Program 1: Create a Sequence to Obtain User Inputs and Display in a Message Box

Objective: To create a simple sequence in UiPath to capture user input and display it using a message box.

Topics Covered in This Program

- User Input Handling
- Message Box
- String Variables
- Concatenation and Formatting

Procedure:

Step 1: Create a New UiPath Project

1. Open UiPath Studio.
2. Click "Process" → Give it a name (e.g., UserInput_Display).
3. Choose the default folder and click "Create".

Step 2: Add an Input Dialog Activity

1. In the Activities Panel, search for Input Dialog.
2. Drag and drop the Input Dialog activity into the Main.xaml workflow.
3. In the Properties Panel, set:

   Title: "User Input"

   Label: "Enter your name:"

   Result: Create a new variable (e.g., userName, Type: String).

Step 3: Add a Message Box Activity

1. Search for Message Box in the Activities Panel.
2. Drag and drop it below the Input Dialog activity.
3. In the Properties Panel, set:

   Message: "Hello " + userName + "! Welcome to UiPath RPA!".

Step 4: Run the Automation

1. Click the "Run" button (F5) in UiPath Studio.
2. A dialog box appears, prompting for the user's name.
3. After entering a name, a message box appears displaying "Hello <userName>!

Welcome to UiPath RPA!".

Result:-

The sequence is created in UiPath studio with input dialog box and message box and it is successfully compiled and output is viewed.

Program 2: Create a Flowchart to Navigate to a Desired Page Based on a Condition

Objective: To create a Flowchart in UiPath that navigates to different web pages based on user input conditions.

Topics Covered in This Program

- Flowchart-based Automation
- User Input Handling
- Conditional Navigation using Flow Decision
- Web Automation using Open Browser

Procedure:

Step 1: Create a New UiPath Project

1. Open UiPath Studio.
2. Click "Process" → Give it a name (e.g., Flowchart_Navigation).
3. Choose the default folder and click "Create".

Step 2: Add a Flowchart Activity

1. In the Activities Panel, search for Flowchart.
2. Drag and drop the Flowchart activity into the Main.xaml workflow.

Step 3: Add an Input Dialog to Get User Choice

1. Search for Input Dialog in the Activities Panel.
2. Drag and drop it inside the Flowchart.
3. In the Properties Panel, set:

    Title: "Website Navigation"

Label: "Enter a website choice (Google/Facebook):"

Result: Create a new variable (userChoice, Type: String).

Step 4: Add a Flow Decision

1. Search for Flow Decision in the Activities Panel.
2. Drag and drop it inside the Flowchart, connecting it to the Input Dialog.
3. In the Condition Field, enter:

   userChoice.ToLower = "google"
4. This checks if the user entered "Google".

Step 5: Add Open Browser Activities

1. For Google Navigation:
   - Search for Open Browser in the Activities Panel.
   - Drag and drop it in the "True" branch of the Flow Decision.
   - In the Properties Panel, set:
   - URL: "https://www.google.com"
   - BrowserType: Chrome (or any preferred browser).
2. For Facebook Navigation:
   - Drag another Open Browser activity into the "False" branch.
   - In the Properties Panel, set:
   - URL: "https://www.facebook.com"

   BrowserType: Chrome.

Step 6: Connect the Components

The Flowchart should now look like this:

Input Dialog → Flow Decision → (Google → Open Browser: Google) / (Facebook → Open Browser: Facebook).

Step 7: Run the Automation

1. Click "Run" (F5).
2. A dialog box appears prompting the user to enter "Google" or "Facebook".
3. Based on input:
   - If "Google", it opens Google.
   - If "Facebook", it opens Facebook.

Expected Output

User enters "Google" → Google.com opens.

User enters "Facebook" → Facebook.com opens.

Result:-

The flowchart to navigate to a desired location based on a condition and display them using a message box in the Ui Path studio environment is successfully designed , executed and output is verified.

Program 3: Create a State Machine Workflow to Compare User Input with a Random Number

Objective:

To create a State Machine workflow in UiPath that generates a random number, takes user input, and provides guidance based on the guessed number using trigger conditions.

Topics Covered in This Program

- State Machine-based Automation
- User Input Handling
- Random Number Generation
- Conditional Logic and State Transitions

Procedure

Step 1: Create a New State Machine Project

1. Open UiPath Studio.
2. Click "Process" → Give it a name (e.g., StateMachine_GuessNumber).
3. Choose the default folder and click "Create".
4. In the Activities Panel, search for State Machine.

5. Drag and drop the State Machine activity into the Main.xaml workflow.

Step 2: Design the State Machine with Four States

1. Initialization State (Start State) – Generates a random number and takes user input.
2. Comparison State – Compares the user input with the generated number.
3. Try Again State – Redirects the user to input another number based on hints.
4. End State – Displays the success message when the user guesses correctly.

Step 3: Create Variables

1. Create a variable to store the randomly generated number.
2. Create a variable to store user input.

Step 4: Configure the Initialization State

1. Add an Assign activity to generate a random number.
2. Add an Input Dialog activity to prompt the user for input.
3. Connect this state to the Comparison State.

Step 5: Configure the Comparison State

1. Add a Flow Decision activity to check if the user's input matches the random number.
2. If the input matches, transition to the End State.
3. If the input does not match, transition to the Try Again State.

Step 6: Configure Try Again State (T2 and T3 Triggers)

1. Add two Trigger Activities inside the Try Again State:
   Trigger 1 (T2 - Try for Small Number)
   - Rename as "Try for Small".
   - Set condition: If the guessed number is greater than the random number.
   - Display a message: "Your guess is too high! Try a smaller number.".
   - Transition back to Initialization State for a new attempt.
2. Trigger 2 (T3 - Try for Bigger Number)

    - Rename as "Try for Bigger".
    - Set condition: If the guessed number is less than the random number.
    - Display a message: "Your guess is too low! Try a bigger number.".
    - Transition back to Initialization State for a new attempt.

Step 7: Configure the End State

1. Add a Message Box activity to display success if the user guesses correctly.
2. Transition from Comparison State to End State when the guess is correct.

Step 8: Connect the Components

- Connect Initialization State → Comparison State.
- Connect Comparison State → End State (if correct).
- Connect Comparison State → Try Again State (if incorrect).
- Connect Try Again State back to Initialization State for another attempt.

Step 9: Run the Automation

1. Click "Run" in UiPath Studio.
2. A dialog box appears, prompting the user to enter a number.
3. The program compares the user input with the random number.
4. Based on the input:
    - If correct, a success message appears.
    - If too high, a hint appears to enter a smaller number.
    - If too low, a hint appears to enter a bigger number.
5. The process repeats until the correct number is guessed.

Expected Output

- User enters a higher number → Prompt: "Your guess is too high! Try a smaller number."
- User enters a lower number → Prompt: "Your guess is too low! Try a bigger number."
- User guesses the correct number → "Congratulations! You guessed the correct number."

Result:-

A State Machine workflow to compare user input with a random number in the Ui Path studio environment is created and output is verified successfully.

Program 4: Build a Process in the RPA Platform Using UI Automation Activities

Objective:

To build an RPA process in UiPath to open a web browser, search for "Bangalore Temperature", extract the displayed temperature, and show the result in a message box.

Topics Covered in This Program

- UI Automation in RPA
- Web Automation using Open Browser
- Interacting with Web Elements (Typing, Clicking, and Extracting Data)
- Using Selectors for Automation
- Error Handling in Automation

Procedure

Step 1: Create a New Sequence

1. Open UiPath Studio.
2. Click "New Process" → Enter a name (e.g., Bangalore_Temperature_Automation).
3. Choose Sequence as the workflow type.

Step 2: Add UI Automation Activities to the Sequence

(a) Open Browser Activity

- Drag and drop the Open Browser activity.
- Set it to open Google (https://www.google.com).

(b) Attach Browser Activity

- Drag and drop Attach Browser after Open Browser.
- Ensures all actions take place in the same browser window.

(c) Type Into Activity (Enter Search Query)

- Drag and drop Type Into inside Attach Browser.
- Indicate the Google Search bar and configure it to type "Bangalore Temperature".
- Add a Delay Activity (1-2 seconds) for page loading.

(d) Click Activity (Click Search Button)

- Drag and drop Click inside Attach Browser.
- Indicate the Google Search button and configure the activity to click it.

(e) Extract Temperature Data (Get Text Activity)

- Drag and drop Get Text after Click Activity.
- Indicate the temperature value displayed on the search results page.
- Store the extracted temperature in a variable.

(f) Message Box Activity (Display Result)

- Drag and drop Message Box after Get Text.
- Configure it to display the extracted temperature value.

Step 3: Configure Each Activity

- Open Browser: Set URL to Google.
- Type Into: Input text as "Bangalore Temperature".
- Click: Select the Google Search button.
- Get Text: Extract the temperature value.

- Message Box: Display the fetched temperature.

Step 4: Run the Process

- Click Run in UiPath Studio. The automation will:
- Open Google in a web browser
- Enter "Bangalore Temperature" in the search bar.
- Click Search
- Extract the temperature data from search results.
- Display the temperature in a message box.

Expected Output

- The browser opens Google.
- The search bar is filled with "Bangalore Temperature".
- The search button is clicked, and results appear.
- The current temperature of Bangalore is extracted and displayed.

Result:

An RPA process using UI Automation in UiPath Studio was successfully created to open a web browser, search for "Bangalore Temperature", extract the displayed temperature, and present the result in a message box. The automation workflow executed successfully, verifying accurate data retrieval and display.

Program 5: Create an Automation Process Using Key System Activities, Variables, and Arguments

Objective:

To create an automation process in UiPath Studio that calculates the area of a rectangle using values entered by the user. The process will be divided into two workflows: one for collecting input and invoking another workflow, and the other for calculating the area. The result will be displayed using a Message Box. This version does not include any conditional checks.

Topics Covered in This Program

- Modular design using Invoke Workflow File
- Using Variables and Arguments
- Input Dialog for user interaction
- Assign activity for performing calculation
- Message Box for displaying output
- Argument mapping between workflows

Procedure

Step 1: Create the Main Workflow

1. Open UiPath Studio.
2. Create a new project and name it appropriately (e.g., Rectangle_Area_Using_Invoke).
3. In the main sequence, add two Input Dialog activities to get the length and width of the rectangle from the user.
4. Store the values in variables (e.g., lengthVar and widthVar).

Step 2: Create a Secondary Workflow for Calculation

1. Add a new Sequence to the project and name it (e.g., CalculateArea.xaml).
2. In this workflow, create the required arguments:
    - Two input arguments for length and width.
    - One output argument to return the calculated area.

3. Use Assign activity to multiply the length and width and store the result in the output argument.

Step 3: Invoke the Calculation Workflow

1. In the main sequence, after the input activities, use the Invoke Workflow File activity to call CalculateArea.xaml.
2. Use the Import Arguments option to map:
    - The user input variables to the input arguments of the calculation workflow.
    - A new variable (e.g., areaResult) to the output argument to capture the calculated area.

Step 4: Display the Result

1. After the Invoke Workflow activity, add a Message Box activity.
2. Display the calculated area stored in the output variable.

Step 5: Run the Process

1. Save and run the project.
2. The workflow will prompt the user to enter length and width, calculate the area in a separate file, and then display the result in a message box.

Expected Output

- When the user enters valid numeric inputs for length and width, the process will calculate the area and show the result in a message box.
- There are no conditions or validations—only calculation and display.

Result:

An RPA process using **Invoke Workflow File** and **arguments** was successfully created in UiPath Studio to calculate the area of a rectangle. The solution demonstrates modular design, argument passing between workflows, and the use of Input Dialog and Message Box for simple user interaction.

Program 6: Implement Automation Using System Trigger in UiPath

Objective:

To create and implement RPA processes in UiPath Studio that are triggered automatically based on predefined system events such as the start of a specific application (e.g., Notepad) or a change in a specific file. This process uses System Triggers to demonstrate unattended, event-driven automation.

Topics Covered in This Program

- Understanding System Triggers in UiPath
- Creating and Publishing Automation Workflows
- Event-Based Automation Using UiPath Assistant
- Configuring Process Start Trigger (e.g., Notepad)
- Configuring File Change Trigger
- Monitoring Automation Execution Logs
- Managing System Triggers through UiPath Assistant

Procedure

Step 1: Create a New Project in UiPath Studio

- Launch UiPath Studio.
- Create a new process and name it appropriately (e.g.,System_Trigger_Notepad_FileChange).
- Design the automation process using a Sequence.
- Add a simple activity (e.g., Message Box, Log Message) to confirm the process was triggered.

Step 2: Define the Automation Task

- In the main workflow, add an activity to execute when the trigger is fired.
    - For Notepad trigger: Add a Message Box saying *"Notepad has started"*.

    - For File change trigger: Add a Log Message or Message Box indicating *"File was modified"* or *"File was created"*.
- Keep the process lightweight for quick execution upon event detection.

Step 3: Publish the Project

- Click “Publish” in UiPath Studio.
- Publish the process to UiPath Assistant or Orchestrator, depending on your environment.
- Verify that the process appears in the Assistant/Orchestrator dashboard.

Step 4: Configure the System Triggers

Option A: Process Start Trigger (Example: Notepad)

- Open UiPath Assistant.
- Go to the “Triggers” tab and click “Create”.
- Select the published process from the list.
- Choose Trigger Type as Process Start.
- Set the condition as Application = Notepad (notepad.exe).
- Save the trigger configuration.

Option B: File Change Trigger

- In UiPath Assistant, create another trigger.
- Choose Trigger Type as File Change.
- Specify the file path to monitor (e.g., C:\Users\YourName\Documents\sample.txt).
- Choose the change type: Created, Changed, or Deleted.
- Save the trigger.

Step 5: Test the Automation

🔹 **To test Notepad Trigger:**

- Close Notepad if open.
- Launch Notepad manually.

- Observe if UiPath triggers the automation and displays the expected message.

🔹 **To test File Change Trigger:**

- Go to the monitored file path.
- Perform the action configured in the trigger (create, modify, or delete the file).
- Confirm the automation is triggered automatically and displays/logs the expected message.

Step 6: Monitor and Manage Automation

- Check the execution status in UiPath Assistant under Jobs/Execution Logs.
- Ensure that the job status is Successful.
- Use UiPath Orchestrator or Assistant to disable, edit, or delete the trigger if needed.

Expected Output

Trigger Type	Triggered By	Output
Process Start	Opening Notepad	Message Box: "Notepad has started"
File Change	File created/modified/deleted	Message Box: "File was modified" (based on action)

Result:

Two RPA processes were successfully created and published using different System Trigger types in UiPath:

1. Process Start Trigger to respond to Notepad launch, and
2. File Change Trigger to detect file modifications.

The automation workflows ran successfully without manual execution, validating UiPath's event-based automation capabilities.

Program 7: Design an automate Login Process to a Web Email Account

Aim:

To automate the login process to a web-based login account using an RPA tool (UiPath) with enhanced security and browser automation practices.

Procedure:

Step 1: Identify the Target Website

URL: Test Login | Practice Test Automation
Open the URL manually in your browser to verify accessibility and UI layout.

Step 2: Launch UiPath Studio

Open UiPath Studio and create a new project:

- Type: Process
- Name: EmailLoginAutomation

Step 3: Use UI Application/Browser Activity

Instead of using the traditional "Open Browser" activity, use "Use Application/Browser" to launch and interact with the Gmail login page.

- Drag "Use Application/Browser"
- Set URL to Test Login | Practice Test Automation

Step 4: Assign Username and Secure Password

Add the following activities to handle login credentials securely:

Assign activity:
userName = "student"
Get Password: Use Get Secure Credential or input password manually using an Input Dialog (for test/demo purposes).

Assign Secure Password:
securePassword = new System.Net.NetworkCredential(String.Empty, password).SecurePassword

Step 5: Type into Email and Password

Use Type Into activities to navigate through the input fields.

- Use Type Into to input the userName.
- Use Type Secure Text to enter the securePassword.

Make sure to validate selectors for:

- Email/username input field
- Password input field

Step 6: Click Submit and Delay

1. Use Click activity to click the Next or Submit button.
2. Add a short Delay activity (e.g., 3–5 seconds) to wait for page transition.

Step 7: Click Logout After Successful Login

1. Use Element Exists or Check App State to detect successful login (e.g., inbox or profile icon).
2. If login is successful, use Click activity to log out.

Step 9: Run and Debug

- Run the workflow and monitor browser behavior.

- Use Log Message for tracking execution.

Expected Output:

Test case 1: Positive LogIn test

1. Verify new page URL contains practicetestautomation.com/logged-in-successfully/
2. Verify new page contains expected text ('Congratulations' or 'successfully logged in')
3. Verify button Logout is displayed on the new page

Test case 2: Negative username test

1. Verify error message is displayed
2. Verify error message text is Your username is invalid!

Test case 3: Negative password test

1. Verify error message is displayed
2. Verify the error message text is Your password is invalid!

Result:

This automation demonstrates a secure and robust login process using UiPath's UI Application/Browser, secure credential handling, and reliable click/input sequences. It ensures proper synchronization, error handling, and data protection.

Program 8: Design and Build a Process for Recording Mouse and Keyboard Actions

Aim: To use UiPath to automate the Windows Calculator by recording mouse clicks and keyboard inputs to perform the operation

Procedure:

Step 1: Launch UiPath Studio

- Open UiPath Studio
- Create a new process named: MouseKeyboardAutomation
- Click Create

Step 2: Choose the Target Application

Use a simple desktop application:

- Example: Notepad or Calculator
- Manually open the application (Win + R → notepad or calc → Enter)

Step 3: Start Basic Recording

- In UiPath, go to the Design tab
- Select Recording → Basic Recording
- The recording panel will appear

Step 4: Perform the Actions to be Recorded

While the recorder is active, perform the following steps:

If using Calculator:

1. Click 2, then 5
2. Click +
3. Click 7, then 5
4. Click =

Step 5: Save and Exit Recording

- Click Save & Exit on the recording panel
- UiPath will auto-generate a sequence with:
    - Click activities (mouse)
    - Type Into activities (keyboard)
    - Send Hotkey activities (keyboard shortcuts)

Step 6: Review and Arrange Activities

- Review the generated workflow
- Optionally insert Delay activities if required for timing
- Optionally use Attach Window to scope actions inside the target app

Step 7: Run the Workflow

- Press Run
- Observe that the workflow mimics all recorded actions step-by-step

Result: A basic recorder in UiPath can effectively capture and replicate both mouse and keyboard actions. This enables automation of repetitive tasks in desktop applications without writing code manually.

Program 9: Design and Build an Automation for Scraping Data from Websites and Writing to Excel using UiPath

Objective:

To create a web automation in UiPath Studio that logs into a secure website (https://acme-test.uipath.com), navigates to the "Work Items" page, extracts tabular data using the Table Extraction feature, and writes the extracted data to an Excel file. This program demonstrates data extraction from authenticated web pages and structured output generation.

Topics Covered in This Program

- Working with secured websites (Login automation)
- Using Table Extraction in UiPath
- DataTable variable handling
- Writing extracted data to Excel using Excel Process Scope
- Automation testing and output validation

Procedure:

Step 1: Create a New Project in UiPath Studio

- Launch UiPath Studio.
- Create a new Process.
- Name the project: WebDataExtraction_Acme.

Step 2: Open the Target Website and Log In

- Use **Open Browser** activity.
    - Input URL: https://acme-test.uipath.com
- Inside the browser container, add **Type Into** and **Click** activities to:
    - Enter Email: your_email@domain.com
    - Enter Password: your_password
    - Click the **Login** button

Step 3: Navigate to the Work Items Page

- After login, use **Click** activity to navigate to the **Work Items** section of the website.
- Ensure the selector points to the correct navigation element.

Step 4: Extract Data from the Table

- Use **Table Extraction Wizard**:
    - Go to **Design** → Click **Table Extraction**
    - Select the entire table (work items table) by choosing the first row and column
    - Enable **pagination** if the table spans across multiple pages
    - Output: Save the result into a variable called ExtractDataTable
        - Variable Type: DataTable

Step 5: Write Data to Excel File

- Use **Excel Process Scope** activity:
    - Inside the scope, use **Use Excel File**
        - File Path: "C:\Users\YourUsername\Documents\Acme_WorkItems.xlsx"
    - Add **Write Range** activity:
        - Input: ExtractDataTable
        - Sheet Name: "WorkItems"
        - Ensure "Add Headers" is checked

Step 6: Configure Properties and Selectors

- Ensure proper selectors for login and table extraction.
- Set delay or retry scope where needed for page load.
- Use the Output **Panel** and **Log Message** to debug.

Step 7: Test and Debug

- Run the full workflow.
- Check if:
    - Login is successful

- Work Items page loads
- Table is extracted correctly
- Excel file is generated with proper data and structure

Expected Output:

- The automation logs in to https://acme-test.uipath.com
- Successfully navigates to the “Work Items” page
- Extracts the entire tabular data including multiple pages
- Writes the extracted data into Acme_WorkItems.xlsx with accurate formatting and headers

Result:

The automation process demonstrates how to securely log in to a dynamic website, extract structured data from a web table using UiPath’s Table Extraction feature, and export it to a clean Excel format. This serves as a foundation for automating reporting or data integration tasks in real-world enterprise scenarios.

Ex. No.: 10– Implementing Robust Error Handling in an RPA Workflow

Aim:

To implement error handling in a UiPath workflow by automating a web interaction and gracefully managing exceptions using Try-Catch blocks, variable assignments, and logging.

Procedure:

Step 1: Set Up the Environment

- Open UiPath Studio.
- Create a new process and name it **ErrorHandlingBrowserDemo**.
- Add a new **Sequence** and name it **BrowserLoginWorkflow**.

Step 2: Automate the Browser Interaction

- Within the sequence, drag an **Open Browser** activity and set the URL to: https://practicetestautomation.com/practice-test-login/.
- Inside the browser scope, add a **Type Into** activity to enter the text “student” into the username input field.
- Use **Indicate on Screen** to select the username field.
- Then **Edit Selector**:
    - Modify the selector to include a specific tag or attribute. For example, adjust the element's tag from input to INPUT or update the name or id attribute.
- Save this sequence.

Step 3: Implement Try-Catch in Main Workflow

- In **Main.xaml**, drag a **Try-Catch** activity into the sequence.
- In the **Try** section:
    - Use **Invoke Workflow File** and select BrowserLoginWorkflow.xaml.

Step 4: Add Exception Handling

- In the **Catch** block, choose exception type: System.Exception.
- Inside the Catch section, add the following **Assign activities**:
    - **Assign Activity:**
        - **To:** systemexception
        - **Value:** exception
    - **Assign Activity:**
        - **To:** exceptionsource
        - **Value:** systemexception.Source
    - **Assign Activity:**
        - **To:** exceptionmessage
        - **Value:** systemexception.Message
- Create these variables:
    - systemexception → Data Type: System.Exception
    - exceptionsource → Data Type: String
    - exceptionmessage → Data Type: String

Step 5: Log or Display the Error

- Use **Write Line** activity to display:
    - "Source: " + exceptionsource
    - "Message: " + exceptionmessage
- Alternatively, use a **Log Message** or **Message Box** for visibility.

Step 6: Test the Workflow

- To simulate an error, modify the selector in BrowserLoginWorkflow.xaml to an incorrect value.
- Run the **Main.xaml** workflow.
- Observe whether:
    - The error is caught.
    - The error details (source and message) are logged.

Result:

The RPA workflow was able to handle runtime errors effectively using a Try-Catch block. Exception details like source and message were assigned and logged using Assign activities, proving the workflow's robustness and reliability during failure scenarios.