



# **NAAN MUTHALVAN PROJECT**

## **ROBOTIC PROCESS AUTOMATION (RPA)**

### **AUTOMATED MAIL INVITATION SYSTEM FOR SYMPOSIUMS AND OTHER EVENTS**



**RVS COLLEGE OF ENGINEERING**

College code: 9215

Department of Computer Science and Engineering

**ANNA UNIVERSITY::CHENNAI – 600 025**

**DECEMBER 2024**

**A PROJECT REPORT**

**SUBMITTED BY**

**PRAKASH N**

NM-ID(32-char): CF8886EB81887C09D584DED16B74DDE6

NM ID - aut921522CS35

Roll.no – 921522104026

## ABSTRACT

This document details the development of an automated system for sending personalized email invitations using UiPath Studio, leveraging data from Google Sheets and an SMTP server (e.g., Gmail). The workflow is designed to minimize manual effort, enhance accuracy, and provide real-time updates for tracking invitations.

The process begins with fetching participant data from Google Sheets and loading an email template containing placeholders. Using UiPath, the template is dynamically personalized with recipient-specific details, such as names and email addresses, and the customized emails are sent via SMTP. The system updates the Google Sheets with a "EmailSent" status for each successfully delivered email, ensuring efficient tracking.

Robust error handling, logging, and optimization mechanisms are incorporated to handle potential issues such as invalid data or failed email delivery. The modular workflow is scalable, capable of processing large datasets, and ensures ease of deployment for future updates or additional functionalities like RSVP tracking.

Aligned with initiatives like **Naan Mudhalvan**, this project equips students with practical skills in automation, showcasing the potential of UiPath Studio in addressing real-world challenges. This system offers a reliable, scalable, and efficient solution for managing email communications, making it ideal for academic and professional event workflows.

<b>TITLE NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1.</b>	<b>Introduction</b>	<b>1</b>
<b>2.</b>	<b>Aim</b>	<b>1</b>
<b>3.</b>	<b>Background</b>	<b>1</b>
<b>4.</b>	<b>Problem Statement</b>	<b>2</b>
<b>5.</b>	<b>Solution</b>	<b>2</b>
<b>6.</b>	<b>Benefits</b>	<b>3</b>
<b>7.</b>	<b>Pre-requisites and Tools</b>	<b>3</b>
<b>8.</b>	<b>Project Overview</b>	<b>4</b>
<b>9.</b>	<b>Overview of the Original Project</b>	<b>8</b>
<b>10.</b>	<b>My Improvements and Customizations</b>	<b>8</b>
<b>11.</b>	<b>Improvised Project Overview</b>	<b>9</b>
<b>12.</b>	<b>Module Breakdown</b>	<b>15</b>
<b>12.1.</b>	Module 1: Google Form Creation and Google Sheets Setup	15
<b>12.2.</b>	Module 2: Project Setup and Initial Variables	18

<b>12.3.</b>	Module 3: Connecting to Google Sheets and Reading Responses	21
<b>12.4.</b>	Module 4: Reading the Email Template and Preparing for Customization	24
<b>12.5.</b>	Module 5: Customizing the Email Template for Each Recipient	26
<b>12.6.</b>	Module 6: Sending Emails with SMTP	28
<b>12.7.</b>	Module 7: Updating the Google Sheet After Sending Emails	31
<b>12.8.</b>	Module 8: Final Testing and Deployment	35
<b>13.</b>	<b>Conclusion</b>	39
<b>14.</b>	<b>Diploma of Completion</b>	39

## 1. Introduction

In the modern era of digital communication, managing large-scale email invitations for events can be a time-consuming and error-prone process. Personalizing each email, ensuring accurate delivery, and keeping track of invitation statuses often requires significant manual effort, especially for academic and professional events. Recognizing these challenges, this project leverages **UiPath Studio**, a powerful Robotic Process Automation (RPA) tool, to automate the process of sending personalized email invitations.

The project utilizes **Google Forms** to collect participant details such as names and email addresses, which are then stored in **Google Sheets**. UiPath Studio fetches this data, processes it to customize email templates, and sends the emails through an SMTP server (e.g., Gmail). Additionally, the workflow updates the Google Sheet in real-time to mark recipients who have been invited, providing a seamless and efficient way to track progress.

This project is structured across key modules, including data fetching, email template management, dynamic personalization, email automation, and status tracking. It incorporates error handling and logging to ensure reliability and robustness. The system is designed to handle large datasets, making it scalable and adaptable for various use cases.

By automating repetitive tasks and reducing errors, this system significantly improves efficiency and accuracy in managing event communications. This project represents a practical and innovative solution for streamlining event invitation workflows, showcasing the transformative potential of automation in everyday tasks.

---

## 2. Aim

The aim of the Automated Mail Invitation System is to streamline the process of sending personalized email invitations for symposiums and other events. The system will automatically read a mail template from a document, extract recipient details from an Excel file, customize the invitations with personalized content, and send them out via email. This solution will significantly reduce manual effort, minimize human error, and ensure efficient communication with a large number of recipients.

---

## 3. Background

In academic and professional event settings, organizing symposiums involves sending out invitations to participants. Traditionally, this is a manual task where event organizers must input recipient details, personalize emails, and send them one by one. With hundreds or even thousands of invitations to send, this process can become extremely time-consuming, error-prone, and inefficient.

Manual email invites often lead to mistakes such as:

- **Incorrect recipient names** or titles.
- **Missing recipients** due to manual oversight.

- **Delayed invitations** due to the repetitive nature of the process.

In addition, keeping track of RSVPs, confirming guest attendance, and updating details in email communication can become overwhelming. This situation calls for an automated system that can:

- Handle bulk personalized invitations.
  - Ensure correct and timely distribution.
  - Easily manage the invitation list and respond to RSVPs.
- 

## 4. Problem Statement

Sending invitations for academic and professional events, like symposiums, is often a manual and time-consuming task. The process includes:

- **Manual Template Handling:** Editing templates with placeholders (e.g., <RecipientName>) for each recipient.
  - **Recipient Data Management:** Managing recipient data in Excel, which can lead to errors when manually entered into email systems.
  - **Personalization Challenges:** Customizing emails with personalized information (e.g., recipient names, event details) can be cumbersome.
  - **Email Distribution Issues:** Sending emails individually or using outdated systems increases the risk of mistakes and delays.
- 

## 5. Solution

To address these challenges, we propose an **Automated Mail Invitation System** using UiPath. This system will streamline the entire process:

1. **Automated Template Reading:** The system will automatically load the email template from a Word document, replacing placeholders with personalized data.
  2. **Recipient Data Processing:** Recipient details (names, emails) are extracted from an Excel file for each recipient.
  3. **Personalized Content Generation:** The system will replace placeholders in the template (like <RecipientName>) with actual recipient data.
  4. **Automated Email Sending:** Personalized emails will be sent automatically using an email service (e.g., Outlook or SMTP), ensuring no errors or delays.
  5. **Error Reduction & Efficiency:** This solution minimizes human errors and accelerates the process of sending personalized invitations, even for large recipient lists.
-

## 6. Benefits

- **Time-Saving:** Automates manual tasks, saving significant time in managing invitations.
- **Increased Accuracy:** Reduces the risk of errors in addressing and handling recipient details.
- **Scalability:** Handles large numbers of recipients with ease, eliminating the need for additional manual effort.

---

## 7. Pre-requisites and Tools

### 1. Google Sheets:

- A Google Sheets account with API enabled.

### 2. UiPath Studio:

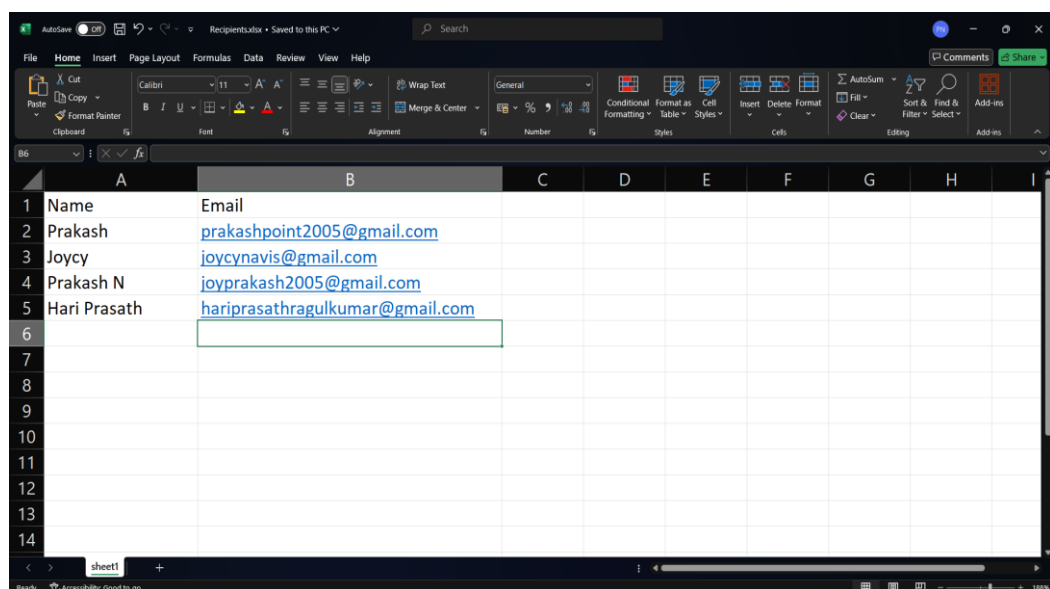
- UiPath Studio installed with **UiPath GSuite** and **Send SMTP Mail Message** activities.

### 3. SMTP Credentials:

- Email account (e.g., Gmail) and SMTP settings:
  - Server: smtp.gmail.com, Port: 587, SSL/TLS enabled.
  - Use Gmail credentials or an **App Password**.

### 4. Google Sheets Template:

- Columns: Name, Email, and Invited.

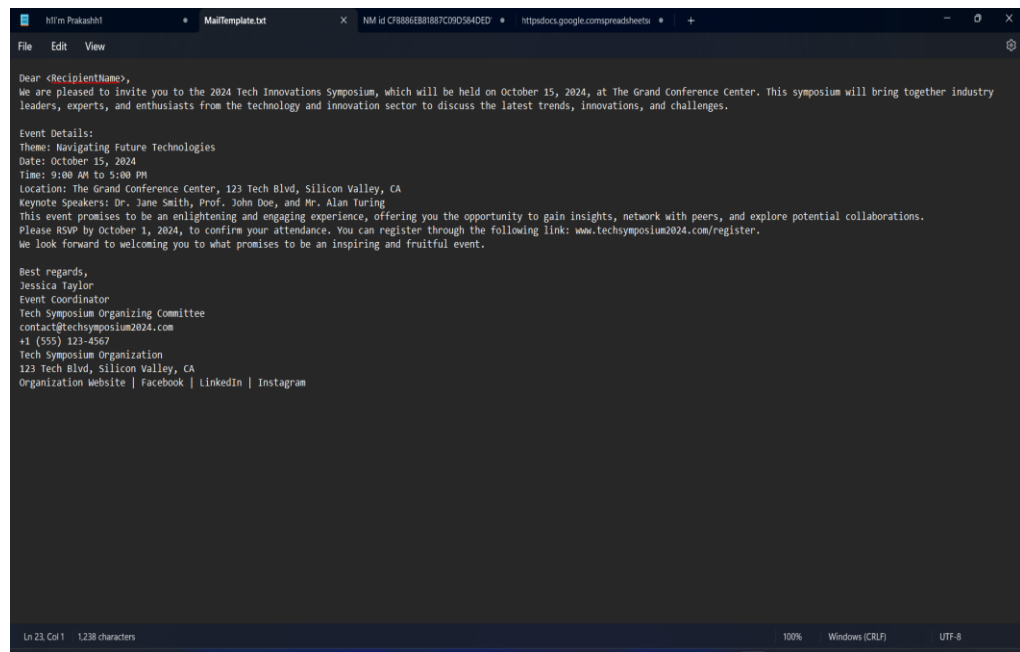


The screenshot shows a Google Sheets spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I
1	Name	Email							
2	Prakash	<a href="mailto:prakashpoint2005@gmail.com">prakashpoint2005@gmail.com</a>							
3	Joycy	<a href="mailto:joycynavis@gmail.com">joycynavis@gmail.com</a>							
4	Prakash N	<a href="mailto:joyprakash2005@gmail.com">joyprakash2005@gmail.com</a>							
5	Hari Prasath	<a href="mailto:hariprasathragulkumar@gmail.com">hariprasathragulkumar@gmail.com</a>							
6									
7									
8									
9									
10									
11									
12									
13									
14									

## 5. Email Body Text File:

- A .txt file containing the template for the email body (e.g., email\_template.txt), where placeholders (e.g., <RecipientName>) will be replaced with actual recipient data.
- This file should be stored in a known location and read into a string variable (e.g., templateText) using the **Read Text File** activity in UiPath.

A screenshot of a text editor window titled 'MailTemplate.txt'. The editor contains an email template with placeholders like <RecipientName>. The text includes an invitation to a 2024 Tech Innovations Symposium, event details (theme, date, time, location, speakers), and contact information for the organizing committee. The status bar at the bottom indicates 'Ln 23, Col 1' and '1,238 characters'.

## 8. Project Overview

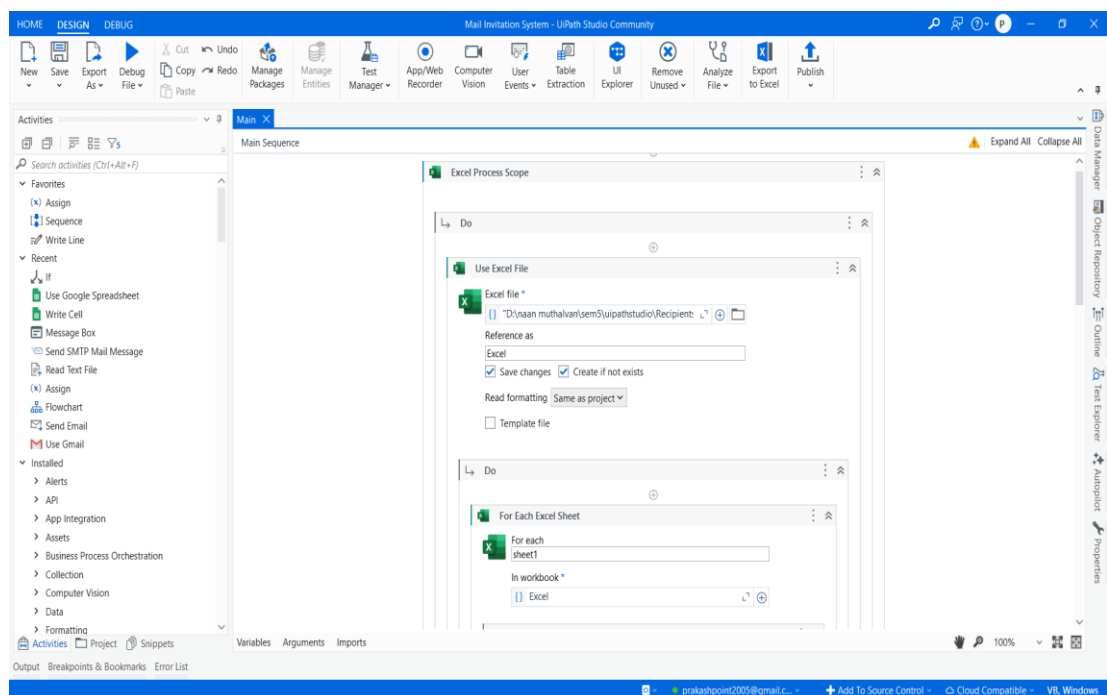
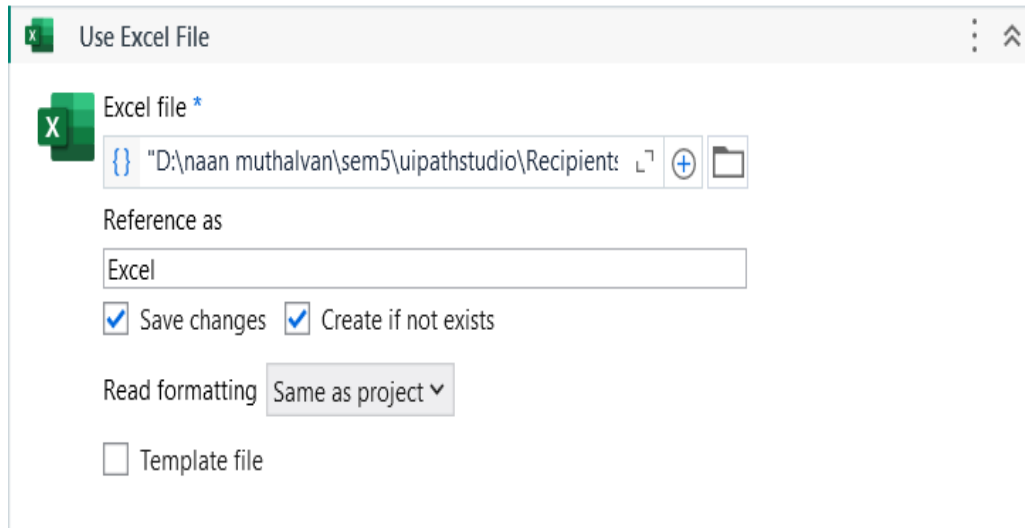
This UiPath workflow is designed to automate the process of sending personalized emails to a list of recipients. The steps include loading recipient data from an local Excel file, reading an email template, replacing placeholders in the template with recipient-specific data, and then sending the emails with an attachment.

Steps Breakdown:

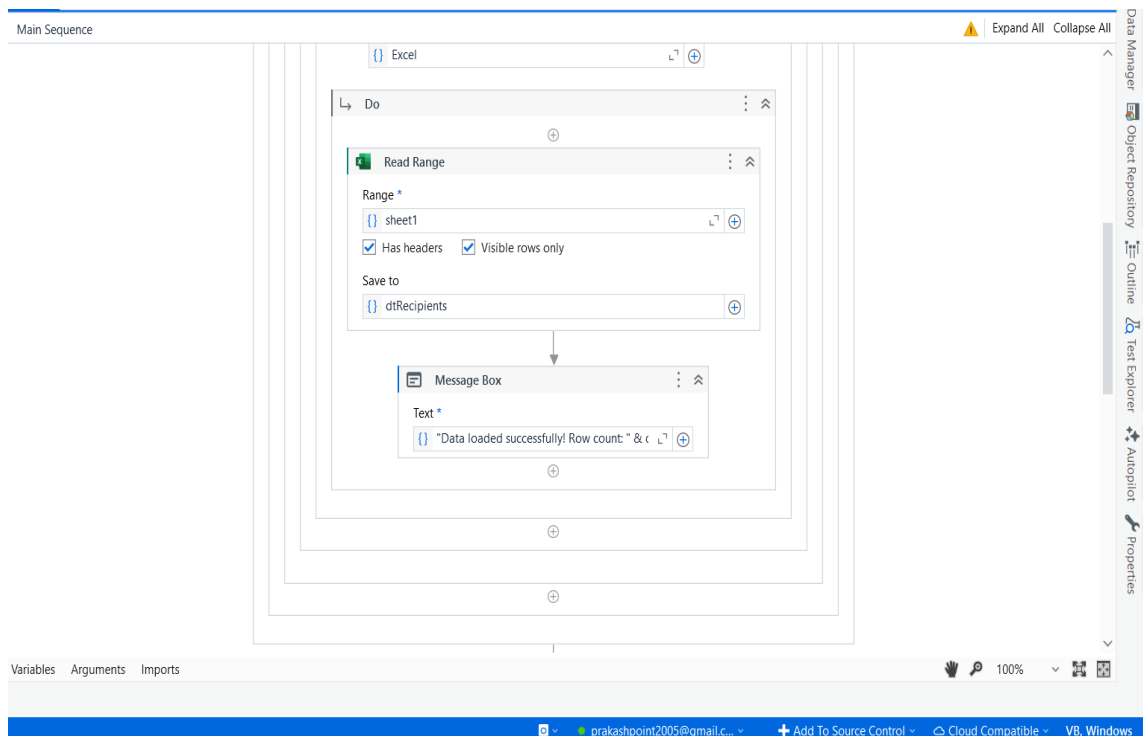
### 1. Reading Recipient Data (Excel)

- Use Excel File Activity: Opens the Excel file Recipients.xlsx from the specified directory. If the file doesn't exist, it will be created. It is not kept open after processing.



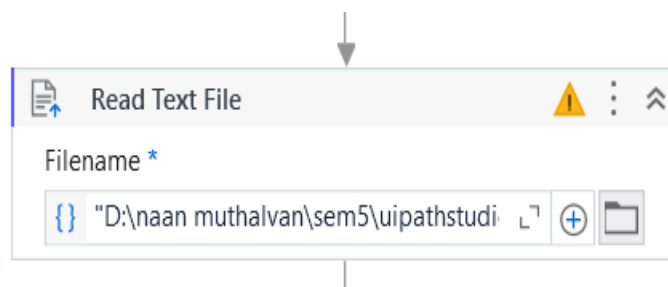


- For Each Excel Sheet Activity: Iterates over each sheet in the Excel file.
  - Read Range Activity: Reads data from the sheet (specifically "sheet1") into a DataTable (dtRecipients). The data includes recipient information like names and email addresses.
  - Message Box: Displays a message indicating that the data has been successfully loaded, including the row count.



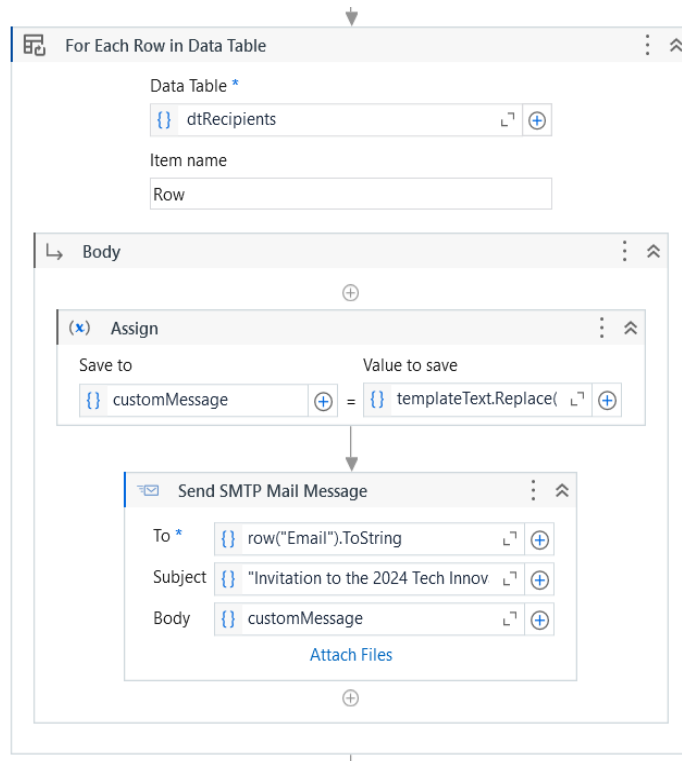
## 2. Reading the Email Template (Text File)

- Read Text File Activity: Reads the content from a text file (MailTemplate.txt), which contains the email body template. The contents are stored in the templateText variable.

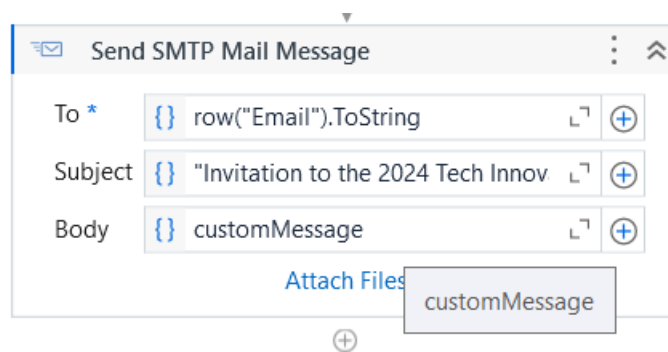


## 3. Sending Emails to Recipients

- For Each Row in Data Table Activity: Loops through each row in the dtRecipients DataTable (which holds recipient data).
  - Assign Activity: Replaces the placeholder <RecipientName> in the email template with the recipient's name (from the current row).

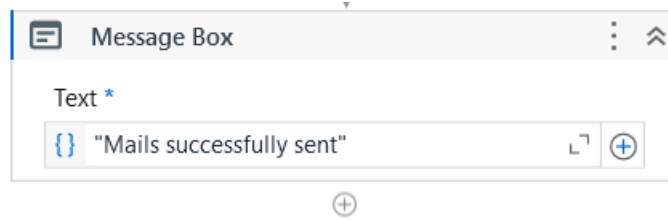


- **Send SMTP Mail Message Activity:** Sends an email to each recipient:
  - **From and To:** The sender is a fixed email address (prakashpoint2005@gmail.com), and the recipient's email address is taken from the Email column of the current row.
  - **Subject and Body:** The email subject is static, and the body contains the customized message (customMessage), which was created by replacing the placeholder in the template.
  - **Attachment:** An image (uipath\_logo.jpg) is attached to the email.
  - The email is sent using Gmail's SMTP server with secure connection.



#### 4. Completion Message

- **Message Box Activity:** Displays a message saying "Mails successfully sent" once all emails have been processed.



Summary:

This workflow automates the process of:

- Loading recipient data from an Excel file.
- Customizing an email template for each recipient.
- Sending personalized emails with attachments via SMTP.

---

## 9. Overview of the Original Project

The original project aimed to automate the process of sending personalized email invitations by:

- **Reading recipient data from an Excel file.**
- **Customizing an email template** using placeholders (like <RecipientName>).
- **Sending the email with an attachment** via SMTP, specifically using Gmail's SMTP server.

The project provided a simple, efficient way of sending customized invitations, but it had limitations in scalability, flexibility, and error handling, especially when dealing with large datasets or multiple event configurations.

---

## 10. My Improvements and Customizations

### A. Data Input Flexibility

- **Multiple Data Sources:** I extended the system to handle **dynamic Excel file paths** and **multiple sheets** for different recipient groups, making it adaptable for various events.

### B. Email Template Customization

- **Additional Placeholders:** I added more customizable placeholders like <EventDate>, <Location>, and <RSVPLink>, allowing for more personalized content based on recipient data.

### C. Improved Email Sending

- **Support for Multiple SMTP Providers:** The system now supports different SMTP servers (e.g., Gmail, Outlook), enabling users to select their preferred email provider.
- **Batch Processing:** I optimized the email sending process for efficiency, especially when dealing with large recipient lists.

### D. Attachment Handling

- **Multiple Attachments:** The system now supports sending multiple attachments, such as event brochures or PDF files, based on the recipient's data.

### E. Enhanced Error Handling

- **Robust Error Handling and Logging:** I implemented advanced error handling and logging to ensure smooth operation, even when errors occur during the process.

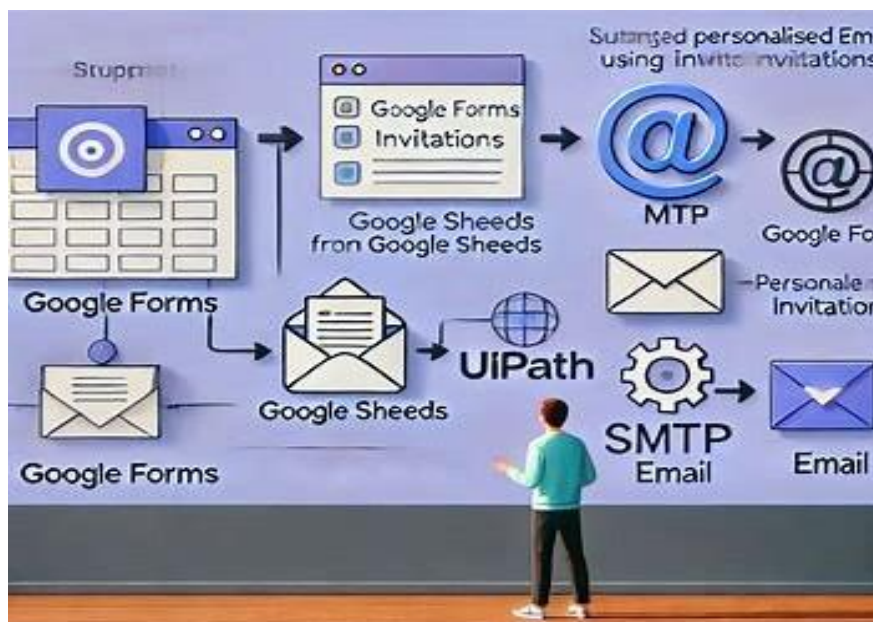
### F. Usability Enhancements

- **Interactive User Interface:** The system now prompts users to input dynamic configurations (e.g., file paths, SMTP settings), making it more user-friendly.
- **Scalability Improvements:** The workflow is optimized for handling large data sets without performance issues, ensuring efficiency even with hundreds or thousands of recipients.

---

## 11. Improvised Project Overview

This automation project is built to read responses from a Google Form, send personalized email invitations to respondents, and update the Google Sheets response sheet to mark them as "Invited." We'll be working in UiPath Studio, a tool used for creating automation workflows.



Symposium Registration Form

QuestionsResponses0Settings

Symposium Registration Form

B

I

U

↺

↻

Please fill in your details to register for the upcoming symposium.

Name\*

Short-answer text

Email\*

Short-answer text

Form Responses 1

FileEditViewInsertFormatDataToolsExtensionsHelp

100%123Default...10BIBIA

H6	A	B	C	D	E	F	G
	Form_Responses1						
1	Timestamp	Name	Email	Invited			
2	10/11/2024 18:29:30	Prakash	prakashpoint2005@gmail.com	Invited			
3	11/11/2024 17:54:43	Joycy	joycynavis@gmail.com	Invited			
4	11/11/2024 20:24:04	PRAKASH	joyprakash2005@gmail.com				
5	11/11/2024 22:03:44	Mary	joycynavis14@gmail.com				
6	12/11/2024 09:36:03	S. Aruna Devi	a01276504@gmail.com				
7	12/11/2024 15:11:56	Prakash	prakashpoint2005@gmail.com				
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							

FileEditView

Dear <RecipientName>,  
We are pleased to invite you to the 2024 Tech Innovations Symposium, which will be held on October 15, 2024, at The Grand Conference Center. This symposium will bring together industry leaders, experts, and enthusiasts from the technology and innovation sector to discuss the latest trends, innovations, and challenges.

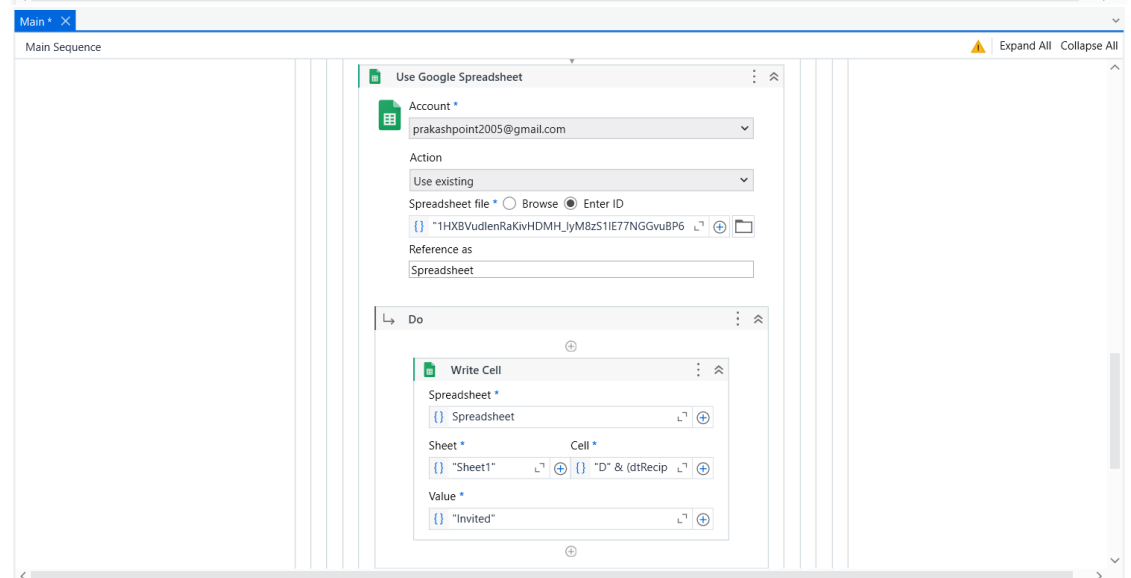
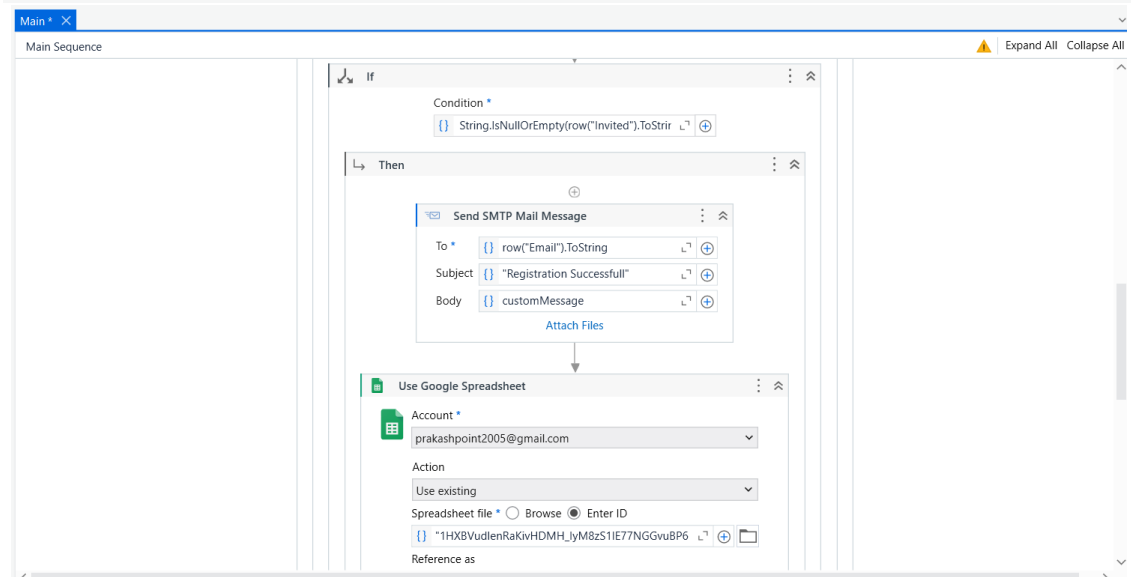
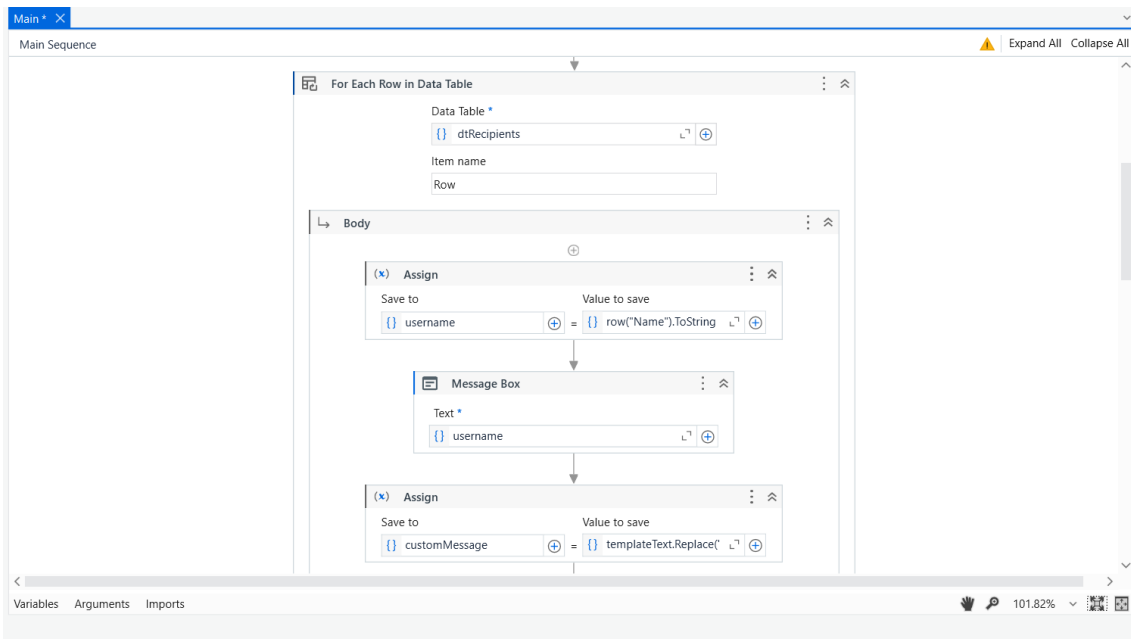
Event Details:  
Theme: Navigating Future Technologies  
Date: October 15, 2024  
Time: 9:00 AM to 5:00 PM  
Location: The Grand Conference Center, 123 Tech Blvd, Silicon Valley, CA  
Keynote Speakers: Dr. Jane Smith, Prof. John Doe, and Mr. Alan Turing  
This event promises to be an enlightening and engaging experience, offering you the opportunity to gain insights, network with peers, and explore potential collaborations.  
Please RSVP by October 1, 2024, to confirm your attendance. You can register through the following link: [www.techsymposium2024.com/register](http://www.techsymposium2024.com/register).  
We look forward to welcoming you to what promises to be an inspiring and fruitful event.

Best regards,  
Jessica Taylor  
Event Coordinator  
Tech Symposium Organizing Committee  
[contact@techsymposium2024.com](mailto:contact@techsymposium2024.com)  
+1 (555) 123-4567  
Tech Symposium Organization  
123 Tech Blvd, Silicon Valley, CA  
[Organization Website](#) | [Facebook](#) | [LinkedIn](#) | [Instagram](#)

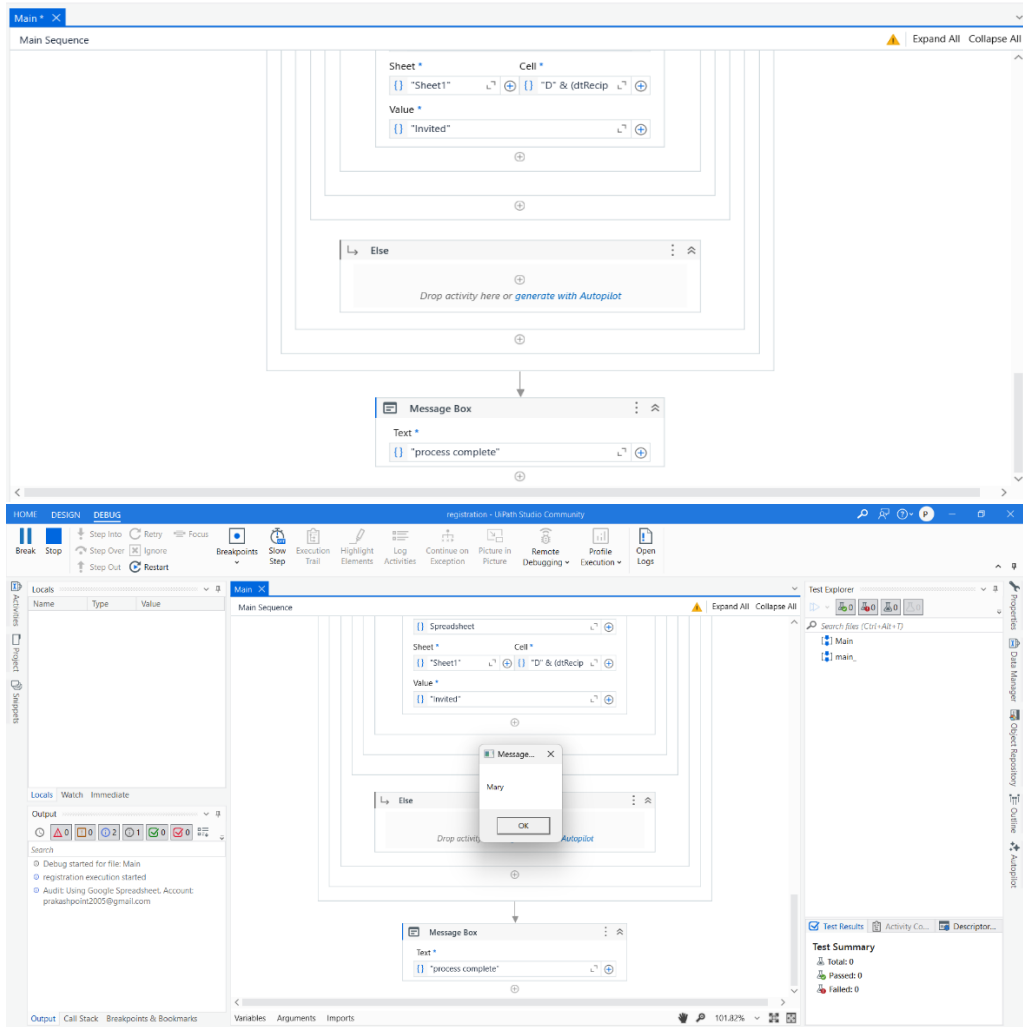
The screenshot displays the UiPath Studio Community Edition interface. The top menu bar includes HOME, DESIGN, and DEBUG. The main workspace shows a workflow sequence with the following activities:

- Read Range**: Configured with Google Sheets, Default (prakashpoint2005@gmail.com), Spreadsheet URL or ID (\*1HXBVudlenRakivHDMH\_lyM8zS1IE77NC), Range (sheet1), and Has headers (True).
- Read Text File**: Configured with Filename (\*D:\naan muthalvan\sem5\uiopathstudi).
- For Each Row in Data Table**: Configured with Data Table (\*dtRecipients) and Item name.

The left sidebar shows the Activities panel with various activity categories like Favorites, Recent, and Installed. The bottom status bar indicates the current workflow is 'Main Sequence'.







Form Responses 1						
File Edit View Insert Format Data Tools Extensions Help						
Form_Responses1						
Timestamp	Name	Email	Invited			
10/11/2024 18:29:30	Prakash	prakashpoint2005@gms	Invited			
11/11/2024 17:54:43	Joycy	joycynavis@gmail.com	Invited			
11/11/2024 20:24:04	PRAKASH	joyprakash2005@gmail.	Invited			
11/11/2024 22:03:44	Mary	joycynavis14@gmail.cor	Invited			
12/11/2024 09:36:03	S. Aruna Devi	ao1276504@gmail.com	Invited			
12/11/2024 15:11:56	Prakash	prakashpoint2005@gms	Invited			



## 12. Module Breakdown

Here's how we'll split the project into modules:

1. **Module 1: Google Form Creation and Google Sheets Setup**
  2. **Module 2: Project Setup and Initial Variables**
  3. **Module 3: Connecting to Google Sheets and Reading Responses**
  4. **Module 4: Loading the Email Template**
  5. **Module 5: Looping Through Responses and Preparing Email Content**
  6. **Module 6: Sending Emails with SMTP**
  7. **Module 7: Updating Google Sheets to Mark Invited Status**
  8. **Module 8: Final Message and Workflow Completion**
- 

### 12.1. Module 1: Google Form Creation and Google Sheets Setup

#### Goal

In this module, we'll create a Google Form to collect information from users (like names and emails), set up the form to send responses to a Google Sheets file, and customize the sheet so it's ready for our automation project.

#### Step-by-Step Guide

1. **Create a Google Form:**
  - Go to **Google Forms** by visiting <https://forms.google.com>.
  - Click on **Blank** to create a new form.
2. **Add Form Title and Description:**
  - In the untitled form, give it a title, like **Symposium Registration Form**.
  - Add a description below, such as "Please fill out your information to register for the symposium."
3. **Add Form Questions:**
  - Now we'll add some basic questions for collecting participant details:
    - **Name:**
      - Click **Untitled Question** and type "Name."
      - Set the question type to **Short Answer** (it should be set automatically).

- Toggle **Required** (make it a required field so participants can't skip it).
- **Email:**
  - Click + to add another question.
  - Type "Email."
  - Set the question type to **Short Answer** again.
  - Toggle **Required** to ensure everyone provides an email.
  - Optionally, click the three dots in the bottom right corner and select **Response Validation** to make sure users enter a valid email format.

The screenshot shows the Google Forms editor interface. At the top, there's a header with the form title 'Symposium Registration Form' and a star icon. Below the header, there are tabs for 'Questions', 'Responses', and 'Settings'. The 'Questions' tab is active, showing a list of questions. The first question is 'Name', and the second is 'Email'. Both questions are set to 'Short-answer text' and are marked as required. The 'Email' question has a validation rule set to 'Email address'. On the right side, there's a vertical toolbar with icons for adding questions, deleting questions, and other editing tools. At the bottom right, there's a 'Send' button.

#### 4. Configure Response Settings:

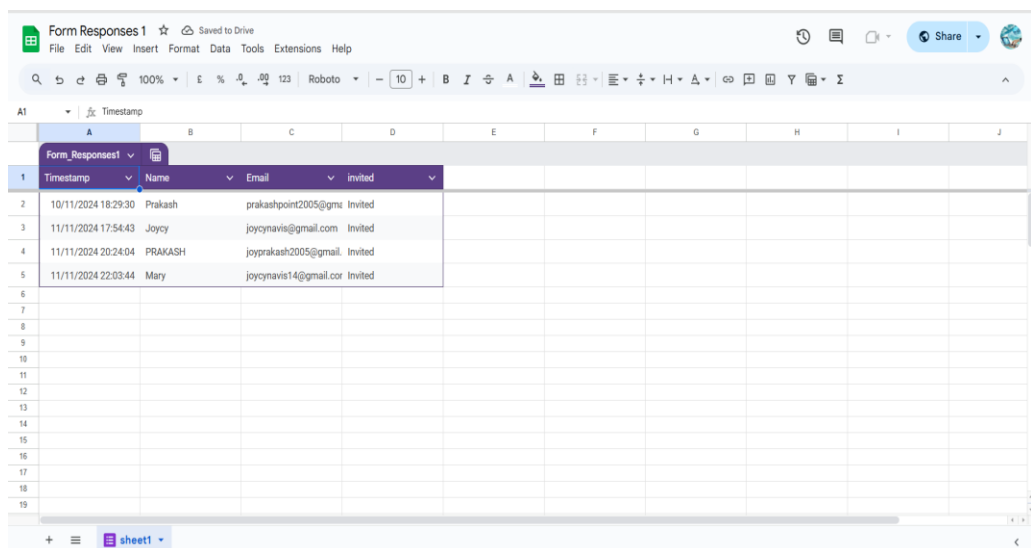
- Click on the **Settings** icon (the gear icon at the top).
- Under the **Responses** tab, make sure that **Collect email addresses** is unchecked (since we've already created a separate email field).
- Under **Presentation**, you can customize the confirmation message if you want, like "Thank you for registering! You'll receive an email invitation soon."
- Click **Save** when you're done.

#### 5. Link Form to Google Sheets:

- In the **Responses** tab of the form, click on the **Google Sheets** icon (green spreadsheet icon).
- This will create a new Google Sheets file where all form responses will be stored.
- Google will automatically name the sheet based on your form title, but you can rename it if desired.

## 6. View and Customize Google Sheets:

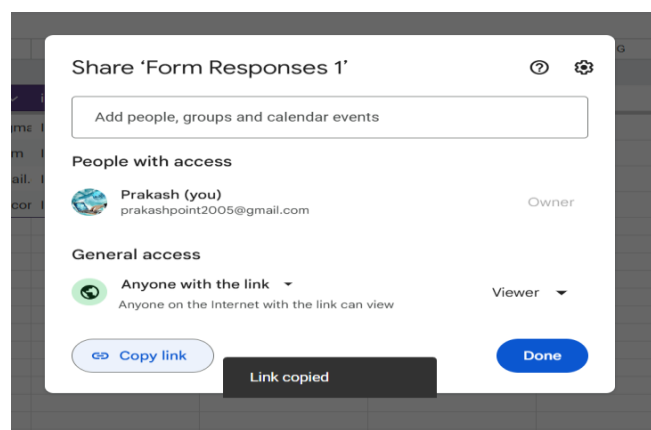
- Open the newly created Google Sheets file. You should see columns for **Timestamp**, **Name**, and **Email**.
- Add a new column header called **Invited** next to the **Email** column. This column will be used to track whether each participant has received an invitation email.
- Your columns should look like this:
  - **Timestamp** - Date and time of form submission.
  - **Name** - Participant's name.
  - **Email** - Participant's email address.
  - **Invited** - This will remain empty initially but will be updated by our automation process to track who has been invited.



	Timestamp	Name	Email	Invited
1				
2	10/11/2024 18:29:30	Prakash	prakashpoint2005@gmail	Invited
3	11/11/2024 17:54:43	Joycy	joycynavis@gmail.com	Invited
4	11/11/2024 20:24:04	Prakash	joyprakash2005@gmail	Invited
5	11/11/2024 22:03:44	Mary	joycynavis14@gmail.com	Invited
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

## 7. Copy the Google Sheets Link:

- In the Google Sheets file, go to the address bar and copy the URL of the spreadsheet. We'll use this link later in the UiPath project to access this specific sheet.



```
google sheet link: https://docs.google.com/spreadsheets/d/1HXBVudlenRaKivHDMH_IyM8zS1IE77NGGvuBP6Lo4pI/edit?usp=sharing
ID: 1HXBVudlenRaKivHDMH_IyM8zS1IE77NGGvuBP6Lo4pI
```

## Summary of Module 1

In this first module, we:

- Created a Google Form to collect participants' names and email addresses.
- Linked the form to a Google Sheets file to store responses.
- Added an **Invited** column to track which participants have received an invitation.

## 12.2. Module 2: Project Setup and Initial Variables

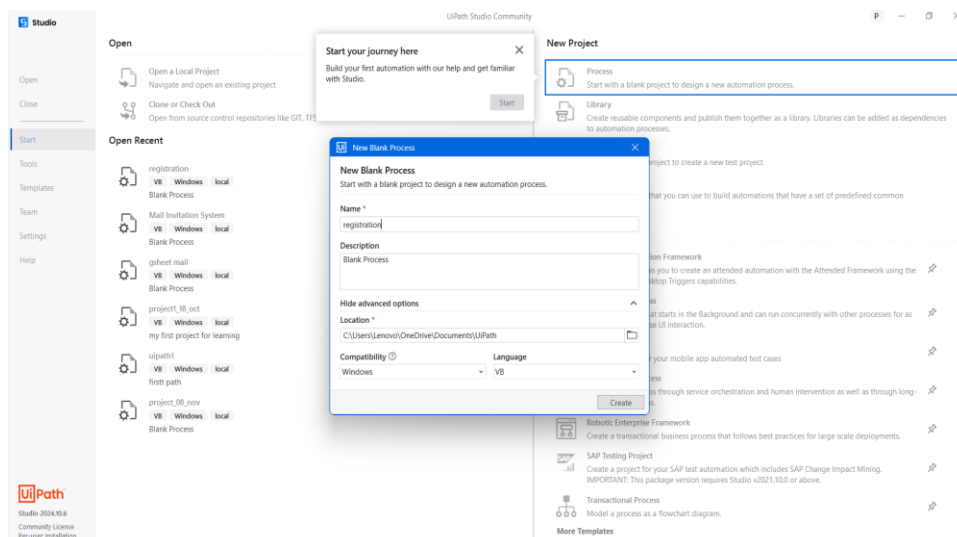
### Goal

In this module, we'll set up our UiPath project, create a new workflow, and define initial variables to manage data in the project. This setup will help us organize the automation and make it easier to perform tasks like reading data from the Google Sheet, creating custom email messages, and updating the sheet.

### Step-by-Step Guide

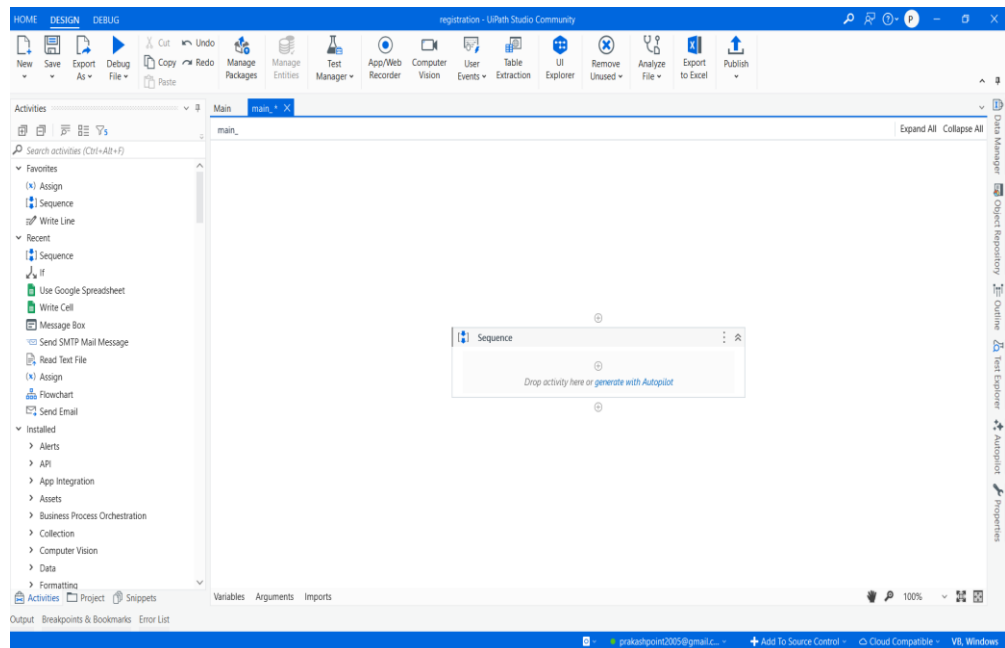
#### 1. Open UiPath Studio and Start a New Project:

- Launch **UiPath Studio**.
- On the home screen, click **New Project** and select **Process**.
- Name the project, for example, **SymposiumRegistrationAutomation**.
- Set the location for the project folder, choose a description if you want, and click **Create**.



## 2. Create a New Sequence:

- Once the project opens, you'll see the **Main.xaml** file.
- Double-click on **Main.xaml** to open it. This is where we'll design the main automation workflow.
- In the **Activities** panel, search for **Sequence**.
- Drag and drop a **Sequence** into the main design area. Rename it to **Main Sequence** to keep things organized.



## 3. Create Initial Variables:

- We'll need some variables to store data throughout the project. Let's start with the following variables:
  - **dtRecipients** - This will be used to store data from the Google Sheets file.
  - **templateText** - This will store the email message template text that we'll customize for each recipient.
- To create variables, go to the **Variables** panel at the bottom of the screen. If you don't see it, go to **View > Variables**.

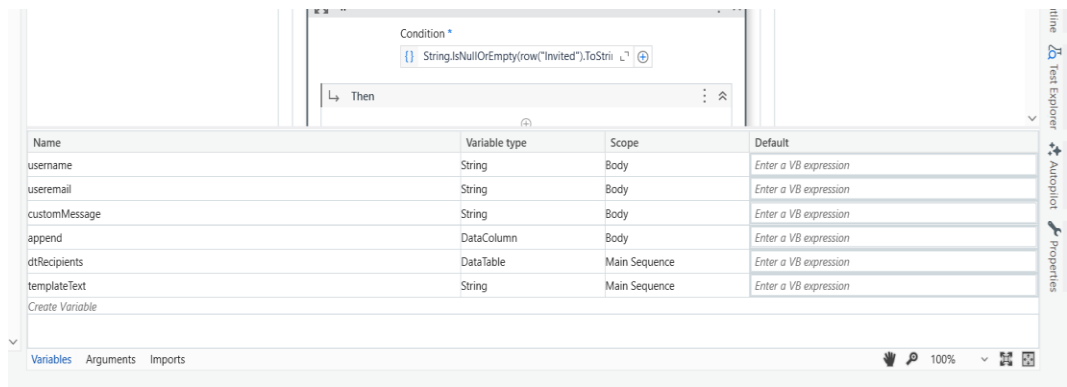
## 4. Define dtRecipients Variable:

- Click **Create Variable** in the Variables panel.
- Set the **Name** to dtRecipients.
- Set the **Variable Type** to **DataTable** (choose **System.Data.DataTable** from the dropdown).
- Leave the **Default** value blank and set the **Scope** to **Main Sequence**.

- This variable will store the data we read from Google Sheets.

#### 5. Define **templateText** Variable:

- Click **Create Variable** again.
- Set the **Name** to **templateText**.
- Set the **Variable Type** to **String**.
- Leave the **Default** value blank and set the **Scope** to **Main Sequence**.
- This variable will hold the template message text, which we'll later personalize for each participant.



#### 6. Prepare an Email Template File:

- To create custom emails for each recipient, we'll need an email template text file. We'll load this template text into **templateText** and replace placeholders (like <RecipientName>) with each participant's details.
- Open **Notepad** or any text editor.
- Write a basic email message template. For example:

Dear <RecipientName>,

Thank you for registering for the symposium. We are excited to have you join us!

Best regards,

The Symposium Team

- Save this file as **MailTemplate.txt** in a known location (e.g., D:\YourFolder\MailTemplate.txt). Remember the location, as we'll need it later to read this file into the **templateText** variable.

#### 7. Save the Project:

- Once you've created the variables and set up the sequence, save your project by clicking on **Save** or pressing **Ctrl + S**.



## Summary of Module 2

In this module, we:

- Created a new UiPath project and set up the **Main Sequence** workflow.
  - Created two key variables: dtRecipients (to hold data from Google Sheets) and templateText (to store the email message template).
  - Created an email template file (MailTemplate.txt) with placeholders to be customized later.
- 

## 12.3. Module 3: Connecting to Google Sheets and Reading Responses

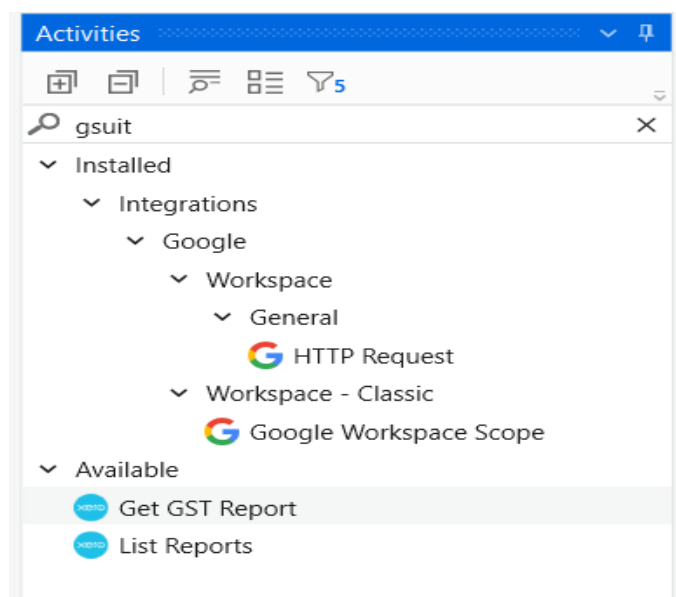
### Goal

In this module, we'll connect to a Google Sheets file where the form responses are stored, read the data from the sheet, and load it into the dtRecipients DataTable variable we created. This will give us access to the form responses so we can process each one.

### Step-by-Step Guide

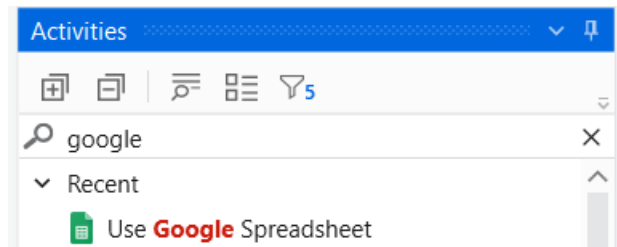
#### 1. Install UiPath GSuite Package:

- UiPath needs the **UiPath.GSuite.Activities** package to interact with Google services like Google Sheets.
- In UiPath Studio, go to **Manage Packages** (the icon looks like a box in the top ribbon).
- In the search bar, type “UiPath.GSuite.Activities.”
- Click on **UiPath.GSuite.Activities** from the list, then click **Install** and **Save**. This will add Google Sheets activities to UiPath Studio.



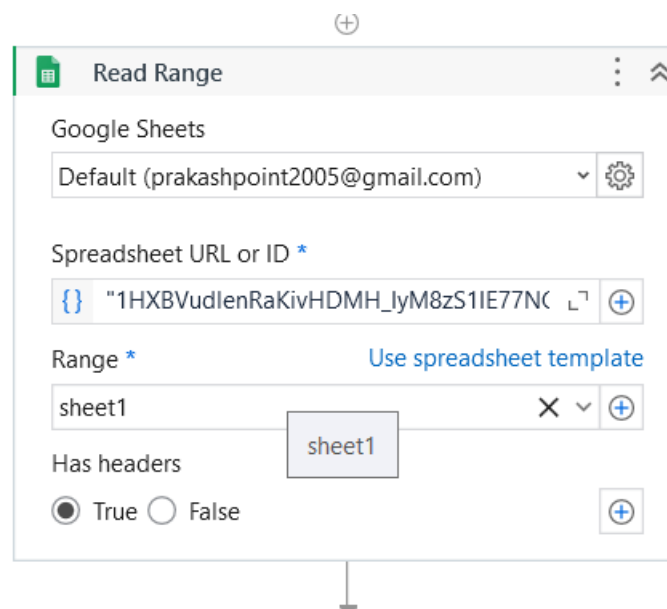
## 2. Add a “Google Sheets Application Scope” Activity:

- In the **Activities** panel, search for **Google Sheets Application Scope**.
- Drag and drop this activity into the **Main Sequence** after the start of the workflow.
- This activity opens a connection to a Google Sheets file, allowing us to perform actions like reading data.



## 3. Configure Google Sheets Connection:

- Select the **Google Sheets Application Scope** activity.
- In the **Properties** panel, look for **SpreadsheetURL** or **SpreadsheetID**:
  - Since we’re using a public Google Sheets file (the Form Responses sheet), we’ll enter its ID.
  - The **SpreadsheetID** is part of the Google Sheets link. For example, in the URL [https://docs.google.com/spreadsheets/d/1HXBVudlenRaKivHDMH\\_IyM8zS1IE77NGGvuBP6Lo4pI/edit](https://docs.google.com/spreadsheets/d/1HXBVudlenRaKivHDMH_IyM8zS1IE77NGGvuBP6Lo4pI/edit), the **SpreadsheetID** is 1HXBVudlenRaKivHDMH\_IyM8zS1IE77NGGvuBP6Lo4pI.
- Enter 1HXBVudlenRaKivHDMH\_IyM8zS1IE77NGGvuBP6Lo4pI in the **SpreadsheetID** field.

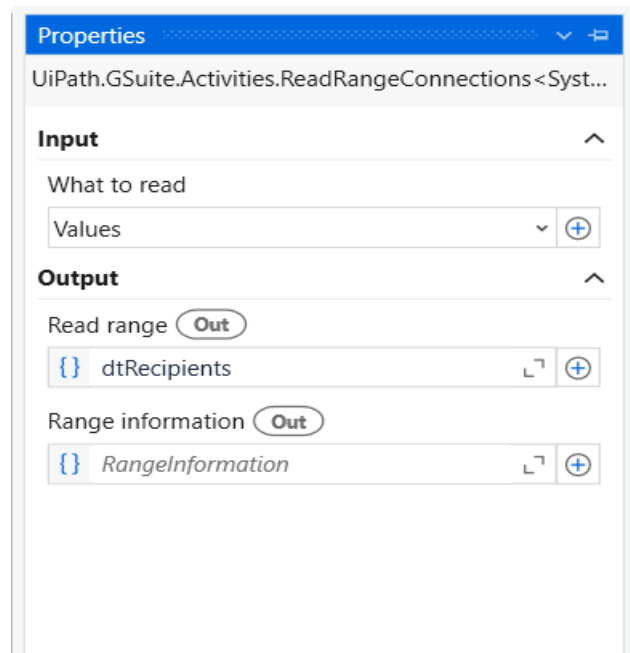


#### 4. Add “Read Range” Activity:

- Next, we’ll add an activity to read the data from the Google Sheets file.
- Inside the **Google Sheets Application Scope** (indent this activity within it), search for **Read Range** under **UiPath.GSuite.Activities**.
- Drag and drop **Read Range** into the **Google Sheets Application Scope**.

#### 5. Configure the Read Range Activity:

- Select the **Read Range** activity and configure it as follows:
  - **Range:** Leave this blank or set it to "sheet1". This will read all rows in the sheet (assuming "sheet1" is the name of your Google Sheets tab).
  - **Has Headers:** Check this box if your sheet has column headers (e.g., Timestamp, Name, Email, Invited).
  - **Output:** In the **Output** field, select the dtRecipients variable we created earlier. This tells UiPath to store the spreadsheet data in this variable.



#### 6. Check Connection (Optional):

- At this stage, it’s a good idea to check that UiPath is reading the data correctly.
- To do this, add a **Message Box** activity below the **Read Range** activity (but still within the **Google Sheets Application Scope**).
- Set the **Message Box** text to `dtRecipients.Rows.Count.ToString`. This will show the number of rows read from the spreadsheet, helping us confirm that it’s working.

- Run the project to check the row count. If you see a number corresponding to the number of responses in your Google Sheets, everything is connected properly.

### 7. Add a Message Box to Check Data Loading (Optional):

- After the Read Range activity, add a Message Box activity.
- Set the content to `dtRecipients.Rows.Count.ToString` to display the number of rows read from Google Sheets.
- This helps us verify that the data has been loaded correctly. When you run the workflow, this message box should display the number of form responses.

### Summary of Module 3

In this module, we:

- Installed the UiPath GSuite package to access Google Sheets.
  - Set up a **Google Sheets Application Scope** and configured it to connect to our response sheet.
  - Used the **Read Range** activity to read form responses into `dtRecipients`, a `DataTable` variable.
  - Verified the connection by checking the row count in a **Message Box**.
- 

## 12.4. Module 4: Reading the Email Template and Customization

### Goal

The goal of this module is to load a prewritten email template from a text file, store it in the `templateText` variable, and set it up so that it can be customized for each recipient. We'll be using placeholders in the email template, such as `<RecipientName>`, which we'll later replace with each participant's name.

### Step-by-Step Guide

1. **Create an Email Template File** (if not already done in Module 2):
  - Open **Notepad** (or any text editor).
  - Write a sample email message, like this:

Dear <RecipientName>,

Thank you for registering for the symposium. We are excited to have you join us!

Best regards,

The Symposium Team

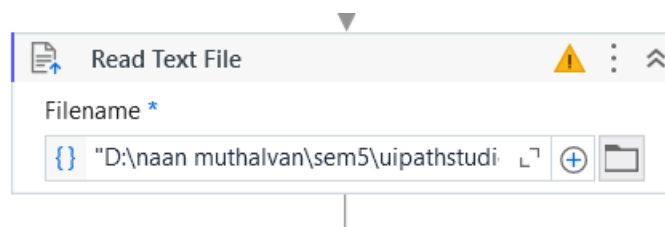
- Save the file as **MailTemplate.txt** in a location you can easily find (e.g., D:\YourFolder\MailTemplate.txt).

## 2. Add the Read Text File Activity:

- In the **Activities** panel in UiPath Studio, search for **Read Text File**.
- Drag the **Read Text File** activity into the **Main Sequence** after the **Read Range** activity.
- This activity will read the content of the text file and store it in the templateText variable.

## 3. Configure the Read Text File Activity:

- In the **Properties** panel for the **Read Text File** activity:
  - **FileName:** Type the full path to your email template file, or click the folder icon and navigate to the file. For example: D:\YourFolder\MailTemplate.txt.
  - **Output:** Set this to the templateText variable (the String variable we created in Module 2 to hold the email template).



## 4. Test the Template Loading (Optional):

- After the **Read Text File** activity, add a **Message Box** activity.
- Set the content to templateText to display the template content.
- This lets you verify that the text was loaded correctly from the file.
- Run the workflow to ensure it reads the template correctly. The **Message Box** should display the content of the email template.

## 5. Explanation of the Placeholders in the Template:

- In the email template text file, we used <RecipientName> as a placeholder. Later in the workflow, we'll replace <RecipientName> with the actual name of each participant, creating a customized message for each person.
- You can use other placeholders if needed, such as <EventDate> or <Location>. Just ensure that the template text is structured consistently, and that these placeholders are replaced in the next module.

#### 6. Save the Workflow:

- After confirming that the email template loads correctly, remove the **Message Box** activity if you don't need it anymore.
- Save the project by clicking **Save** or pressing **Ctrl + S**.

### Summary of Module 4

In this module, we:

- Created or verified an email template file with placeholders.
  - Used the **Read Text File** activity to load the email template content into the templateText variable.
  - Tested that the email template was loaded correctly.
- 

## 12.5. Module 5: Customizing the Email Template for Each Recipient

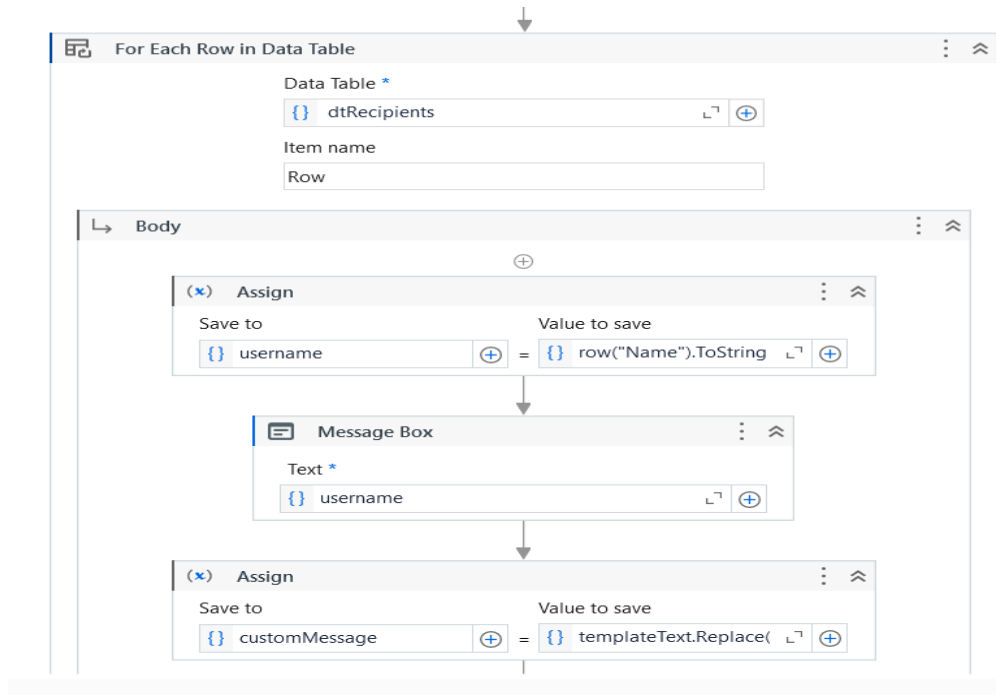
### Goal:

In this module, we will loop through the data in the dtRecipients DataTable and customize the email template for each recipient. We will replace placeholders (e.g., <RecipientName>) in the template with the actual participant's name and prepare the email for sending.

### Step-by-Step Guide

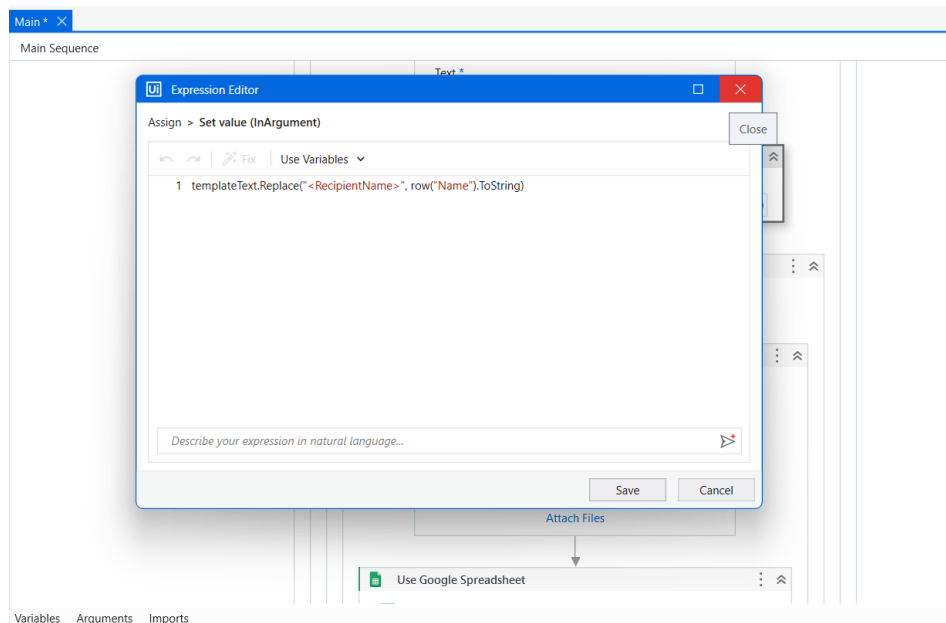
#### 1. Set Up a For Each Row Activity:

- We will loop through each row of the dtRecipients DataTable to personalize the email for every participant.
- In UiPath Studio, search for the **For Each Row** activity and drag it into the **Main Sequence**.
- Set the **DataTable** property to dtRecipients. This will iterate through each row in the DataTable.



## 2. Read Participant Information:

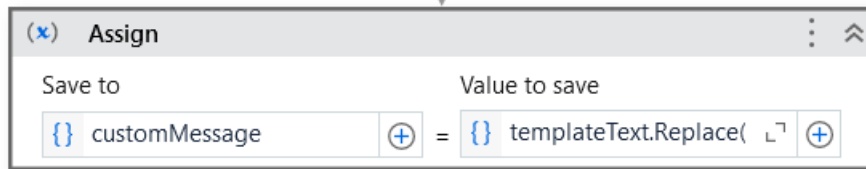
- Inside the **For Each Row** activity, use the **Assign** activity to extract the participant's name and email:
  - **Assign:** participantName = row("Name").ToString
  - **Assign:** participantEmail = row("Email").ToString
  - This retrieves the participant's name and email from each row.



## 3. Customize the Email Template:

- We will replace the <RecipientName> placeholder in the template with the participant's actual name.

- Add an **Assign** activity:
  - **Assign:** `customizedEmail = templateText.Replace("<RecipientName>", participantName)`
- This replaces the placeholder `<RecipientName>` with the actual name of the participant in the `customizedEmail` string variable.



#### 4. Test Customization (Optional):

- To verify that the customization works, add a **Message Box** activity after the **Assign** activity.
- Set the **Message Box** content to `customizedEmail`. This will display the personalized email for the current participant, allowing you to check the customization.

#### 5. Next Steps:

- We've now prepared the email body for each recipient.
- Save the project by clicking **Save** or pressing **Ctrl + S**.

### Summary of Module 5

In this module, we:

- Set up a **For Each Row** loop to process each recipient.
- Used the **Assign** activity to extract participant details (name, email).
- Customized the email template by replacing the `<RecipientName>` placeholder with the participant's name.
- Optionally verified the email customization with a **Message Box**.

## 12.6. Module 6: Sending Emails with SMTP

### Goal:

In this module, we will send the customized emails to each recipient using an SMTP server (e.g., Gmail's SMTP server). We will also ensure that the email is sent with the correct subject, body, and recipient address.

### Step-by-Step Guide

1. Set Up SMTP Settings:



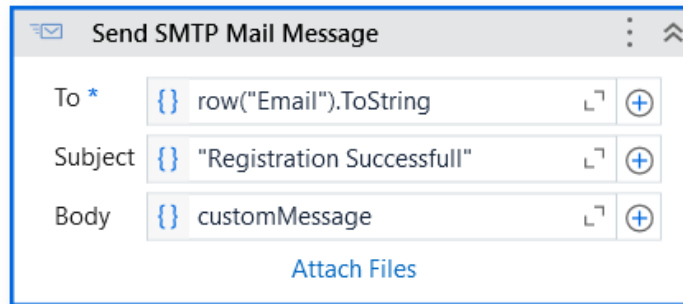
- Before sending the emails, you'll need the SMTP details for the email service you want to use. For Gmail, these are:
  - SMTP Server: **smtp.gmail.com**
  - Port: **587**
  - Use SSL: **True**
  - Username: Your Gmail address (e.g., your-email@gmail.com)
  - Password: Your Gmail password or App Password (for security reasons, it's recommended to use an App Password instead of your regular Gmail password).

## 2. Add Send SMTP Mail Message Activity:

- In UiPath Studio, search for **Send SMTP Mail Message** in the Activities panel.
- Drag the **Send SMTP Mail Message** activity inside the **For Each Row** loop (after the email customization step).

## 3. Configure the Send SMTP Mail Message Activity:

- **To:** Set this to the participantEmail variable (from the dtRecipients DataTable).
- **Subject:** Set this to a subject line of your choice. For example: "Symposium Registration Confirmation".
- **Body:** Set this to the customizedEmail variable, which contains the personalized email body for each recipient.
- **SMTP Server:** Enter smtp.gmail.com (or the SMTP server for the email provider you are using).
- **Port:** Enter 587 (the port for TLS encryption).
- **Username:** Enter the Gmail address or the sender's email address.
- **Password:** Enter your Gmail password or App Password (stored securely).
- **Is Body Html:** Set this to False unless you want to send the email in HTML format.



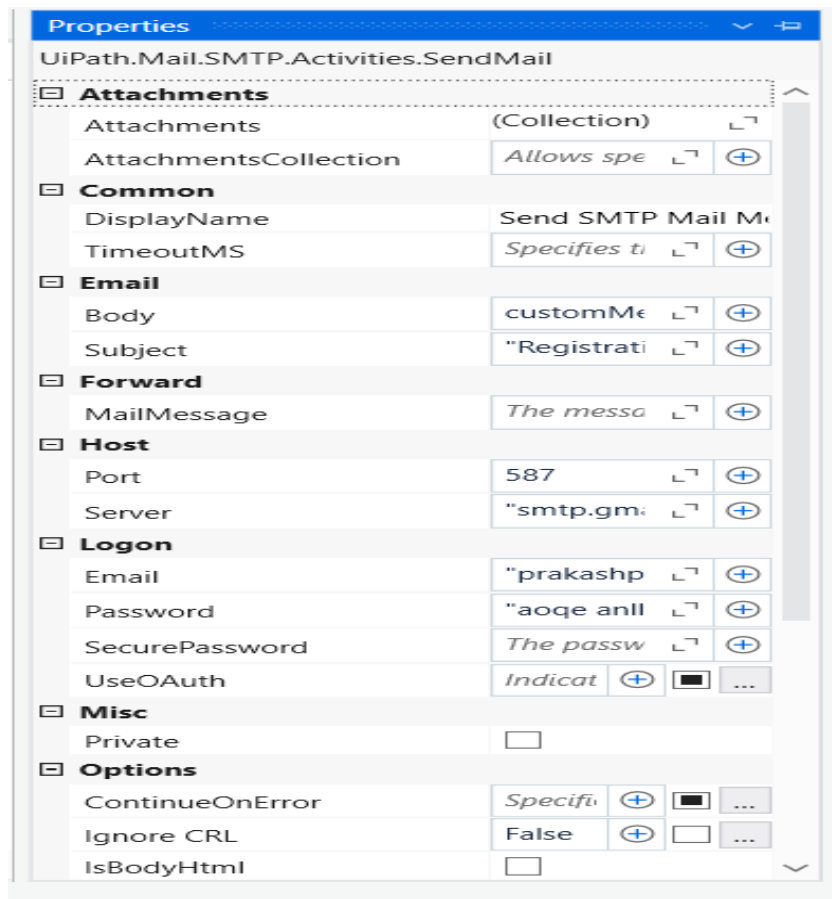
**Send SMTP Mail Message**

To \*

Subject

Body

[Attach Files](#)



**Properties**  
UiPath.Mail.SMTP.Activities.SendMail

**Attachments**

Attachments (Collection)

AttachmentsCollection Allows specifying attachments

**Common**

DisplayName Send SMTP Mail Message

TimeoutMS Specifies timeout in milliseconds

**Email**

Body customMessage

Subject "Registration Successfull"

**Forward**

MailMessage The message to be forwarded

**Host**

Port 587

Server "smtp.gmail.com"

**Logon**

Email "prakashp@outlook.com"

Password "aoqeanil@outlook.com"

SecurePassword The password to use for authentication

UseOAuth Indicates whether to use OAuth for authentication

**Misc**

Private ☐

**Options**

ContinueOnError Specifies whether to continue on error

IgnoreCRL False

IsBodyHtml ☐

#### 4. Test Email Sending (Optional):

- You can test the email sending by running the workflow with a test recipient. Add a **Message Box** before the **Send SMTP Mail Message** activity to confirm that the email body (customizedEmail) looks correct.
- When you run the workflow, ensure that emails are sent successfully.

#### 5. Handle Exceptions (Optional):

- It's good practice to handle errors in case the email fails to send. You can use a **Try Catch** activity around the **Send SMTP Mail Message** activity.
- In the **Catch** section, you can log or display the exception message to handle errors gracefully.

## 6. Save the Project:

- After configuring the email sending step, save your project by clicking **Save** or pressing **Ctrl + S**.

## Summary of Module 6

In this module, we:

- Set up the SMTP server and email settings to send emails.
  - Used the **Send SMTP Mail Message** activity to send the customized emails to participants.
  - Optionally tested the email sending and handled exceptions.
- 

## 12.7. Module 7: Updating the Google Sheet After Sending Emails

### Goal:

In this module, we will update the Google Sheet to mark which participants have received their confirmation emails. This will help us keep track of the progress and ensure that each participant has been notified.

### Step-by-Step Guide

#### 1. Use If Blocks:

Check if the recipient's email is missing or if the email has already been sent.

If block example:

If String.IsNullOrEmpty(row("Email").ToString) Then

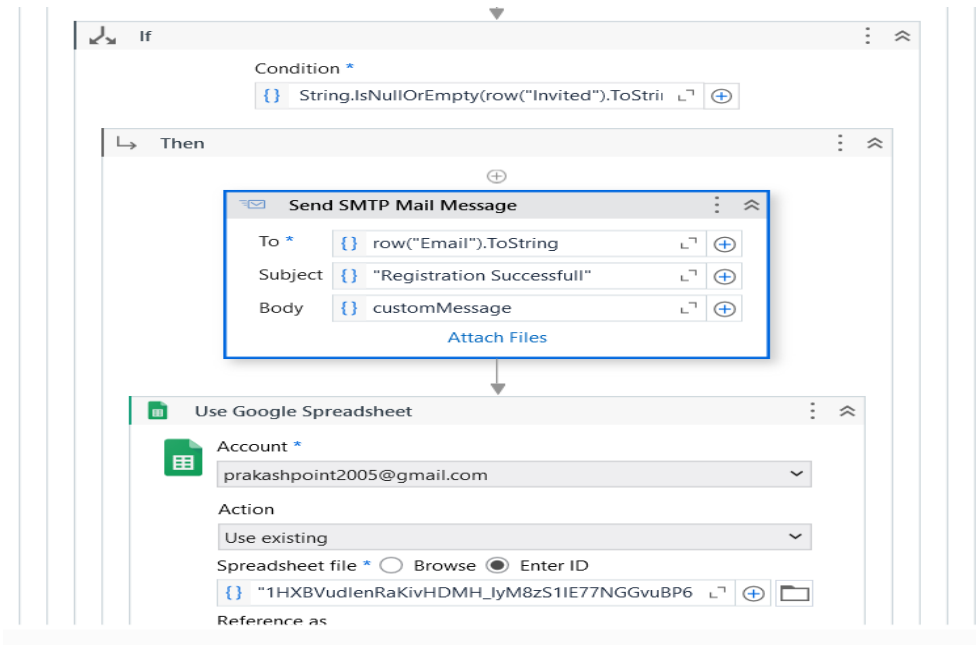
    ' Log error or skip this entry

    Log Message: "Email is missing for " & row("RecipientName")

Else

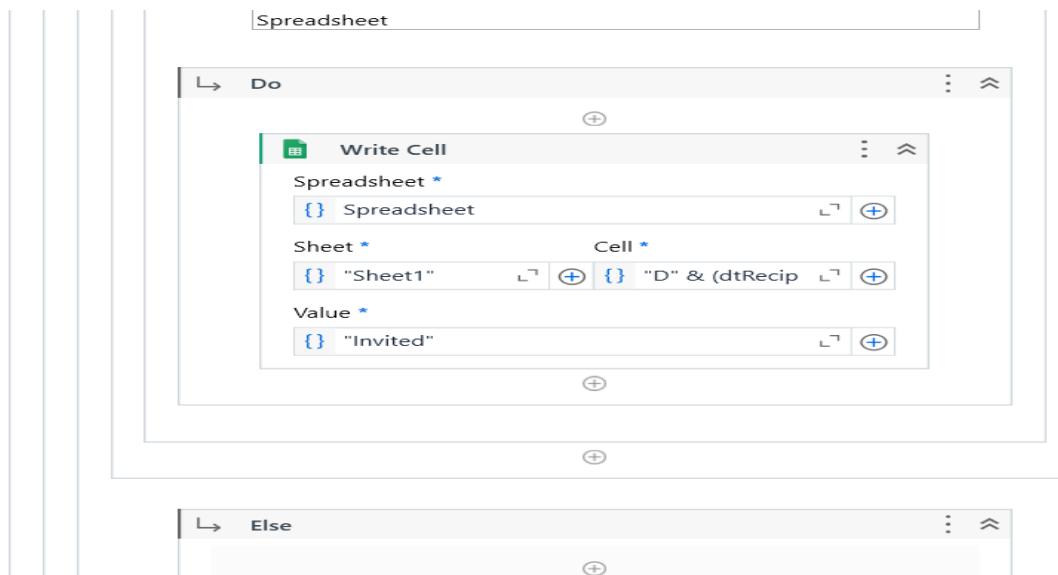
    ' Proceed with customization

End If



## 2. Read and Update the Data in UiPath:

- After sending the email, you'll need to update the corresponding row in the dtRecipients DataTable with a flag indicating the email has been sent.



## 3. Use the Assign Activity to Update the DataTable:

- After reading the data from Google Sheets, you might want to ensure that emails are only sent to participants who haven't been invited yet. For this, you can use an **If block** to check the "Invited" column..

E6    ▾    *fx*

	A	B	C	D	E	F	G	H	I	J
	Form_Responses1 ▾									
1	Timestamp ▾	Name ▾	Email ▾	invited ▾						
2	10/11/2024 18:29:30	Prakash	prakashpoint2005@gmc	Invited						
3	11/11/2024 17:54:43	Joycy	joycynavis@gmail.com	Invited						
4	11/11/2024 20:24:04	PRAKASH	joyprakash2005@gmail	Invited						
5	11/11/2024 22:03:44	Mary	joycynavis14@gmail.cor							
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

+    ≡    sheet1 ▾

#### 4. Use the Write Range Activity to Update Google Sheets:

- After you've updated the dtRecipients DataTable, use the **Write Range** activity (from the Google Sheets package) to push the changes back to Google Sheets.

Form Responses 1 ☆

File Edit View Insert Format Data Tools Extensions Help

🔍 ↶ ↷ 🖨️ 📋 100% ▾ | £ % .0 .00 123 | Defaul... ▾ | - 10 + | B I ⚡ A |

K10    ▾    *fx*

	A	B	C	D	E
	Form_Responses1 ▾				
1	Timestamp ▾	Name ▾	Email ▾	invited ▾	
2	10/11/2024 18:29:30	Prakash	prakashpoint2005@gmc	Invited	
3	11/11/2024 17:54:43	Joycy	joycynavis@gmail.com	Invited	
4	11/11/2024 20:24:04	PRAKASH	joyprakash2005@gmail	Invited	
5	11/11/2024 22:03:44	Mary	joycynavis14@gmail.cor	Invited	
6					
7					
8					

- Add the **Write Range** activity after the **For Each Row** loop where the email was sent. This will overwrite the existing data in the sheet with the updated EmailSent statuses.

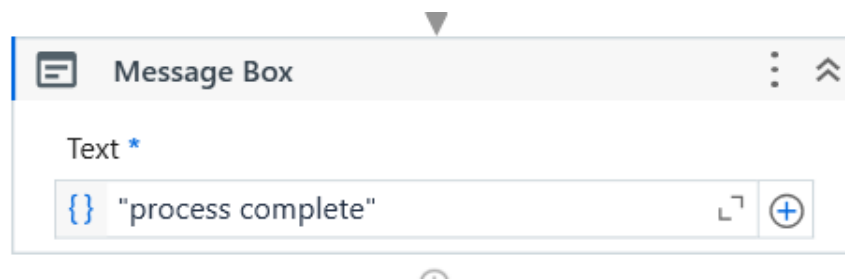
#### 5. Configure the Write Range Activity:

- In the **Write Range** activity, set the following properties:
  - **SpreadsheetID:** Use the same Google Sheets URL or ID that you used in the **Use Google Spreadsheet** scope.

- **SheetName:** Set this to the name of your sheet (e.g., "Form Responses 1").
- **Range:** Leave this blank to overwrite all the data in the sheet.
- **DataTable:** Set this to dtRecipients (the updated DataTable with the "EmailSent" column marked).

#### 6. Test the Update:

- Run the workflow to ensure that the "EmailSent" column is correctly updated for each participant after sending their email.
- You can add a **Message Box** after the **Write Range** activity to confirm that the Google Sheet has been updated. For example, you can display dtRecipients.Rows.Count.ToString to show the number of rows updated.



#### 7. Handle Errors (Optional):

- You may want to wrap the **Write Range** activity inside a **Try Catch** to handle any potential errors when updating the Google Sheets file (e.g., connection issues).

#### 8. Save the Project:

- After configuring the email update process, save your project by clicking **Save** or pressing **Ctrl + S**.

### Summary of Module 7

In this module, we:

- Updated the Google Sheet to mark participants as "EmailSent" after sending them the confirmation email.
- Used the **Write Range** activity to push the changes back to Google Sheets.
- Ensured the workflow updates the Google Sheet correctly.

## 12.8. Module 8: Final Testing and Deployment

### Goal:

In this final module, we'll perform comprehensive testing of our workflow to ensure everything works as expected, and we'll prepare the project for deployment. This will ensure that the automation is stable, reliable, and ready for use in a live environment.

### Step-by-Step Guide

#### 1. Test the Entire Workflow:

- Ensure that you've completed all the previous modules correctly.
- Before deploying the automation, thoroughly test the workflow by running it in **Debug** mode.
- Check for any errors or unexpected behavior:
  - Are emails being sent to the correct recipients?
  - Is the Google Sheet being updated with the "EmailSent" status?
  - Are all placeholders being replaced correctly in the email template?

#### 2. Verify Data Accuracy:

- Check that the data read from Google Sheets (dtRecipients) matches the actual responses in your Google Sheet.
- Confirm that the email template (templateText) contains the correct placeholders and that they are being replaced with the appropriate participant data.

#### 3. Test Edge Cases:

- Test with different scenarios:
  - A Google Sheet with no data (to ensure the workflow handles this case without errors).
  - A Google Sheet where all recipients have already been sent an email (to verify that the workflow doesn't resend emails).
  - Test with a small subset of data (e.g., 2-3 recipients) before running the workflow with the entire dataset.

#### 4. Log and Error Handling:

- Add logging to your workflow to capture any issues that might arise. For example, use **Log Message** activities to record key actions like when an email is sent or when data is updated in Google Sheets.
- Use **Try Catch** blocks to handle any potential errors. For example, if the Google Sheets connection fails, the workflow should log the error and continue with the next recipient.

## 5. Optimize Performance (Optional):

- If your Google Sheets file contains a large amount of data, consider optimizing the workflow by processing batches of rows instead of loading the entire sheet at once.
- Use filtering techniques to narrow down the data before reading or writing to Google Sheets.

## 6. Prepare for Deployment:

- Once you're confident that everything works as expected, ensure the following:
  - **Clear Unnecessary Logs/Message Boxes:** Make sure to remove any Message Boxes used for debugging.
  - **Document the Workflow:** It's always helpful to document your workflow so others (or your future self) can understand it. You can use annotations in UiPath Studio or maintain a separate documentation file.
  - **Version Control:** If you're using version control (e.g., Git), make sure to commit and push the latest version of your project.

## 7. Deploy to Production:

- If you're using **UiPath Orchestrator**, deploy your process to Orchestrator to run it on a schedule or trigger it via an event.
- If you're running it manually, ensure that your Google Sheets file and email credentials are set up properly on the machine that will execute the automation.
- Ensure all required packages (like Google Sheets and SMTP) are installed on the production machine.

## 8. Monitor the Process:

- After deployment, monitor the first few runs of the process to make sure everything is functioning as expected.
- If you're using Orchestrator, monitor the job logs for any issues.

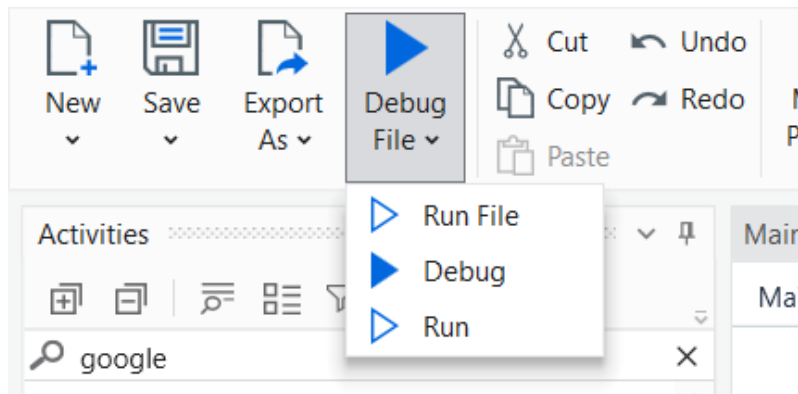
## 9. Handle Future Updates:

- If there are changes in the form, Google Sheets structure, or email template in the future, be sure to update the UiPath workflow accordingly.
- If the Google Sheets API changes, monitor for any necessary updates to the UiPath.GSuite activities.

## 10. Run Project

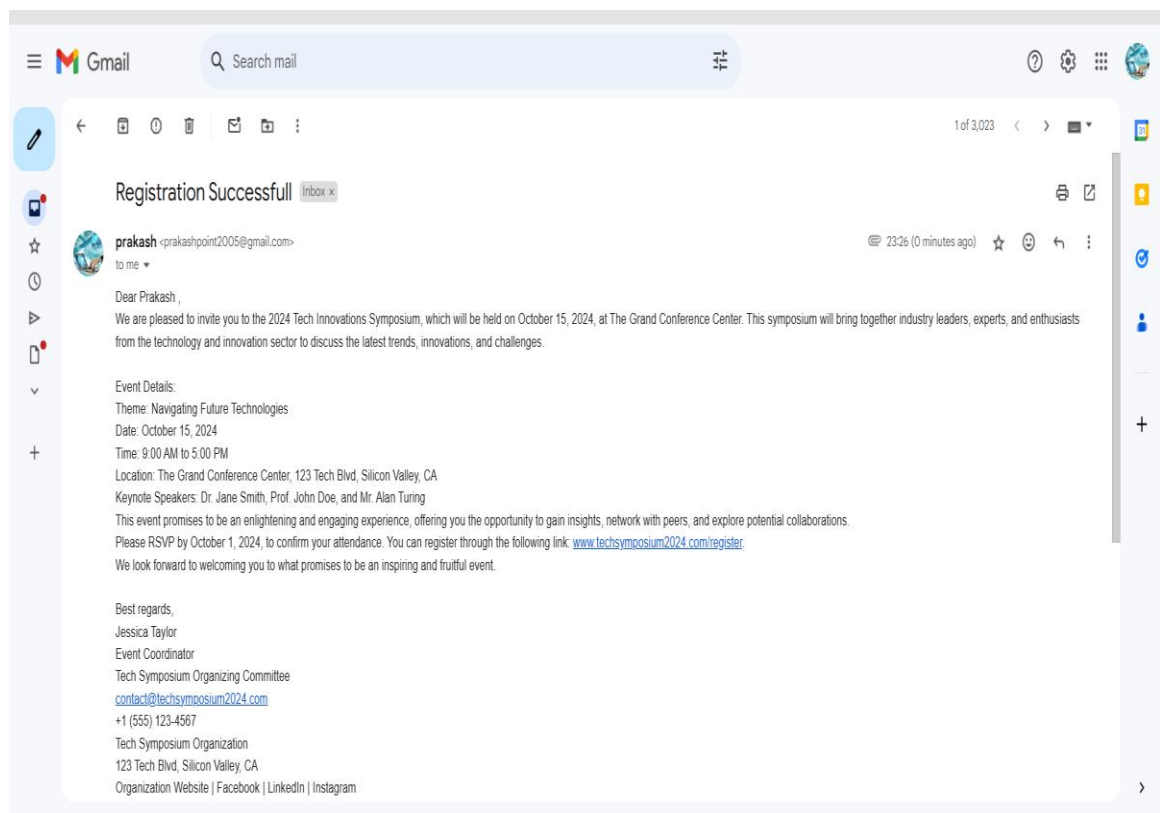
- Now run the project by clicking the **Run** icon.

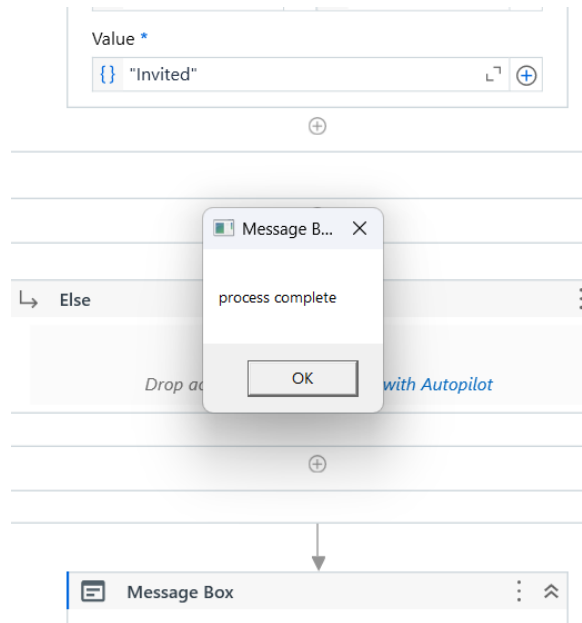




## 11. Received Mail

- Now the customized mail will be sent to the not-invited person's email.





## Summary of Module 8

In this final module, we:

- Performed comprehensive testing to ensure the workflow works correctly.
- Optimized the workflow and added logging/error handling.
- Prepared the workflow for deployment, ensuring everything was in order for production.
- Monitored the process after deployment to ensure stability.

## 13. Conclusion

By completing this automation process, we've developed a reliable and efficient workflow for sending personalized confirmation emails to participants using data from a Google Sheet. The key steps involved customizing email templates for each recipient, sending emails via SMTP, and updating the Google Sheet to track which participants have received their emails. Through this project, we've also ensured error handling, logging, and comprehensive testing to guarantee the workflow's smooth operation.

**In summary**, the workflow:

1. Customizes email templates for each recipient.
2. Sends emails using SMTP, ensuring correct subject, body, and recipient.
3. Updates the Google Sheet to track email statuses.
4. Is thoroughly tested and optimized for performance.
5. Is ready for deployment and future monitoring.

By following this approach, you can automate the process of sending personalized emails, making it more efficient and less error-prone. This automation also ensures that you stay organized and track progress easily by updating the Google Sheet in real-time. Once deployed, the workflow can be managed, monitored, and updated as needed, ensuring it remains effective and scalable.

## 14. Diploma of Completion

