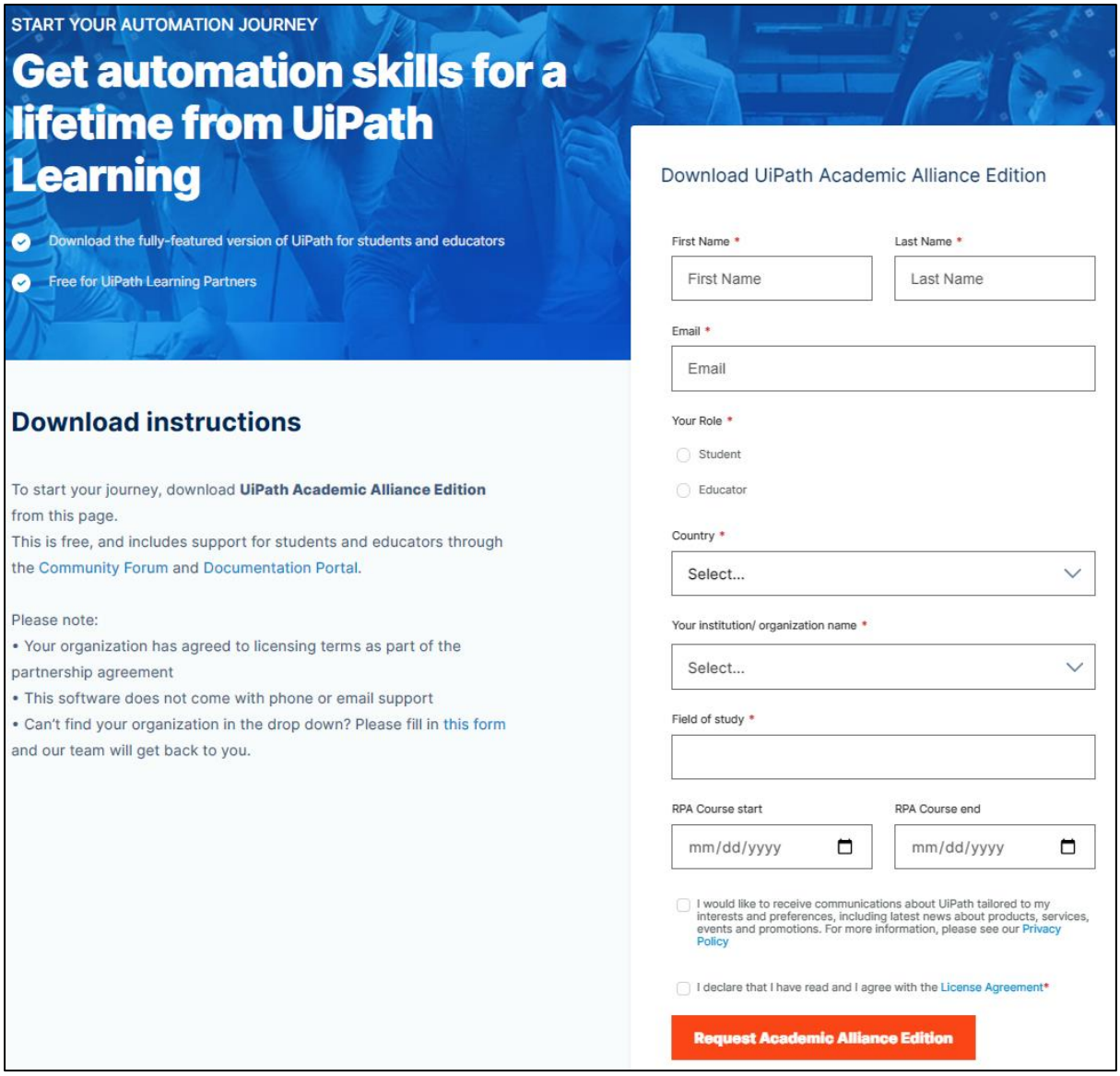
1.1 Personal Machine Setup

If you have installed UiPath Studio Community Edition previously, please uninstall it first before installing the UiPath Studio Academic Alliance Edition. Failure to do this may result in both editions of the software not working correctly.

If you have previous versions of the Academic Alliance Edition (19.4 or 19.10), please install the new one above it (i.e. no need to uninstall).

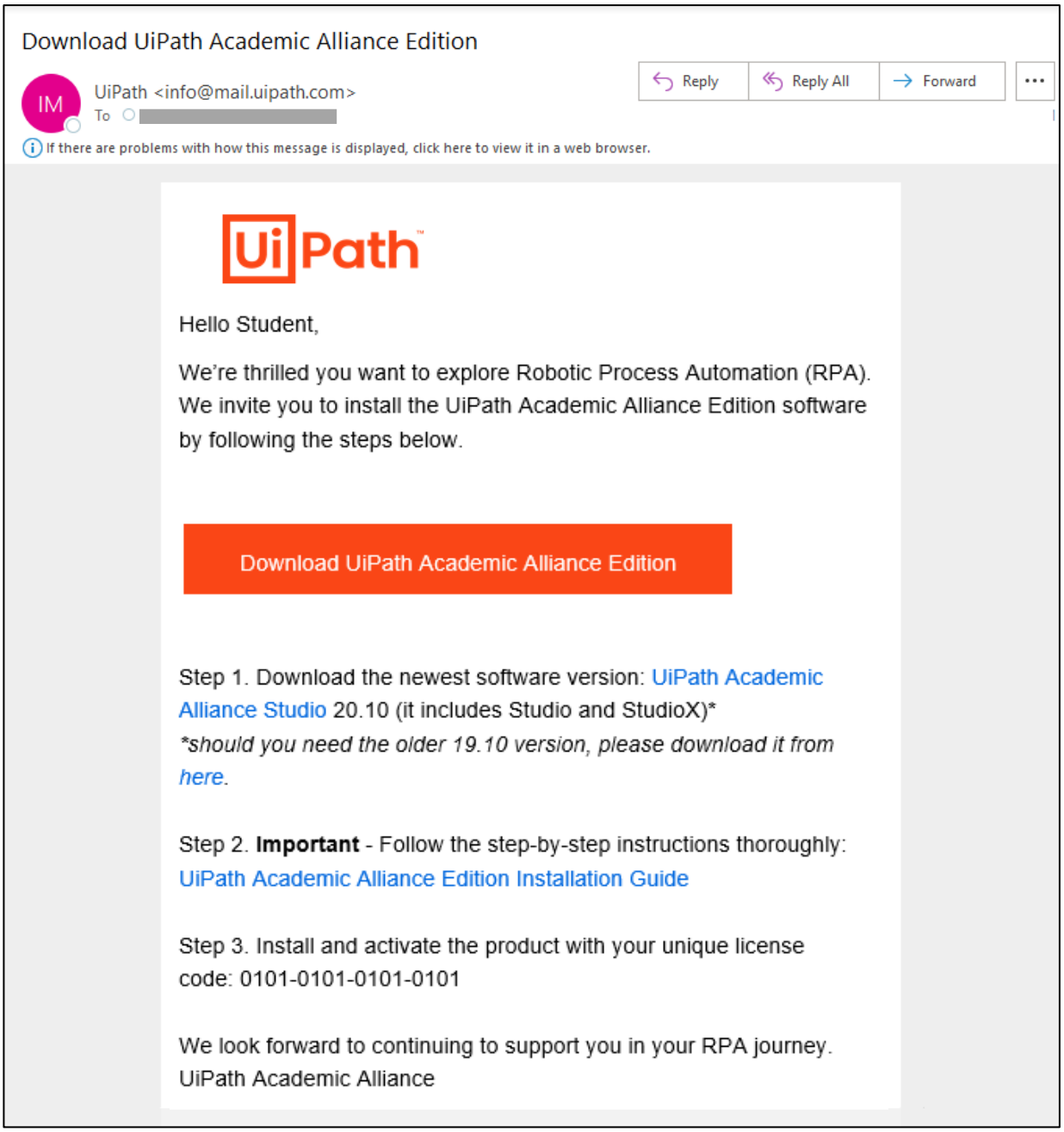
Step 1: To download your free copy of UiPath software, open the URL: https://www.uipath.com/landing/academic-studio-download It is a private form designated only for the students and educators who are affiliated to the UiPath Academic Alliance & UiPath Learning Partner institutions.

Step 2: Fill in your contact details and choose the institution you belong to.

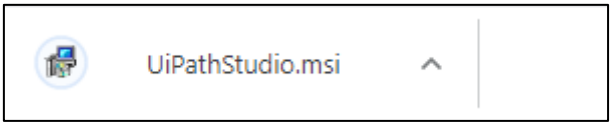


Step 3: Click **Request Academic Alliance Edition**. Then, check your email in a couple of minutes. You will receive a link to download the software by email. The email you receive contains the installation guide, the link to download the *UiPathStudio.msi* file along with your dedicated license key.

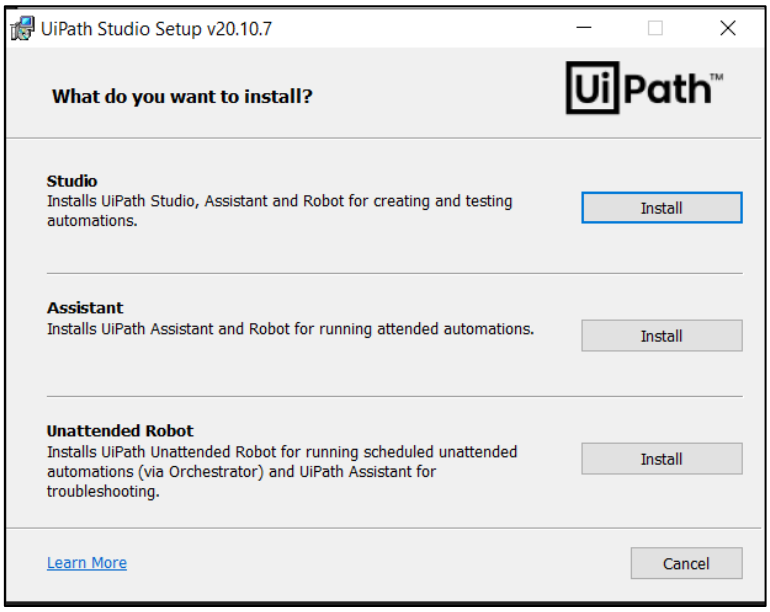
Make sure you follow the steps in the email and carefully follow this guide.



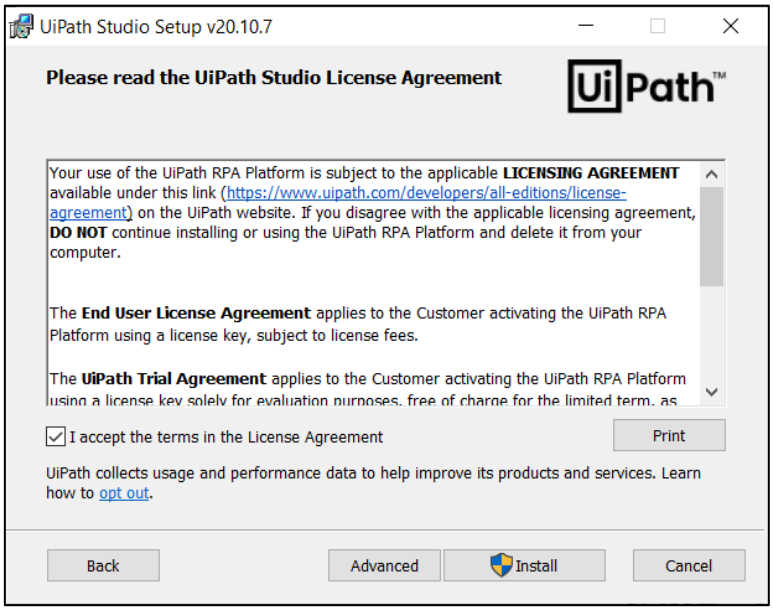
The file will begin downloading. Click to open the **UiPathStudio.msi** once it is downloaded.



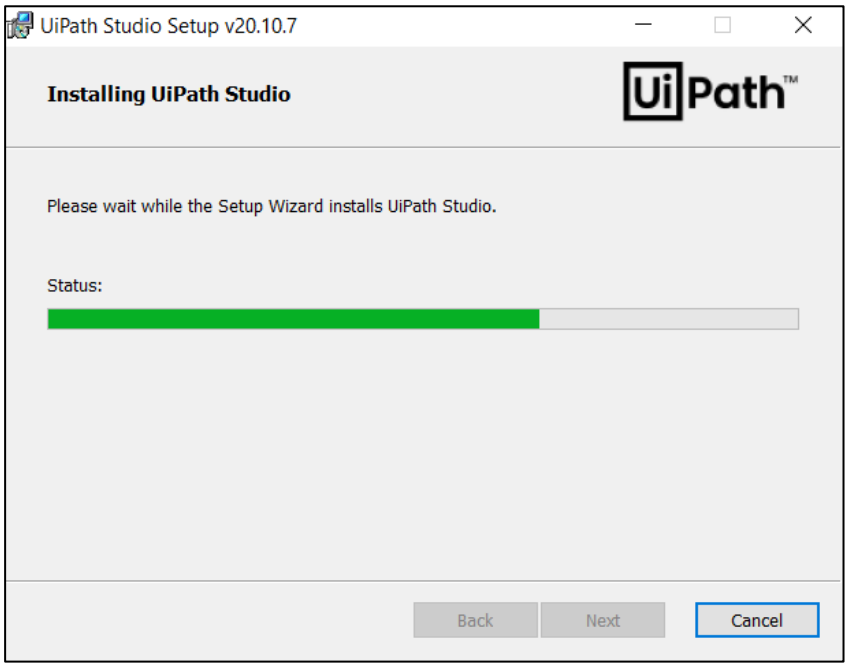
Step 4: Click on the *Install* button right beside Studio.



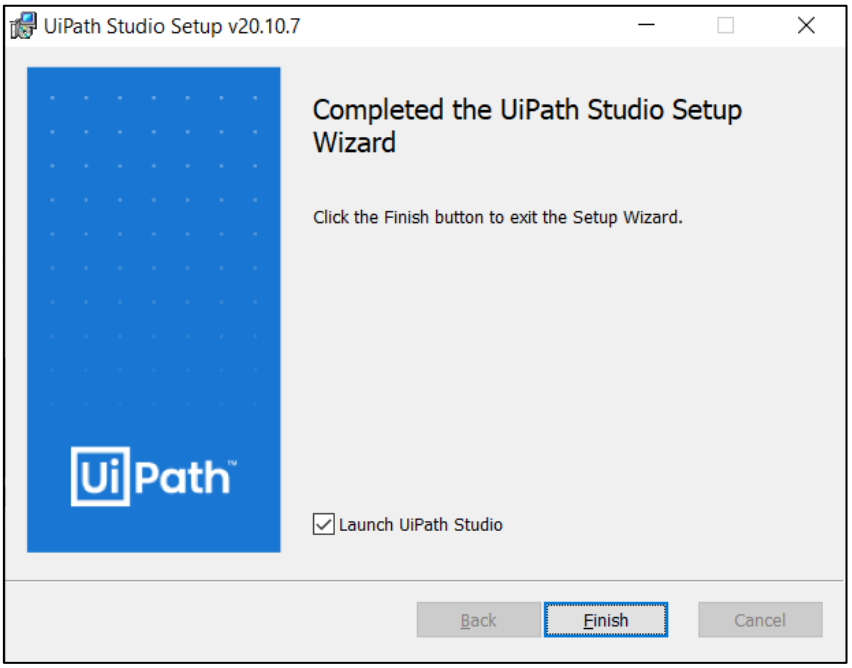
Step 5: Accept the terms of the License Agreement and click on *Install*.



Step 6: The software will begin its installation.

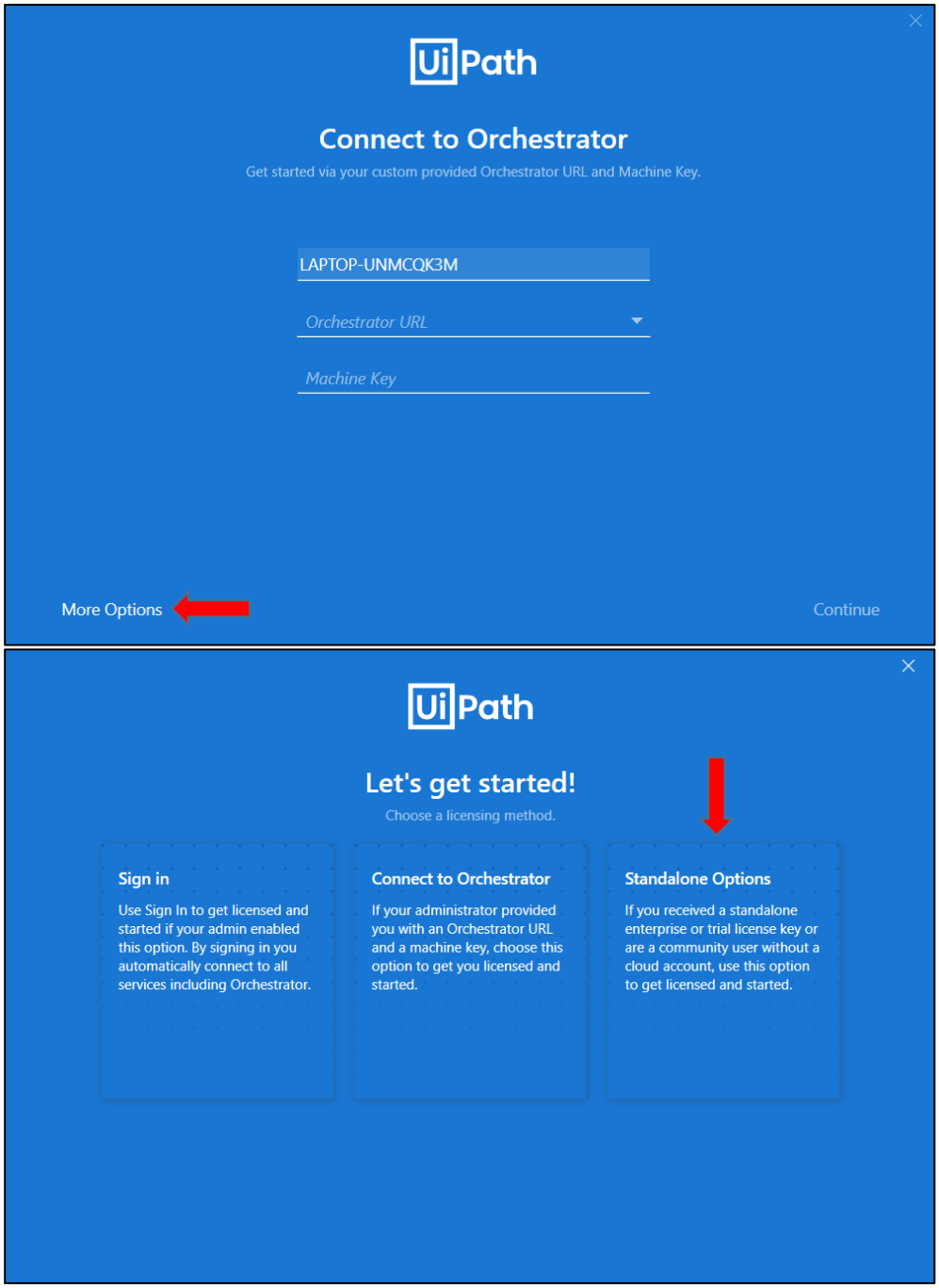


Step 7: Click **Finish**.



If "Launch UiPath Studio" was checked, then UiPath Studio is launched. If not, you can search in your Windows explorer for UiPath Studio to open it.

Step 8: In the "Welcome to UiPath Studio" window, click on *More Options* and then *Standalone Options* to activate your dedicated Academic Alliance Edition of Studio with your license key.

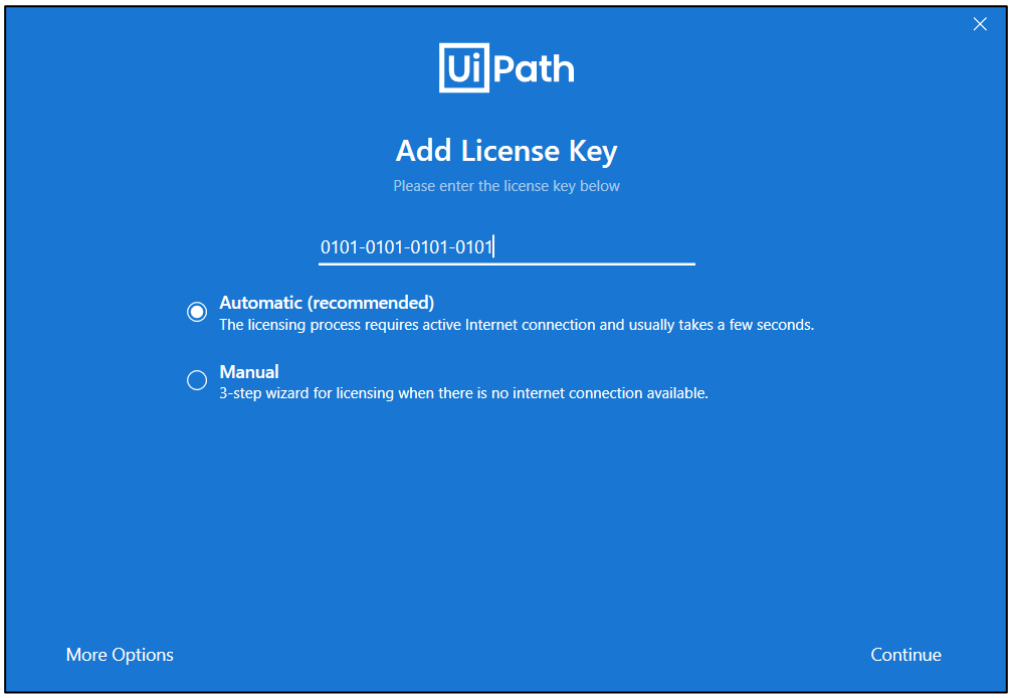


Step 9: The UiPath Registration window is displayed

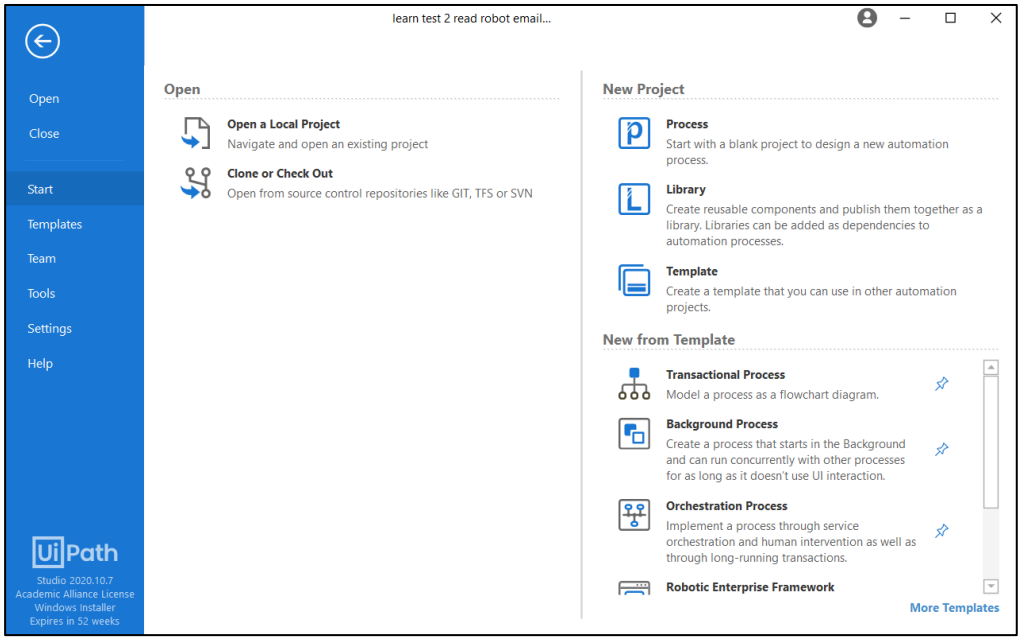
a. Enter the **License Key** you received by email, including the dashes.

b. Select **Automatic** activation, but ensure you have internet connection.

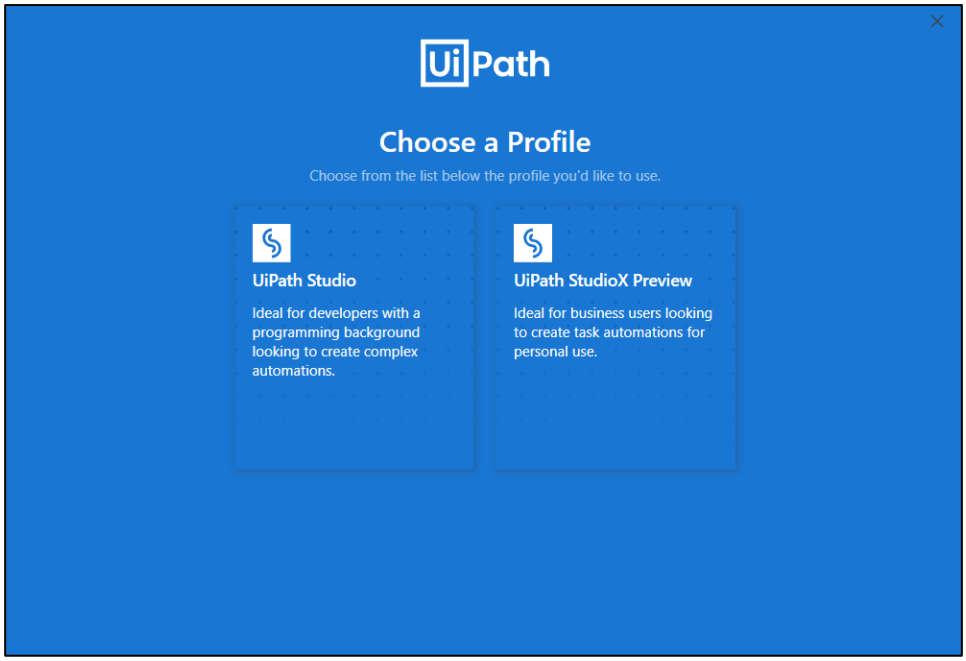
c. Click **Continue**. Your UiPath license is now activated.



Step 10: Now, you can start using UiPath Studio and StudioX software to create automation workflows.



Step 11: You can switch between the two profiles (Studio or StudioX) at any time by going to Settings – License and Profile – View or Change Profile. Choose whether you want to use UiPath Studio or StudioX.


UiPath
Choose a Profile
Choose from the list below the profile you'd like to use.
UiPath Studio
Ideal for developers with a programming background looking to create complex automations.
UiPath StudioX Preview
Ideal for business users looking to create task automations for personal use.

1.2 Networked Computer Lab Setup

a. Instructor/IT installs the software on the computer lab machine(s); making it available to each student, but they do NOT apply a license key to UiPath
b. If this is the first time the student is logging into the physical machine, they will open UiPath Studio and apply their unique license key. The activation steps can be found here.
c. If the student has logged into the physical machine before using the same domain/username then the activation will occur automatically each time UiPath Studio is opened. Note that in some scenarios, the student may still be prompted to enter their unique license key when opening UiPath Studio.

Special Case: Cloud-Based Computer Labs

A cloud-based computer lab may be used to provision each student with an identical machine on which to use UiPath Studio. While there are many options for a cloud-based computer lab, two of the most popular options are listed below:

- Azure Lab Services - https://labs.azure.com
- Amazon Workspaces - https://aws.amazon.com/workspaces

Both options charge by the hour of use. It is recommended for the instructor or IT administrator to deploy a standard Windows 10 "template" and modify it by downloading and installing UiPath Studio.

Special Case: UiPath Studio on Macintosh

UiPath can only run on Windows Machine, therefore Mac users must either use another machine (e.g., a lab computer) or they must load a Windows Operation System onto their Mac. The list below includes several tools that may be used for running Windows on a Mac Computer:

- Apple Bootcamp (free; loads Windows upon booting) - https://support.apple.com/boot-camp
- Parallels Virtual Machine (paid) - https://www.parallels.com/products/desktop
- VMWare Fusion Virtual Machine (paid) - https://www.vmware.com/products/fusion.html
- VirtualBox & Virtual Machine – https://www.virtualbox.org | https://developer.microsoft.com/en-us/windows/downloads/virtual-machines

Additional guidance can be found at this link: https://www.uipath.com/community/rpa-community-blog/how-to-use-uipath-studio-on-mac

For questions or issues please raise a ticket here: https://www.uipath.com/company/contact-us/contact-learning

Follow Academic Alliance conversations on the UiPath Forum at https://forum.uipath.com/c/learn/academic-alliance/113

Lesson 2 – Input Activities and Input Methods

2.1 Specific Properties of Click, Type Into and Send Hotkey

2.1.1 Objective

Build a workflow that uses specific properties of Click, Type Into, and Send Hotkey activities to input data in a Notepad file.

- Open a Notepad file and type “Welcome to UiPath.”
- Minimize the Notepad file after the delay of 10 seconds using the ‘SimulateClick’ and DelayBefore property.
- Type “Let’s explore the new world of automation” using the SendWindowMessages’ and ‘EmptyField’ property.
- Select all the text in the file using the right-click option.
- Change the Font type to ‘Times New Roman’, Font style to ‘Italic’, and Font size to ‘15’. Keep the delay of 2 seconds before each change using the DelayAfter property.
- Close the Font window using Enter as a special key.

2.1.2 Process Overview

- START
- Use an Open Application activity to indicate a Notepad file.
- Use a Type Into activity to enter “Welcome to UiPath”.
- Minimize the Notepad window using the ‘Simulate Click’ method with a delay of 10 seconds using the ‘DelayBefore’ property of the Click activity.
- Use a Type Into activity to enter “Let’s explore the new world of automation” using the ‘Send Window Messages’ and ‘Empty Field’ property.
- Use a Click activity to hit right click on the notepad editor window using the ‘MouseButton’ property to get the context menu.
- Use a Click activity to select the "Select All" option from the context menu.
- Use Click, Attach Window, and Type Into activities to change:
    - Font type to ‘Times New Roman’ and add a delay of 2 seconds using the ‘DelayAfter’ property of the Type Into activity.

  - Font style to ‘Italic’ with a delay of 2 seconds using the ‘DelayAfter’ property of the Type Into activity.
  - Font size to ‘15’ with a delay of 2 seconds using the ‘DelayAfter’ property of the Type Into activity.
- Use a Send Hotkey activity to close the Font window of the Notepad window.
- STOP

2.1.3 Step by Step Process

Step 1: Open a new Notepad file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as “Specific Properties of Click, Type Into, and Send Hotkey”.

Step 4: Drag a Sequence activity from the Activities panel and drop it in the Designer panel. Rename it to “Sequence-Specific properties of Click, Type Into, and Send Hotkey activities."

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu. Add an annotation: “This block of code uses specific properties of Click, Type Into, and Send Hotkey activities to input data in a Notepad.”

Step 6: Drag and drop an Open Application activity within the Sequence activity and rename it to “Open Application - Open Notepad”. Add an annotation: “Open application is used to open a new notepad file.”

Step 7: Click on the “Indicate element on screen” link and indicate the Notepad window.

Step 8: Click the hamburger button and select Edit Selector. In the bottom panel of the Selector Editor, rename the title of the Notepad to '* - Notepad'. Click OK to save the changes.

Step 9: In the Do section of the Open Application activity, drag and drop a Type Into activity from the Activities panel and rename it to “Type First Text”. Add an annotation: “Types the first text in the notepad file.”

Step 10: Click on the “Indicate element on screen” link of the Type Into activity and indicate the editor area of the Notepad.

Step 11: In the text box of the Type Into activity, enter the text “Welcome to UiPath”.

Step 12: Drag and drop a Click activity from the Activities panel below the Type Into activity and rename it to “Click Minimize - Simulate Click”.

Step 13: Right-click on the Click activity container and select *Annotations* from the context menu. Add an annotation: “Minimizes the notepad window using SimulateClick.”

Step 14: Click on the “Indicate element on screen” link of the Click activity and indicate the minimize button of the Notepad window.

Step 15: In the Properties panel of the Click activity, check the box of SimulateClick property and add a delay of 10000 milliseconds in the DelayBefore property.

Step 16: Drag and drop another Type Into activity below the Click activity, rename it to “Type into - Type Second Text using SendWindowMessages”. Add an annotation: “Types the second text in Notepad file.”

Step 17: Click on the “Indicate element on screen” link of the Type Into activity and indicate the editor area of the Notepad file.

Step 18: In the Properties panel of the second Type Into activity, check the boxes of the SendWindowMessages and Empty Field property.

Step 19: In the text area of the second Type Into activity, enter the text: “Let’s explore the new world of automation.”

Step 20: Insert a Click activity below the second Type Into activity, rename it to “Click – Right Click on the Notepad Editor Window”. Add an annotation: “Right-click on the notepad file”.

Step 21: Click on the “Indicate element on screen” link and indicate the Notepad editor area.

Step 22: In the properties panel, click on the dropdown menu of the MouseButton property and select “BTN_RIGHT”.

Step 23: Insert a Click activity from the Activities panel and rename it to “Click - Select 'Select All' option from the menu”.

Step 24: Click on the “Indicate element on screen” link and indicate the *Select All* option from the context menu. To indicate Select All option, press F2 to add a delay. It will help to first select the context menu manually and then indicate the Select All option.

Step 25: Insert another Click activity from the Activities panel and rename it to “Click Format Button.” Add an annotation: "Click Format button from the menu section of the notepad file."

Step 26: Click on the “Indicate element on screen” link and indicate the *Format* button.

Step 27: Insert an Attach Window activity below the Click activity and rename it to “Attach Window - Format Menu Window”. Right-click on the Attach Window activity and select *Annotations* from the context menu. Add an annotation: "Performs an action on the format menu window.”

Step 28: Click on the “Indicate element on screen” link and indicate the *Format* menu of the Notepad window.

Step 29: Inside the Do container of the Attach Window activity, insert a Click activity, and rename it to “Click Font”. Add an annotation: "Click the Font button from the menu section of Format."

Step 30: Click on the “Indicate element on screen” link and indicate the *Font* option from the Format menu of the Notepad window. To indicate the Font button, press F2 to add a delay. It will help to first select the format button manually and then indicate the Font button.

Step 31: Insert another Attach Window activity after the previous Attach Window activity and rename it to “Attach Window - Font window”. Add an annotation: “Performs Action on Font window.”

Step 32: Click on the “Indicate element on screen” link and indicate the Font window.

Step 33: Insert a Type Into activity within the second Attach Window activity and rename it to “Type into - Font type”. Add an annotation: “Types "Times New Roman" as Font Type.”

Step 34: Click on the “Indicate element on screen” link and indicate the input field of the Font names section in the Font window.

Step 35: In the text area of the Type Into activity, enter the text: “Times New Roman”. Add a delay of 2000 milliseconds using the ‘DelayAfter’ property.

Step 36: Drag and drop another Type Into activity and rename it to “Type into – Font Style”. Add an annotation: “Types ‘Italic’ as Font Style.”

Step 37: Click on the “Indicate element on screen” link and indicate the input field of the Font style section in the Font window.

Step 38: In the text box of the Type Into activity, enter the text: “Italic”. Add a delay of 2000 milliseconds using the ‘DelayAfter’ property.

Step 39: Insert a Type Into activity and rename it to “Type Into – Font Size”.

Step 40: Click on the “Indicate element on screen” link and indicate the input field of the Font Size from the Font window.

Step 41: In the text box of the Type Into activity, enter the text: “15”. Add a delay of 2000 milliseconds using the ‘DelayAfter’ property.

Step 42: Insert a Send hotkey activity below the Type Into activity and rename it to “Send Hotkey - Enter”. Add an annotation: “Sends an Enter Key to close the Font window.”

Step 43: Select *enter* from the dropdown menu of the Key option.

Step 44: Save and run the workflow.

Lesson 3 – UI Synchronization Activities

3.1 Find and Exists Activities

3.1.1 Objective

Build a workflow using Find and Exists activities which does the following:

- Opens a browser and sign in to UiPath ACME. Use Windows Credential Manager and a Find Element activity.
- Uses an Element Exists activity to confirm the login.
- Downloads 'Work Item 1' pdf file and record the action in a Notepad file.
- Resets test data and confirm reset using an Element Exist activity.
- Extracts description of two Work Items using a Get Text activity and a Text Exists activity.
- Logs Out of UiPath ACME and closes the browser.
- Logs all the key project actions in a Notepad file called 'ProjectLog.txt'.

3.1.2 Prerequisites

- ✓ Create an account in UiPath ACME website. Visit - https://acme-test.uipath.com/login.
- ✓ Store UiPath ACME username and password in Windows' Credential Manager.

3.1.3 Process Overview

- START
- Store username and password of UiPath ACME in Windows Credential Manager.
- Install the UiPath.Credentials.Activities package in UiPath Studio.
- In UiPath Studio, use a Get Secure Credential activity to store the username and password in the string variables Username and Password, respectively.
- Use an Open Browser activity to open https://acme-test.uipath.com/.
- Use a Find Element activity to identify the login form.
- Use a Type Into activity and a Type Secure Text activity to enter username and password, respectively.
- Use an Element Exists activity to identify successful login.

- Upon successful login,
    - Store the text, “Logged into UiPath ACME successfully”, in a notepad file with the current date and time.
    - Download the ‘Work Item 1’ PDF file from UiPath ACME in the local folder.
    - Store the text, “Work Item 1 downloaded into the local folder”, in the notepad file with the current date and time.
    - Reset the data of UiPath ACME and confirm successful reset using Element Exist activity.
        - Upon successful reset, store the text, “UiPath ACME data reset successful.” in the notepad file.
        - Upon unsuccessful reset, store the text, “UiPath ACME data reset failed.” in the notepad file.
    - Identify two predefined work item IDs using Text Exists activity.
    - If both IDs exist, extract their work descriptions and store them in the notepad file. Store a text, “Successfully found both WIIDs”, in the notepad file along with the current date and time.
    - If IDs do not exist, store a text, “Failed to find the predefined WIIDs”, in the notepad file along with the current date and time.
    - Use a Click activity to log-out of UiPath ACME. Store a text “System logged out”, in the notepad file along with the current date and time.
- Close browser. Store a text, “Browser closed successfully.”, in the notepad file along with the current date and time.
- STOP

3.1.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Find and Exist”.

Step 3: Drag and drop a Sequence activity from the Activities panel into the Designer panel.

Step 4: Name the Sequence activity as “Sequence – Find and Exists”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Add an annotation: “This block of code demonstrates the use of Find and Exists activities.”

Step 7: Click on the Manage Packages option in the design ribbon and Install the UiPath.Credentials.Activities package.

Step 8: Drag and drop a Get Secure Credential activity from the Activities panel and name it as “Get Secure Credential – ACME Login”. It uses the username and the password of UiPath ACME stored in Windows’ Credential Manager.

Step 9: In the Properties panel of the Get Secure Credential activity, enter the ‘Internet or Network Address’ name from Windows’ Credential Manager of the stored credentials of UiPath ACME in the Target property in double quotes.

Step 10: In the Username property of the Get Secure Credential activity, enter a new string variable named **Username**, and in the Password property, enter a new string variable named Password.

Step 11: Drag and drop an Open Browser activity below the Get Secure Credential activity and rename it to “Open Browser – ACME Website”.

Step 12: Enter the URL, “https://acme-test.uipath.com/”, in the URL text box.

Step 13: Drag and drop a Sequence activity in the Do section of the Open Browser activity and rename it to “Sequence – Login to UiPath ACME”. Add an annotation: “This sequence will log in to the UiPath ACME website.".

Step 14: Drag and drop a Find Element activity within the Sequence activity and rename it to “Find Element - Username”. Click on the “Indicate element on screen” link and indicate the Username field on the ACME login page.

Step 15: Drag and drop a Type Into activity below the Find Element activity and rename it to “Type Into – Username”. Click on the “Indicate element on screen” link and indicate the Username field on the ACME login page.

Step 16: Enter the variable **Username** in the text area of the Type Into activity. In the Properties panel, set the EmptyField property to True.

Step 17: Drag and drop a Type Secure Text activity below the Type Into activity and rename it to “Type Secure Text - Password”.

Step 18: Click on the “Indicate element on screen” link and indicate the Password field on the ACME login page.

Step 19: In the Properties panel, enter the variable **Password** in the SecureText property and set the EmptyField property to True.

Step 20: Drag and drop a Click activity below the Type Secure Text activity and rename it to “Click – Login Button”.

Step 21: Click on the “Indicate Element on screen” link and indicate the Login button on the ACME Login page.

Step 22: Insert an Element Exists activity below the Click activity and rename it to “Element Exists – Dashboard”.

Step 23: Click on the “Indicate element on screen” link and indicate the text ‘Dashboard’ on the ACME dashboard page.

Step 24: In the Output property of Element Exists activity, enter a new Boolean variable named **dashElement**.

Step 25: Drag and drop an If activity below the “Sequence – Login to UiPath ACME” activity and rename it to “If - Dashboard”. Add an annotation: “If the 'Dashboard' element is found, the process continues, else it will log the failed statement in ‘ProjectActions.txt’.”

Step 26: In the condition text box, enter the variable **dashElement**.

Step 27: In the Then section of the If activity, drag and drop a Write Text File activity and rename it to “Write Text File - Log”.

Step 28: In the “Text” box, enter the expression: **now().toString + " Logged into UiPath ACME successfully."**

Step 29: In the **Write to filename** box, enter the text: “ProjectLog.txt”.

Step 30: Drag and drop a Sequence activity below the “Write Text File - Log” activity and rename it to “Sequence – Download Work Item 1.” Add an annotation: “This sequence is to download Work Item 1.”

Step 31: Drag and drop a Click activity in the “Sequence – Download Work Item 1.” activity and rename it as “Click – User Options”.

Step 32: Click on the “Indicate Element on screen link” and indicate the “User options” button on the ACME Dashboard page.

Step 33: Insert another Click activity and rename it to “Click – Download Client and Support”.

Step 34: Click on the “Indicate element on screen” link and indicate the “Download Client and Support” button in the User Options menu.

Step 35: Drag and drop another Click activity and rename it to “Click - Work Item 1”.

Step 36: Click on the “Indicate element on screen” link and indicate the “Work Item 1” link on the “Download Client and Support” page.

Step 37: In the Properties panel of the Click activity, open the dropdown menu of the MouseButton property, and select BTN_RIGHT. It will open the right-click context menu for the link.

Step 38: Drag and drop another Click activity and rename it to “Click – Save target as..”.

Step 39: Click on the “Indicate element on screen” link and indicate the “Save target as…” option from the context menu for the link.

Step 40: Drag and Drop an Attach Window activity. Click on Indicate window on Screen link and select “Save As” window.

Step 41: Drag and drop a Click activity in the Do section of the attach Window activity and rename it as “Click – Save Button”.

Step 42: Click on the “Indicate element on screen” link and indicate the ‘Save’ button in the “Save As” window.

Step 43: Drag and drop an Append Line activity below the Click activity and rename it to “Append Line – File Downloaded”.

Step 44: In the **Text** box of the Append Line activity, enter the expression: **now().toString + " Work Item 1 downloaded into the local folder."**

Step 45: In the **Write to filename** box of the Append Line activity, enter the text: "ProjectLog.txt".

Step 46: Drag and drop a Sequence activity below the “Sequence – Download Work Item 1” activity and rename it to “Sequence – Reset Test Data”.

Step 47: Add an annotation: “This sequence will reset the test data on UiPath ACME website.”

Step 48: Drag and drop a Click activity and rename it to “Click - Home”.

Step 49: Click on the “Indicate element on screen” link and indicate the ‘Home’ button to go to the ACME Dashboard.

Step 50: Drag and drop another Click activity and rename it as “Click – User Options”.

Step 51: Click on the “Indicate element on screen” link and indicate the “User options” button on the ACME Dashboard page.

Step 52: Insert another Click activity and rename it to “Click – Reset Test Data”.

Step 53: Click on the “Indicate element on screen” link and indicate the “Reset Test Data” button from the menu of the User Options button.

Step 54: Insert another Click activity and rename it to “Click – Reset Test Data”.

Step 55: Click on the “Indicate element on screen” link and indicate the “Reset Test Data” button on the screen.

Step 56: Drag and drop an Element Exists activity and rename it to “Element Exists – Data Reset Successful.”

Step 57: Click on the “Indicate element on screen” link and indicate the text “Your Test Data has been successfully reset.”

Step 58: Create a Boolean variable named “resetSuccess” by pressing Ctrl+K in the Output property of the Element Exists activity.

Step 59: Drag and drop an If activity below the Element Exists activity and rename it to “If - resetSuccess”. Enter the variable **resetSuccess** as the condition.

Step 60: In the Then section of the If activity, drag and drop an Append Line activity and rename it to “Append Line – Data Reset Successful”.

Step 61: In the **Text** box of the Append Line activity, enter the expression: **now().toString + " UiPath ACME data reset successful."**

Step 62: In the **Write to filename** box of the Append Line activity, enter the text: "ProjectLog.txt”.

Step 63: Drag and drop a Click activity and rename it to “Click - Ok”.

Step 64: Click on the “Indicate element on screen” link and indicate the OK button on the “Message from webpage” window.

Step 65: In the Else section of the If activity, drag and drop an Append Line activity and rename it to “Append Line – Data Reset Failed”.

Step 66: In its **Text** box enter the expression: **now().toString + " UiPath ACME data reset failed."**

Step 67: **I**n the **Write to filename** box, enter the text: “ProjectLog.txt”.

Step 68: Drag and drop a Sequence activity below the “Sequence – Reset Test Data” and rename it to “Sequence – Extract WIID Description”.

Step 69: Add an annotation: “This sequence will extract the description of WIIDs.”

Step 70: Drag and drop a Click activity and rename it to “Click - Home”.

Step 71: Click on the “Indicate element on screen” link and indicate the ‘Home’ button to redirect to the ACME Dashboard.

Step 72: Drag and drop a Click activity and rename it to “Click – Work Items”.

Step 73: Click on the “Indicate element on screen” link and select the “Work Items” button on the ACME Dashboard page.

Step 74: Drag and drop a Text Exists activity and place it below the Click activity. Rename the activity to “Text Exists – First Work ID”.

Step 75: Click on the “Indicate element on screen” link and indicate the first work Id from the first row in the data table.

Step 76: Enter the same value as that of the indicated work ID in the text box of the **Text Exists** activity.

Step 77: Create a variable named **wiidFirst** by pressing Ctrl+K in the Exists property of Text Exists activity.

Step 78: Drag and drop another Text Exists activity and place it below the Click activity. Rename the activity to “Text Exists – Second Work ID”.

Step 79: Click on the “Indicate element on screen” link and select the second work Id from the second row in the data table.

Step 80: Enter the value the same as the indicated work Id in the text box of the **Text Exists** activity.

Step 81: Create a variable named **wiidSecond** by pressing Ctrl+K in the Exists property of Text Exists activity.

Step 82: Drag and drop an If activity below the Text Exists activity and rename it to “If – wiidFirst AND wiidSecond”. In the condition box, enter the expression: **wiidFirst AND wiidSecond**.

Step 83: In the Then section of the If activity, drag and drop a Get Text activity and rename it to “Get Text – First ID Description”.

Step 84: Click on the “Indicate element on screen” link and indicate the description adjacent to the first work ID.

Step 85: Create a string variable named **descFirst** by pressing Ctrl+K in the Value property of the Get Text activity.

Step 86: Drag and drop another Get Text activity and rename it to “Get Text – Second ID Description”.

Step 87: Click on the “Indicate element on screen” link and indicate the description adjacent to the second work ID.

Step 88: Create a string variable named **descSecond** by pressing Ctrl+K in the Value property of the Get Text activity.

Step 89: Drag and drop an Append Line activity and rename it to “Append Line – Description of both IDs found”.

Step 90: In the **Text** box, enter the expression: **now().toString + " Successfully found both WIIDs." + vbCr + "Desc 1: " + descFirst + vbCr + "Desc 1:" + descSecond.**

Step 91: In the **Write to filename** box, enter the text: “ProjectLog.txt”.

Step 92: In the Else section of the “If – wiidFirst AND wiidSecond” activity, drag and drop an Append Line activity and rename it to “Append Line – Failed to find WIIDs”.

Step 93: In the **Text** box, enter the expression: **now().toString + " Failed to find the predefined WIIDs."**

Step 94: In the **Write to filename** box, enter the text: “ProjectLog.txt”.

Step 95: Drag and drop a Write Text File activity below the Append Line activity and rename it to “Write Text File – Failed to find Dashboard”.

Step 96: In the **Text** box, enter the expression: **now().toString + " Dashboard UI element not found. Execution stopped."**

Step 97: In the **Write to filename** box, enter the text: “ProjectLog.txt”.

Step 98: Drag and drop a Sequence activity below the “Sequence – Extract WIID Description” and rename it to “Sequence - LogOut”.

Step 99: Drag and drop a Click activity and rename it to “Click – LogOut”.

Step 100: Click on the “Indicate element on screen” link and indicate the “Log Out” link on the ACME Dashboard.

Step 101: Drag and drop an Append Line activity and rename it to “Append Line – System Logged Out”.

Step 102: In the **Text** box, enter the expression: **now().toString + " System Logged Out"**.

Step 103: In the **Write to filename** box, enter the text: "ProjectLog.txt".

Step 104: Drag and drop a Sequence activity below the Open Browser activity and rename it to “Sequence – Close Browser”.

Step 105: Add an annotation: “This sequence will close the browser.”

Step 106: Drag and drop a Close Window activity within the “Sequence – Close Browser” activity and rename it to “Close Window - Browser”.

Step 107: Click on the “Indicate element on screen” link and indicate the browser.

Step 108: Drag and drop an Append Line activity and rename it to “Append Line – Browser Closed”.

Step 109: In the **Text** box, enter the expression: **now().toString + " Browser Closed successfully".**

Step 110: In the **Write to filename** box, enter the text: "ProjectLog.txt".

Step 111: Save and Run the workflow.

Lesson 4 – Excel, PDF, and Email Activities

4.1 Use of Excel activities to read and write

4.1.1 Objective

Build a workflow that checks for books issued to the students and impose a fine if the students have not returned it after a week of the issued date.

- Merge the data of Borrowers.csv and Data.xlsx.
- Filter the data of books issued to the students and transfer the data to an excel file named as ‘Results.xlsx’.
- Calculate the fine of $10 per day and store the calculated amount in the ‘Fine’ column of Results.xlsx.
- Fill the cell color of the “Name” column as ‘Red’ for defaulters and ‘Green’ for non-defaulters.

4.1.2 Prerequisites

- ✓ Install UiPath.Excel.Activities in UiPath Studio.
- ✓ Use Borrowers.csv and Data.xlsx files provided with this manual.

4.1.3 Process Overview

- START
- Use a Read Range activity to read the Excel files.
- Use a Join Data Table activity to join both the data tables taking “Borrower’s Name” as a median-joining constraint. Select the Join type as ‘Left’. Keep the data from Borrowers on the left of Join Data Table.
- Use a Filter Data Table activity to filter the data table using the borrower's profession, such that it only contains students. Remove the column "ID" and "Borrower's Name" from the final data table. The Results table is to be filtered.
- Write the final Data Table in the ‘Results.xlsx’ file using a Write Range activity.
- Use a For Each Row activity to:
    - Calculate the number of days difference between the Issued Date and the Returned Date.

- Use an If activity to check if the number of days difference is greater than seven.
- If the difference is greater than seven,
  - Multiply the difference with the fine amount.
  - Store the amount in ‘Results.xlsx’ using a **Write Cell** activity.
  - Use a **Set Range Color** activity to fill the cell color of the "Name" column as 'Red'.
- If the difference is smaller than seven:
  - Set the fine amount as ‘0’.
  - Store the amount in ‘Results.xlsx’ using a **Write Cell** activity.
  - Use a **Set Range Color** activity to fill the cell color of the "Name" column as 'Green'.

- STOP

4.1.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Excel Automation (Read and Write)”.

Step 3: Drag and drop a Sequence activity and name it as “Sequence – Total Fine Calculation”.

Step 4: Add an annotation as “This workflow that checks for books issued to the students and impose a fine if the students have not returned it after a week of the issue date.”

Step 5: Drag and drop a Read Range activity and name it as “Read Range – Data.xlsx”.

Step 6: Add an annotation as “Reading the Data Excel file and creating a data table called dt_LibraryData to store the information.”.

Step 7: Enter the Sheet Name as “Sheet 1” and Workbook Path as “Data.xlsx”.

Step 8: In the DataTable property of the Read Range activity, enter a new variable named as **dt_LibraryData**.

Step 9: Drag and drop a Read CSV activity and place it below the Read Range activity. Name the activity as “Read CSV – Borrowers.csv”.

Step 10: Add an annotation “Read the Students CSV file and create a data table called dt_Borrowers to store the information.”

Step 11: Enter the file path as “Borrowers.csv” and Output DataTable as “dt_Borrowers”.

Step 12: Drag and drop a Join Data Table activity below the Read CSV activity and name it as “Join DataTable – dt_LibraryData and dt_Borrowers”.

Step 13: Click on the Join Wizard option and enter the input data tables name. Enter Input Data Table 1 as “dt_Borrowers” and Input Data Table 2 as “dt_LibraryData”.

Step 14: Press Ctrl+K in the Output Data Table box of the Join wizard and create a new variable named as **dt_Result**.

Step 15: In the Column Table 1 field, enter the column name "Borrower's Name," and in the Column Table 2, enter the column name as "Borrower's Name".

Step 16: Select "=" operation between the two-column names.

Step 17: Drag and drop a Filter Data Table activity below the Join Data Table activity and name it as “Filter Data Table – dt_Filtered”.

Step 18: Click on the ‘Filter Wizard’ button and enter **dt_Result** in the Input DataTable.

Step 19: Press Ctrl+K in the Output Data Table box of the Filter Wizard and create a new variable named as **dt_Filtered**.

Step 20: Select the “Keep” radio button under the Filter Rows section and enter “Profession” in the column box, “STUDENT” in the value box, and select "=" operation between the column and its value.

Step 21: Select the Output Column section of the Filter Wizard and select the Remove radio button.

Step 22: Enter the column names, i.e., "Borrower's Name" and "ID".

Step 23: Drag and drop the Write Range activity from the workbook category and name it as “Write Range – Results.xlsx”.

Step 24: Enter the Sheet Name as “Sheet 1”, Input Data Table as **dt_Filtered**, Workbook Path as “Results.xlsx”, and Starting Cell as “A1”.

Step 25: In the Properties panel of the Write Range activity, check the box of AddHeaders property.

Step 26: Drag and drop an Assign activity under the Write Range activity and name it as “Assign - Count”.

Step 27: In the To box of the Assign activity, press Ctrl+K to create a new integer variable named as **intCount**, and in the adjacent box, enter ‘1’ as its value.

Step 28: Go to the Variables panel and change the Variable Type of **intCount** from String to Int32.

Step 29: Drag and drop a For Each Row activity below the Assign activity and name it as "For Each Row – dt_Filtered".

Step 30: In the Properties panel of the For Each Row activity, enter the input data table as **dt_Filtered**.

Step 31: Drag and drop an Assign activity in the Body section of the For Each Row activity and name it as "Assign – Increment 'intCount' by 1".

Step 32: Enter **intCount** in the To box and **intCount+1** in the Value box of the Assign activity.

Step 33: Drag and drop another Assign activity and name it as "Assign – NumberOfDays".

Step 34: Press Ctrl+K in the To section of the Assign activity to create a new variable named **intNumberOfDays**.

Step 35: Go to the Variables panel and change the Variable Type of **intNumberOfDays** from System.String to System.Double.

Step 36: In the Value box of the Assign activity, enter the expression: **(Convert.ToDateTime(row("Returned Date"))-Convert.ToDateTime(row("Issued On"))).TotalDays.**

Step 37: Drag and drop an Excel Application Scope activity and name it as "Excel Application Scope – Results.xlsx".

Step 38: Enter the workbook name as "Results.xlsx".

Step 39: Drag and drop an If activity in the Do container of Excel Application Scope activity and name it as "If – intNumberOfDays>7".

Step 40: Enter the condition as **intNumberOfDays>7** in the condition box of the If activity.

Step 41: In the Then section of If activity, drag and drop a Set Range Color activity and rename it to "Set Range Color - Red".

Step 42: In the Properties panel of Set Range Color activity, enter the Input Color as **Color.Red**, Input Range as **"C"+ intCount.ToString,** and Input Sheet Name as "Sheet 1".

Step 43: Drag and drop a Write Cell activity below the Set Range color activity and name it as "Write Cell – Fine Amount".

Step 44: In the Properties panel of the Write Cell activity, enter the Destination Range as **"G"+ intCount.ToString,** Destination Sheet Name as “Sheet 1,” and Input Value as **Convert.ToString(Convert.ToInt32(intNumberOfDays)*10)**.

Step 45: In the Else section of the If activity, drag and drop a Set Range Color activity and name it as “Set Range Color - Green”.

Step 46: In the Properties panel of Set Range Color activity, enter the Input Color as **Color.Green**, Input Range as **"C"+ intCount.ToString,** and Input Sheet Name as “Sheet 1”.

Step 47: Drag and drop a Write Cell activity below the Set Range color activity and name it as “Write Cell – Fine Amount”.

Step 48: In the Properties panel of the Write Cell activity, enter the Destination Range as **"G"+ intCount.ToString,** Destination Sheet Name as “Sheet 1”, and Input Value as **“0”.**

Step 49: Save and Run the workflow.

4.2 Advanced Activities of Excel

4.2.1 Objective

Build a workflow that uses the data of an Excel file to create a Pivot Table and refreshes the Pivot Table after deleting a specific range from the pre-existing data.

- Open file Example.xlsx using an Excel Application Scope activity.
- Create a Pivot Table in a new sheet using the data in Example.xlsx.
- Select and delete a specific range from the data in the Excel file using a Delete Range activity.
- Refresh the pivot table using the Refresh Pivot Table activity.

4.2.2 Prerequisites

- ✓ Install UiPath.Excel.Activities in UiPath Studio.
- ✓ Use the example excel file provided with this manual named Example.xlsx.

4.2.3 Process Overview

- Use an Excel Application Scope activity to open the Example.xlsx file.
- Use a Create Pivot Table activity, and enter the Table Name, Sheet Name, and Source Table Name in the Properties panel.
- Use a Message Box activity to notify the user to check the pivot table in the excel file.
- Use a Delete Range activity to delete a specific range from the Excel sheet. Enter the Range and Sheet Name in the Properties panel.
- Use a Refresh Pivot Table activity to refresh the Pivot Table. Enter the Sheet name and Table name in the Properties panel of the Refresh Pivot Table activity.
- STOP

4.2.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Drag and drop a Sequence activity and name it as "Sequence – Advanced Activities of Excel".

Step 3: Add an annotation as “This workflow uses the data of an Excel file to **Create a Pivot Table** and **Refreshes the pivot table** after deleting a specific range of pre-existing data.”

Step 4: Drag and drop an Excel Application Scope activity and place it in the Sequence activity. Name this activity as "Excel Application Scope - example.xlsx."

Step 5: Enter the filename “example.xlsx” in the ‘workbook path’ field of the Excel Application Scope activity.

Step 6: Drag and drop a Create Pivot Table activity in the Do section of the Excel Application Scope activity and rename it to “Create Pivot Table - Pivot”.

Step 7: In the Properties panel of the Create Pivot Table activity, enter the following details:

Range	“A1”
Table Name	“pivot”
Sheet Name	“PivotTable”
Source Table Name	“table”

Step 8: Drag and drop a Message Box activity and name it as “Message Box – Notify User”. In the text area, enter the text: "Check the pivot table in the Excel file."

Step 9: Drag and drop a Delete Range activity and name it as “Delete Range – A6:G16”.

Step 10: In the Properties panel of the Delete Range activity, enter the following details:

Range	“A6:G16”
SheetName	“SalesOrders”

Step 11: Drag and drop a Refresh Pivot Table activity and name it as “Refresh Pivot Table - pivot”.

Step 12: In the Properties panel of the Refresh Pivot Table activity, enter the following details:

SheetName	“PivotTable”
TableName	“pivot”

Step 13: Save and Run the workflow.

4.3 PDF Extraction

4.3.1 Objective

Build a workflow that reads and extracts a scanned PDF using a Read PDF with OCR activity and store the data in an excel file.

- Read the provided scanned PDF invoice.
- Extract Company Name, Invoice Number, Invoice Date, and Total Amount from the PDF.
- Use string manipulation techniques such as the Substring and Split string method to separate extracted texts.
- Store the extracted data in an excel file.

4.3.2 Prerequisites

- ✓ Use scanned PDF invoice provided with this manual named as 'sample1.pdf'.
- ✓ Install UiPath.PDF.Activities in UiPath Studio.

4.3.3 Process Overview

- START
- Use a Read PDF with OCR activity to read the scanned PDF.
- Use the Microsoft OCR engine to read the PDF.
- Use an Assign activity along with the String Manipulation techniques to separate Company Name, Invoice Number, Invoice Date, and Total Amount from the extracted text.
- Use a Build Data Table activity to create a data table and specify the field names as column names.
- Use an Add Data Row activity to store the data in the data table.
- Use an Excel Application Scope activity to open 'Result.xlsx'.
- Use a Write Range activity to write the Data Table in the excel file.
- STOP

4.3.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “PDF Extraction”.

Step 3: Drag and drop a Sequence activity into the designer’s panel and name it as “Sequence – PDF Extraction”.

Step 4: Add an annotation: "This workflow reads a scanned PDF using a 'Read PDF with OCR' activity to extract Company name, Invoice Number, Invoice Date, and Total Amount. The extracted details are stored in an Excel File”.

Step 5: Drag and drop a Read PDF with OCR activity and rename it to “Read PDF with OCR – sample1.pdf”.

Step 6: In the Properties panel of the Read PDF with OCR activity, enter the filename as “sample1.pdf”.

Step 7: Drag and drop a Microsoft OCR activity into the Read PDF with OCR activity.

Step 8: Rename the Microsoft OCR activity as “Microsoft OCR – sample1.pdf”.

Step 9: In the Properties panel of the Microsoft OCR activity, enter the value for Scale as ‘3’.

Step 10: Select the Profile as ‘OCRProfile.Scan’ for the specified image to achieve a better OCR read. The Scan preprocessing file is used as it is suitable for scanned files.

Step 11: In the Properties panel of the Microsoft OCR activity, press ‘Ctrl + K’ in the Text section and name the string variable as **Extracted Text**.

Step 12: Drag and drop an Assign activity below the Read PDF with OCR activity.

Step 13: Name the Assign activity as “Assign - Company Name”.

Step 14: Add an annotation as “Extracting ‘Company name’ using String Manipulation techniques from the variable ‘**ExtractedText**’”.

Step 15: In the To box of the Assign activity, create a new string variable by pressing Ctrl+K and name the variable as **CompanyName**.

Step 16: In the Value box of the Assign activity enter the expression: **ExtractedText.Substring(0,14)**.

Step 17: Drag and drop another Assign activity and name it as “Assign – Invoice Number”.

Step 18: Add an annotation as “Extracting ‘Invoice Number’ using String Manipulation techniques from the variable ‘**ExtractedText**’”.

Step 19: In the To box of the Assign activity, create a new string variable by pressing Ctrl+K and name the variable as **InvoiceNumber**.

Step 20: In the Value box of the Assign activity, enter the expression: **ExtractedText.Substring(ExtractedText.IndexOf("#"),7)**.

Step 21: Drag and drop another Assign activity and name it as “Assign – Invoice Date”.

Step 22: Add an annotation as “Extracting Invoice Date using String Manipulation techniques from the variable ‘ExtractedText’”.

Step 23: In the To box of the Assign activity, create a new variable by pressing Ctrl+K and name the variable as **InvoiceDate**.

Step 24: Go to the Variables panel and change the Variable Type of **InvoiceDate** from ‘String’ to ‘DateTime'.

Step 25: In the Value box of the Assign activity, enter the expression: **Convert.ToDateTime(ExtractedText.Substring(ExtractedText.IndexOf("#")+7,ExtractedText.IndexOf("$")-(ExtractedText.IndexOf("#")+7)))**.

Step 26: Drag and drop another Assign activity and name it as “Assign – Total Amount”.

Step 27: Add an annotation as “Extracting Total Amount using String Manipulation techniques from the variable ‘ExtractedText’”.

Step 28: In the To box of the Assign activity, create a new string variable by pressing Ctrl+K and name the variable as **Total Amount**.

Step 29: In the Value box of the Assign activity, enter the expression: **ExtractedText.Substring(ExtractedText.IndexOf("$"),ExtractedText.IndexOf("Quantity")-ExtractedText.IndexOf("$"))**.

Step 30: Drag and drop a Build Data Table activity and name it as “Build Data Table – dt_ExtractedData”.

Step 31: Click on the DataTable button and create four columns with the column names as "Company Name", "Invoice Number", "Invoice Date", and "Total Amount".

Step 32: Select the DataType of "Company Name", "Invoice Number", and "Total Amount" as System.String, and "Invoice Date" as System.DateTime.

Step 33: In the Properties panel of the Build Data Table activity, press Ctrl+K in the DataTable field and create a new data table variable named as **dt_ExtractedData**.

Step 34: Drag and drop an Add Data Row activity below the Build Data Table activity and name it as “Add Data Row – Extracted Fields”.

Step 35: In the ArrayRow property of the Add Data Row activity's Properties panel, enter the expression:
{CompanyName,InvoiceNumber,InvoiceDate,TotalAmount}

Step 36: In the DataTable property, enter the variable **dt_ExtractedData**.

Step 37: Drag and drop an Excel Application Scope activity below the Add Data Row activity and name it as "Excel Application Scope – Result.xlsx".

Step 38: In the Workbook Path box, enter the file name as "Result.xlsx".

Step 39: Drag and drop a Write Range activity in the Do section of the Excel Application Scope activity and name it as "Write Range – dt_ExtractedData in Result.xlsx".

Step 40: In the Properties panel of the Write Range activity, enter the following details:

SheetName	"Sheet1"
StartingCell	"A1"
DataTable	dt_ExtractedData

Step 41: In the Properties panel of the Write Range activity, check the box of the AddHeaders property.

Step 42: Save and Run the workflow.

4.4 Email Automation

4.4.1 Objective

Build a workflow that accesses the email account of the user and does the following:

- Read emails using a Get IMAP Mail Messages activity.
- Use a Read Range activity to read the Rules.xlsx file.
- Check the sender's address using a For Each activity.
- If the string is present in the sender's address specified in the Sender column, then move the mail to the email folder specified in the Folder column of Rules.xlsx file using a Move IMAP Mail Message activity.

4.4.2 Prerequisites

- ✓ Send 4-5 dummy emails to the email ID that you will use for practice. Sender's address must contain the string specified in the Rules.xlsx.
- ✓ Folder names specified in the 'Rules.xlsx' should be available for the email account we are working with.

4.4.3 Process Overview

- START
- Use a Get IMAP Mail Messages activity to read the unread emails in the Inbox folder.
- Use a Read Range activity to read the 'Rule.xlsx' file.
- Use a For Each activity to iterate through each email.
- Use a Log Message activity on an Info level to print Mail Subject in the output panel.
- Use a For Each Row activity to iterate through each row of 'Rule.xlsx' file.
- Use an If activity to check whether the string is present in the sender's address specified in the Sender column of 'Rule.xlsx' file.
- If the sender's address contains the string specified in the Sender column of Rule.xlsx file, then use a Move IMAP Mail Message activity to move the mail to the email folder specified in the Folder column of the 'Rule.xlsx' file.
- STOP

4.4.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Email Automation”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – Email Automation”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter annotation as “This block of code reads the Inbox and sorts the emails by moving them to various folders, based on “rules” defined in Rules.xlsx”.

Step 7: Create a variable through the Variables panel as under:

Name	Variable Type	Scope	Default
mailMessages	List<MailMessage>	Sequence - Email Automation	

Step 8: Drag and drop a Get IMAP Mail Messages activity. Name it as "Get IMAP Mail Messages from Email" and add an annotation as “Receive mails from the emails and store in mailMessages.”

Step 9: In the Properties panel of the Get IMAP Mail Messages activity, enter the values as under:

Mail Folder	“Inbox”
Port	993
Server	“imap.gmail.com”
Email	Enter your email address
Password	Enter your password
Messages	mailMessages

Step 10: Drag and drop a Read Range Activity from the Workbook category and rename it to “Read Range – Rules.xlsx”.

Step 11: Create a variable through the Variables panel as under:

Name	Variable Type	Scope	Default

dt_mailRules	DataTable	Sequence - Email Automation	

Step 12: In the Properties panel of the Read Range activity, enter the values as under:

Range	“”
SheetName	“Sheet1”
WorkbookPath	“Rules.xlsx”
DataTable	dt_mailRules

Step 13: Insert a For Each activity and enter the *VB expression* as **mailMessages**. Set the Type argument of For Each activity to "System.Net.Mail.MailMessage".

Step 14: Rename the For Each activity to “For Each Mail Item”, and add an annotation as “Iterate through each email item in mailMessages.”

Step 15: In the Body section of the For Each activity, insert a Log Message Activity and rename it to “Log Message – Mail Subject”.

Step 16: Select the Log Level as Info and type message **mail.Subject** in the Message text area.

Step 17: Drag and drop a For Each Row activity below the Log Message activity and rename it to “For Each Row in Rules.xlsx”. Add an annotation as “Iterate through each row in dt_mailRules”.

Step 18: Enter the VB expression as **dt_mailRules**.

Step 19: In the Body section of the For Each Row activity, insert an If activity and enter the condition **mail.From.Address.Contains(row("Sender").toString).** Rename it to “If contains Sender Rule”, and add annotation as “Identify mails that contain the string specified in the Sender column of Rules.xlsx”.

Step 20: In the Then section of the If activity, insert a Move IMAP Mail Message activity and rename it to “Move IMAP Mail Messages to Specified Folder”, and add an annotation as “Move the Mail Message to the specified folder in the Folder column of Rules.xlsx”.

Step 21: In the first text box of the Move IMAP Mail Message activity, enter **mail**, and in the second text box, enter **row(“Folder”).ToString**.

Step 22: In the Properties panel of the Move IMAP Mail Messages activity, enter the values as under:

Port	993
Server	“imap.gmail.com”
FromFolder	“Inbox”
MailMessage	Mail
Email	Enter your Email Address
Password	Enter your password

Step 23: Save and Run the workflow.

Lesson 5 – Structured Data Extraction Wizard

5.1 Extraction Wizards

5.1.1 Objective

Build a workflow using Data Scraping wizard that scrapes whitepaper's details from UiPath website.

- Open UiPath Whitepaper webpage in the browser - https://www.uipath.com/resources/automation-whitepapers.
- Extract titles, URLs and descriptions of all the available whitepapers in the first page.
- Store the scraped data in a CSV file.

5.1.2 Process Overview

- START
- Use an Open Browser activity to open UiPath Whitepaper webpage.
- Use the URL – https://www.uipath.com/resources/automation-whitepapers.
- Use the Data Scraping tool from the Design ribbon.
- Extract the Whitepaper titles, URLs and description of each whitepaper.
- Use the Extract Co-related data option to extract multiple fields.
- Use a Write CSV activity to store the extracted data in 'ScrapedData.csv' file.
- STOP

5.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Data Extraction".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'Demonstrating the use of Data Scraping Wizard".

Step 5: Right click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Add an annotation: “This block of code uses Data Scraping wizard to scrape details of whitepapers from UiPath website.”

Step 7: Insert an Open Browser activity and name it as “Open Browser - UiPath Whitepaper Webpage.” Add an annotation as “Open UiPath Whitepaper Webpage”.

Step 8: Enter URL – "https://www.uipath.com/resources/automation-whitepapers".

Step 9: Save and run the workflow to ensure that the URL opens successfully.

Step 10: Click the Data Scraping button in the Design ribbon.

Step 11: In the Extract Wizard window, click Next and select the title of the first whitepaper.

Step 12: Again, click Next and select the title of the second whitepaper.

Step 13: In the Configure Columns section, replace “Column1” with “Whitepaper Title”. Check the Extract URL box. Replace “Column 2” with “URL”.

Step 14: Click Next.

Step 15: From below the Preview panel, select the Extract Co-related Data and select the description of the first whitepaper.

Step 16: Click Next again and select the description of the second whitepaper.

Step 17: In the Text Column Name text box, enter the text: “Description”.

Step 18: Click Next and then click “Finish” in the preview window.

Step 19: In the Indicate Next Link window, click “No”.

Step 20: Insert a Write CSV activity. Rename it to “Write CSV – Store Data in a CSV File”. Add an annotation: “Write in a new CSV file in the background”.

Step 21: In the “Write to what file:” text box, enter “ScrapedData.csv”.

Step 22: In the “Write From:” text box, enter the variable in which the extracted text is stored.

Step 23: Save and run the workflow.

Lesson 7 – Utilizing External Code

7.1 Power Shell

7.1.1 Objective

Build a workflow using Invoke Power Shell activity that creates two text files in the project folder and move both the files to a new folder.

- Create two text files in the project folder.
- Open both the files to show their contents and close the files.
- Inform the user that a folder will be created to move both the files.
- Create a folder and inform the user that the files will be moved here.
- Move both the text files within the folder.
- Inform the user about the successful execution.

7.1.2 Process Overview

- START
- Store a PowerShell script in a text file in the project folder that creates two text files and stores some text in them.
- Use a Read Text File activity to read the text file.
- Use an Invoke PowerShell activity to run the script.
- Use a Start Process activity to open and show the content of the created text files.
- Use a Delay activity to allow the user some time to read the content of the files.
- Use a Kill Process activity to kill the Notepad process.
- Use a Message Box activity to inform the user that a folder will be created in the project folder where both the text files will be moved.
- Store a PowerShell script in a text file in the project folder that creates a folder in the current directory.
- Use an Invoke PowerShell activity to run the second script.
- Use a Message Box activity to inform the user that the folder is created, and the files will be moved.
- Store a PowerShell script in a text file in the project folder that moves files from the current directory to the new folder.

- Use an Invoke PowerShell activity to run the third script.
- Use a Message Box activity to inform the user that the files moved successfully.
- STOP

7.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Power Shell”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel. Rename it to “Demonstrating the use of Power Shell”.

Step 4: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 5: Add an annotation: “This block of code uses Invoke Power Shell activity to create two files and move them into a folder.”

Step 6: Open a Notepad file and insert the following lines of code:

```
1  New-Item -Path . -Name "testFile1.txt" -ItemType "file" -
   Value "This is a text in the FIRST file."
2  New-Item -Path . -Name "testFile2.txt" -ItemType "file" -
   Value "This is a text in the SECOND file."
```

Step 7: Create a folder named Scripts in the project folder. Save the Notepad file as CreateTwoFiles.txt within the Scripts folder.

Step 8: Insert a Read Text File activity in the Sequence activity. Rename it to “Read Text File - CreateTwoFiles.txt”.

Step 9: Click on the folder icon of the FileName text box and select CreateTwoFiles.txt from the Scripts folder.

Step 10: In the Properties panel, press Ctrl + K and enter a string variable named **createTwoFiles**.

Step 11: Insert an Invoke Power Shell activity below the Read Text File activity. Rename it to “Invoke Power Shell - Create Two Files”. In the Properties panel, enter the values as shown below:

Properties	**Value**
CommandText	createTwoFiles

IsScript	Check the box.
TypeArgument	System.IO.FileInfo

Step 12: Insert a Start Process activity below the Invoke Power Shell activity. Rename it to “Start Process – testFile1.txt”. In the FileName property, enter the name of the first notepad file to be created upon the workflow execution. Here, enter “testFile1.txt”.

Step 13: Insert a Delay activity below the Start Process activity. Rename it to “Delay – 3 Seconds for testFile1.txt”. Set the duration as 00:00:03.

Step 14: Insert another Start Process activity below the Delay activity. Rename it to “Start Process – testFile2.txt”. Enter “testFile2.txt” in the FileName property.

Step 15: Insert another Delay activity below the second Start Process activity. Rename it to “Delay – 3 Seconds for testFile2.txt”. Set the duration as 00:00:03.

Step 16: Insert a Kill Process activity below the Delay activity. Rename it to “Kill Process - Notepad”. In the ProcessName property, enter the text “Notepad”. It will close all the open Notepad files in the system.

Step 17: Insert a Message Box activity below the Kill Process activity. Rename it to “Message Box – Creating Foder”. In the text box, enter the text: “A folder called TwoFiles will be created to store both the text files.”

Step 18: Open a new Notepad file and enter the following code:

```
1  New-Item -Path . -Name "TwoFiles" -ItemType "directory"
```

Step 19: Save the Notepad file as CreateFolder.txt in the Scripts folder.

Step 20: Insert a Read Text File activity below the Message Box activity. Rename it to “Read Text File – CreateFolder.txt”. Click on the folder icon of the FileName text box and select the CreateFolder.txt file from the Scripts folder. In the Content property, press **Ctrl + K** and enter a new variable called **createFolder**.

Step 21: Insert an Invoke Power Shell activity below the Read Text File activity. Rename it to “Invoke Power Shell - Create Folder”. In the Properties panel, enter the values as shown in the table below:

Properties	**Value**

CommandText	createFolder
IsScript	Check the box.
TypeArgument	System.IO.DirectoryInfo

Step 22: Insert a Message Box activity below the Invoke Power Shell activity. Rename it to "Message Box – Moving Files". In the text box, enter the text: "Folder created. Both files will be moved.".

Step 23: Open a new Notepad file and enter the following code:

```
Move-Item -Path "testFile1.txt" -Destination TwoFiles
Move-Item -Path "testFile2.txt" -Destination TwoFiles
```

Step 24: Save the Notepad file as MoveFiles.txt in the Scripts folder.

Step 25: Insert a Read Text File activity below the Message Box activity. Rename it to "Read Text File – MoveFiles.txt".

Step 26: Click on the folder icon of the FileName property and select the MoveFiles.txt from the Scripts folder. In the Content property, press Ctrl + K and enter a new variable named **moveFiles**.

Step 27: Insert an Invoke Power Shell activity below the Read Text File activity. Rename it to "Invoke Power Shell – Move Files". In the Properties panel, enter the values as shown in the table below:

Properties	**Value**
CommandText	moveFiles
IsScript	Check the box.
TypeArgument	System.IO.DirectoryInfo

Step 28: Insert a Message Box below the Invoke Power Shell activity. Rename it to "Message Box – Files Moved". In the text box, enter the text: "Files moved successfully."

Step 29: Save and run the workflow.

7.2 Python

7.2.1 Objective

Build a workflow using Python activities that checks if the number is a prime number.

- Ask the user to input a number.
- Check if the number is a prime number.
- Inform the user through a message in a message box that whether the number is a prime number.

7.2.2 Prerequisites

- ✓ Install UiPath.Python.Activities in UiPath Studio.
- ✓ Install Python 3.6 in the system.

7.2.3 Process Overview

- START
- Use an Input Dialog activity to ask the user to input a number.
- Store the Python script to check prime numbers in the project folder.
- Use a Load Python Script activity within a Python Scope activity to load the script file.
- Use an Invoke Python Method activity to call methods from the script.
- Use a Get Python Script activity to convert the output from Python.object variable to .NET datatype.
- Use a Message Box activity to display whether the number is a prime number.
- STOP

7.2.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Python”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – Demonstrating the use of Python activities”.

Step 5: Right click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Add an annotation: “This block of code uses Python activities to check if a number is a prime number.”

Step 7: Insert an Input Dialog activity in the Sequence activity. Rename it to “Input Dialog – Ask Number”.

Step 8: In the Label text box, enter the text: “Check Prime”. In the Title text box, enter the text “Enter a number to check if it is a prime:”.

Step 9: In the Properties panel, press Ctrl + K in the Result property, and insert an integer variable named **intCheckPrime**.

Step 10: Insert a Python Scope activity below the Input Dialog activity. Rename it to “Python Scope – Check Prime Number”. Add an annotation: “Run Python activities and script within this container to check for prime number.”

Step 11: In the Properties panel of the Python Scope activity, enter the values as shown in the table below:

Properties	Value
Path	Insert the path of Python.exe application file. Example: "C:\Users\UserName\AppData\Local\Programs\Python\Python36"
Target	Select x86 or x64 depending on the Python version installed on your system.
Version	Select Python version as installed in your system. You can select Auto to allow UiPath choose by itself.

Step 12: Open a Notepad file and insert the following lines of code:

```
1  def primeTest (num):
2        if num > 1:
3              for i in range (2,num):
4                    if (num % i) ==0:
5                          isPrime = "{0} is not a prime
   number.".format(num)
6                          Break
```

```
7               else:
8                     isPrime = "{0} is a prime
    number.".format(num)
9       else:

10              isPrime = "{0} is not a prime
    number.".format(num)
11      return isPrime
```

Step 13: Save the Notepad file as CheckPrime.py.

Step 14: Insert a Load Python Script activity in the Do section of the Python Scope activity. Rename it to “Load Python Script – CheckPrime.py”.

Step 15: In the File text box, enter “CheckPrime.py”. In the Result property with the Properties panel, press Ctrl + K and enter a new variable named **loadCheckPrime**.

Step 16: Insert an Invoke Python Method activity below the Load Python Script activity. Rename it to “Invoke Python Method – primeTest”. In its Properties panel, enter the values as shown in the table below (The data type of **invokePrimeTest** variable is ‘PythonObject’):

Properties	**Value**
Input parameters	{intCheckPrime}
Instance	loadCheckPrime
Name	“primeTest”
Result	invokePrimeTest

Step 17: Insert a Get Python Object activity below the Invoke Python Method activity. Rename it to “Get Python Object – invokePrimeTest”. In its Properties panel, enter invokePrimeTest in the Python Object property. Select Object in the TypeArgument property. In the Result property, press Ctrl + K and enter a new variable named **finalResult**.

Step 18: Insert a Message Box activity below the Get Python Object activity and rename it to “Message Box – Display Final Result”.

Step 19: Save and run the workflow.