| Diploma in Computer Engineering  Intelligent Automation (INAUTO) |  |
| --- | --- |

**Project**

**(100 marks - 30% subject component weightage)**

**Objective**

To develop a standalone **Robotic Process Automation (RPA) application** executable on the **UiPath** Studio platform.

**Requirements**

1. You are required to design & build RPA Application in areas of web applications (MS Bing, ChatGPT, MS Teams), MS Office applications, Email and PDF applications.
2. The RPA application development shall incorporate developmental best practices, such as proper naming conventions, effective exception handling, and documentation standards. This shall help in instilling professional work habits for software maintenance and scaling.
3. The RPA application must include the use of argument as a means of passing between workflows and must include the use of the Flowchart facility from the UiPath platform. The RPA application must also include use of DataTable.
4. Assessment of Business Impact: the RPA project shall provide benefits to the users such as time saved, errors reduced, and productivity improvements.

**Project Schedule**

| **Week** | Tasks |
| --- | --- |
| 7 | Project Requirements Brief |
| 9 | **Project Proposal Submission (Draft)** |
| 11 | Project Development  **Project Proposal Submission (Final)** |
| 13 | Project Development  Project Development (**Progress Review**) |
| 15 | Flex Week 2 |
| 16-17 | **Project Submission & Assessment** |

#### Project Scope and Requirements

#### Design & build an RPA app in the following domains:

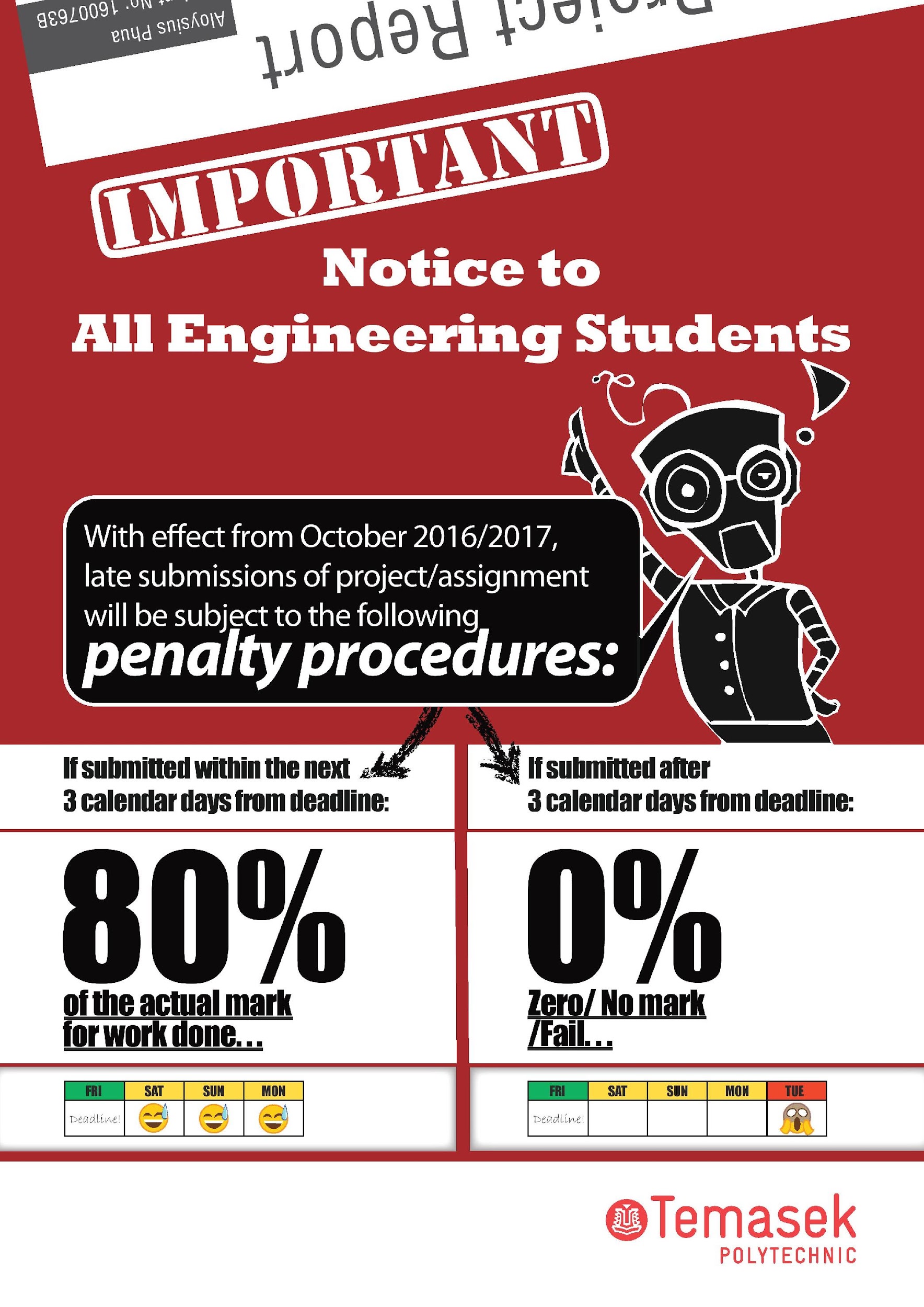
#### Web applications, business, education, finance, or others.

1. Deliverables:
   1. **Project Proposal**
   2. **Progress Review**
   3. **Project integration and demo of UiPath Application with annotation of codes**
   4. **Submission of the complete project and softcopy of Final Project Report.** **At project completion, a reflection of the learning journey, challenges faced, and how they overcame them must be included in the Final Project Report.**
2. Students will work in team of **two persons**. Each student in the team must work on creating the workflows using UiPath Studio.

#### IMPORTANT: Plagiarism of any app solution from the Internet is strictly prohibited.

#### All projects must be submitted latest by Week 17 based on allocation.

#### Late submissions of project will be subjected to the following penalty procedures:



**Final Project Submission & Interview (Week 16-17)**

Before or on the deadline and prior to the interview, you need to submit the following. (Your tutor has the discretion of modifying this. Please refer to him/her on this):

1. **Soft copy of your project** (zip of the entire folder containing the UiPath “xaml” files, the excel/word/pdf files used, soft copy of project report and any other data files used). Your tutor must be able to run your scripts from the contents you have provided for evaluation.
2. Project report
   1. Objectives
   2. Functionalities
      1. Description of functionalities (indicating clearly individual contributions)
   3. Learning Journey & Challenges Overcame
   4. Conclusion
   5. Appendix: UiPath scripts (screenshots of workflow and output results)
3. Signed Declaration form.

Interviews will only be done after the project is submitted. Failure to attend the interview and show the demonstration may lead to failure in the project.

**Summary of Deadlines**

| **Description** | **Dates** |
| --- | --- |
| Project Proposal (Final) | 31 Dec 2023 **(Week 11)** |
| Project Progress Review 1 | **Week 13** |
| Final Project Submission including Project Report, Video or Demonstration & Interview | 4 Feb 2024 **Week 16**  Make up: **Week 17** |

| **IMPORTANT NOTE:**  Students must submit their **own work** and **not copy programs from fellow classmates or some other sources.** Failure to comply will result in disciplinary action. Also, if a project group is found to have submitted work done by another project group, both groups **WILL NOT** have any marks for their projects. Take note that the disciplinary action taken will depend on severity and can range from failing the affected component of the subject to suspension/removal from course. |
| --- |

**INAuto Project Report**

| Team Name: | Applicant Screener | Class: PE04 |
| --- | --- | --- |
| Team Members: | Andrina Wei Ning Morrison | |
| Yoong Wai Kit | |

Applications Used

MS Office App 1 🗹 Excel\_\_\_\_ MS Office App 2 🗹 Outlook\_\_

MS Office App 3 ☐ \_\_\_\_\_\_\_\_ MS Office App 4 ☐ \_\_\_\_\_\_\_\_

Web Application 1 ☐ \_\_\_\_\_\_\_\_ Web Application 2 ☐ \_\_\_\_\_\_\_\_

Web Application 3 ☐ \_\_\_\_\_\_\_\_ Web Application 4 ☐ \_\_\_\_\_\_\_\_

Email 🗹 PDF 🗹

Others, please indicate 🗹 OpenAI API\_ Others, please indicate ☐ \_\_\_\_\_\_\_\_\_\_

**Objectives**

The Applicant Screener app is a comprehensive solution designed to streamline and automate the manual and labour-intensive recruitment screening process for organisations. Leveraging UiPath Studio for robotic process automation, the app integrates AI technologies such as OpenAI Generative Pre-trained Transformer (GPT) models for text classification and resume information extraction as well as the Doc2Vec Natural Language Processing (NLP) algorithm for applicant match scoring.

By harnessing the power of AI with automation, the app helps to expedite the screening process of large volumes of job applications in organisations while maintaining a high standard of accuracy, consistency, and fairness in evaluating the potential of candidates across various job postings. Organisations using the app will experience lower manpower costs, reduced hiring time, and better-informed hiring decisions.

**Functionalities**

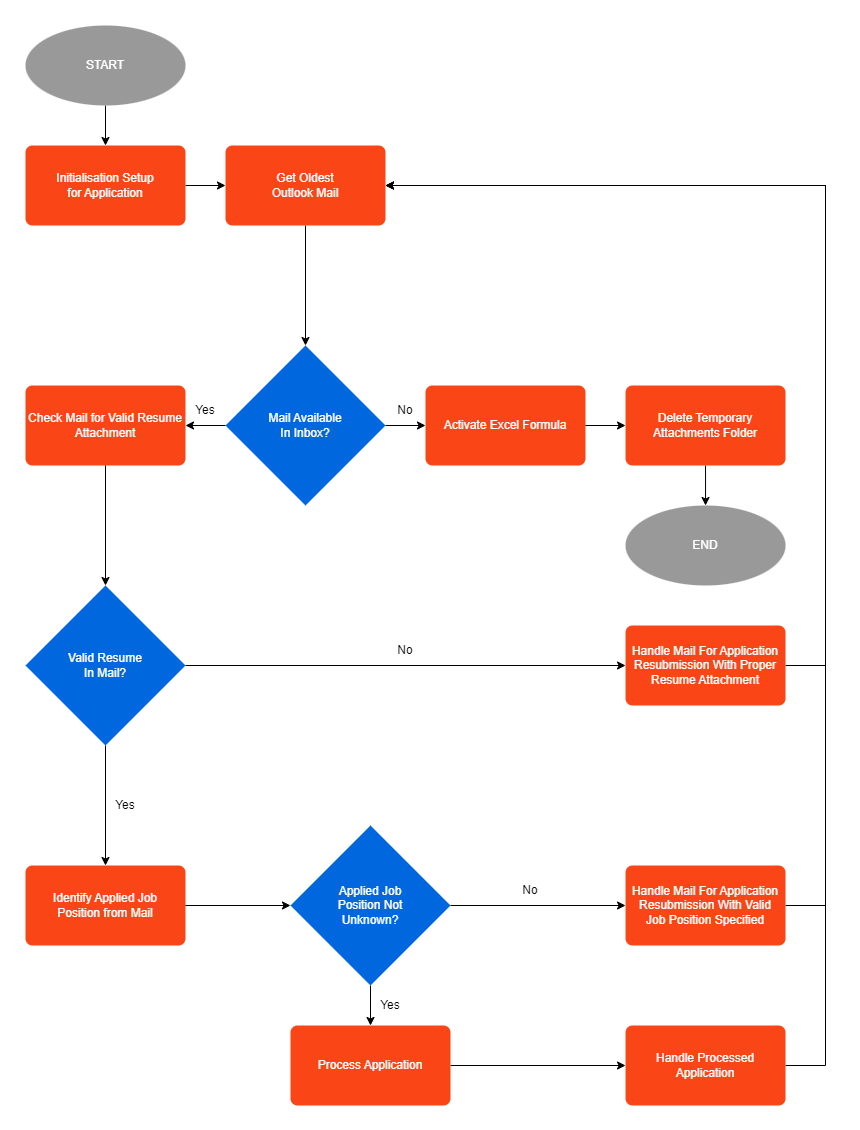


Figure 1: Applicant Screener Workflow/Flowchart Design Structure

The Applicant Screener app screens and shortlists applicants from job application emails in Outlook via the following processes:

1. **Initialisation Setup for Application** - Perform the necessary setup required for the app to begin processing job application emails.

* Obtain email accounts from the Outlook app.
* Prompt users to select the company Outlook email account via a dropdown menu.
* Read job positions, descriptions, and shortlisting scores from the JobPosting.xlsx file.

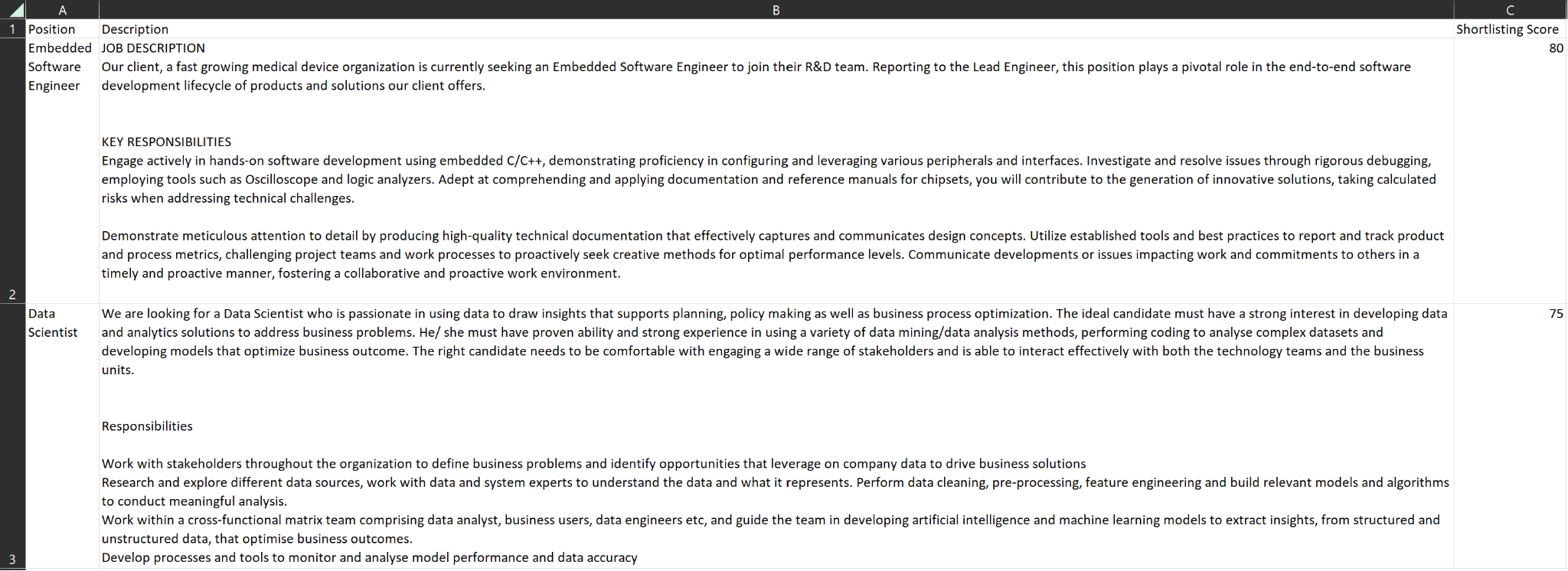


Figure 2: JobPosting.xlsx File with Position, Description, and Shortlisting Score Columns

* Create folders for listed job positions and an 'Invalid' folder in the Outlook Inbox if they are not already created.

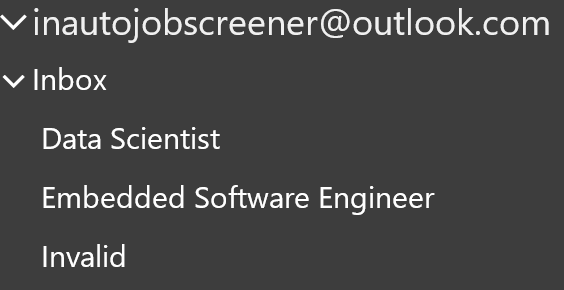


Figure 3: Outlook Inbox With Job Position Folders and the Invalid Folder

* Build a datatable with the required headings for processed job applications as the columns.
* Create an empty JobApplication Excel file with the current date and time from the datatable, with a sheet for each job position.

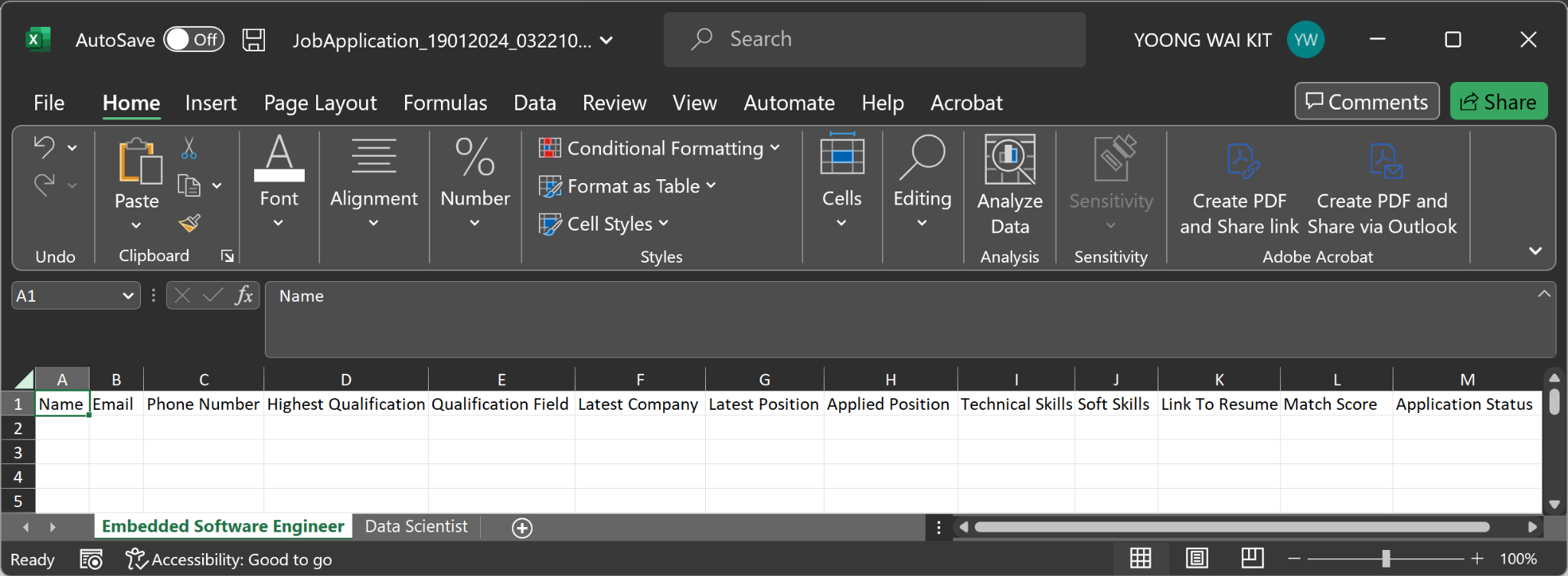


Figure 4: Empty JobApplication Excel File With Current Date & Time and a sheet for each job position

* Create a ‘Resume’ Folder with subfolders for each job position in the program directory to store applicants' resumes.

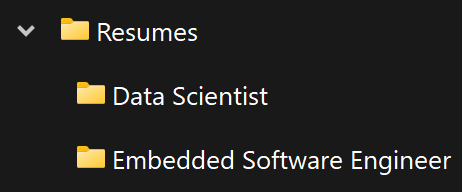


Figure 5: Resume Folder With Job Position Subfolders In Program Directory

1. **Get Oldest Outlook Mail** - Retrieve the oldest email in the company Outlook email account.
2. **Check Mail for Valid Resume Attachment** - Check whether the email contains a single valid resume attachment in .pdf format.

* If there are zero attachments, multiple attachments, or the attachment is not a .pdf file, the mail is considered to have no valid resume.
* Otherwise, the attachment is saved under the “Attachments” folder and used in UiPath's classify document activity.
* The mail is only considered to have a valid resume if the classify document activity identifies that the document type of the attachment is a resume.

1. **Identify Applied Job Position from Mail** - Classify the applicant’s applied job position from the email subject & body as an available job position or ‘Unknown’.

* Create a request body in JSON Object format with the prompt to identify the applied job position from the email subject & body.
* Serialise the JSON request and send it to OpenAI’s API Endpoint via HTTP request.
* Deserialise OpenAI API's response into JSON Object format.
* Get the applied job position from OpenAI API’s response.
* Set the applied job position to ‘Unknown’ if it is not found in the listed job positions.

1. **Process Application** - Extract the applicant’s details and determine the applicant’s match score with the job description from the resume. Move the applicant’s resume to the applied job position subfolder in the program directory’s ‘Resume’ folder.

* Extract text from the resume .pdf format using UiPath Document OCR.
* Retrieve the job description and shortlisting score for the applied job position.
* Perform text preprocessing on the resume content and job description.
* Use Powershell to activate the Python virtual environment and execute the script to obtain the match score based on the resume content and job description.
* Set the application status to ‘Shortlisted’ if the match score is greater than or equal to the shortlisting score. Otherwise, set the application status to ‘Not Shortlisted’.
* Extract the applicant details from the resume using UiPath’s extract document data activity.
* Move the applicant’s resume to the applied job position subfolder in the program directory’s ‘Resume’ folder.
* Create an object array with the extracted applicant details, match score, and Excel hyperlink formula with the path to the applicant’s resume.

1. **Handle Processed Application** - Update the processed applicant details into the JobApplication Excel file and move the email to the applied job position folder in Outlook

* Retrieve details of previously processed applicants from the applied job position Excel sheet into a datatable.
* Add the processed applicant details from the object array as a new datarow in the datatable.
* Sort the applicants by match score, from highest to lowest, in the datatable.
* Write the sorted datatable back into the applied job position Excel sheet.
* Move the email to the Outlook folder for the applied job position.

1. **Handle Mail for Application Resubmission** - Reply emails with instructions for resubmission of the job application and move the email to the ‘Invalid’ folder in Outlook.

* Possible instructions for resubmission:

1. Attach a single resume in .pdf format.
2. Clearly state a valid position from the job postings.

* Reply the email with the specific instruction for resubmission.
* Move the email to the ‘Invalid’ Outlook folder.

1. **Activate Excel Formula** - Activate the Excel formula for the hyperlink to the applicant’s resume in the JobApplication Excel file.

* Activate the Excel hyperlink formula by rewriting it for each individual cell in the ‘Link To Resume’ column for every sheet of the JobApplication Excel file.

1. **Delete Temporary Attachments Folder** - Delete the ‘Attachments’ folder from the program directory if it exists after processing all emails in Outlook.

We also added an additional **Send Sample Job Application Emails** process to send out job application emails stored in an excel file for easy testing of the Applicant Screener app.

| **Week** | **Work Done** | **Action By** |
| --- | --- | --- |
| 7 | Project Idea Research | Andrina & Wai Kit |
| 8 | Finalise Project Idea | Andrina & Wai Kit |
| 9 | UiPath Studio Generative Extractor Research & Evaluation | Wai Kit |
| 9 | Online Resume Parser & Scorer Research & Evaluation | Andrina |
| 9 | Match Score Model Training & Evaluation | Wai Kit |
| 9 | Complete Project Proposal | Andrina & Wai Kit |
| 10 | Initialisation Setup for Application Workflow | Andrina & Wai Kit |
| 10 | Get Oldest Outlook Mail | Andrina |
| 10 | Check Mail for Valid Resume Attachment Workflow | Andrina |
| 10 | Identify Applied Job Position from Mail Workflow | Wai Kit |
| 11 | Process Application Workflow | Wai Kit |
| 11 | Handle Processed Application Workflow | Andrina & Wai Kit |
| 11 | Handle Mail for Application Resubmission Workflow | Andrina |
| 12 | Activate Excel Formula Workflow | Wai Kit |
| 12 | Delete Temporary Attachments Folder Workflow | Andrina |
| 13 | Create Main Workflow with Flowchart & Invoked Workflows | Andrina & Wai Kit |
| 13 | Error Handling | Andrina & Wai Kit |
| 14 | Send Sample Job Application Emails Workflow | Wai Kit |
| 14 | Project Demonstration Video Filming | Andrina & Wai Kit |
| 14 | Complete Project Report | Andrina & Wai Kit |

**Learning Journey & Challenges Overcame**

We found the learning journey throughout the project development process extremely meaningful and fulfilling. The idea for this project started when Andrina saw Wai Kit's organised Outlook account with all emails grouped into folders. After talking about this observation, we suddenly had the idea for our INAuto project to be able to help companies manage the large number of job applications they receive via email. This idea eventually became the Applicant Screener app built for our INAuto project.

For the development of the project, we faced several challenges. However, with our determined attitude to overcome every challenge, we have learned a lot of new knowledge while successfully developing the Applicant Screener app.

Firstly, a challenge we faced was the creation of folders in Outlook. When looking through UiPath Studio, we could not find any activity that could create folders in Outlook. However, after we conducted thorough online research, we found that we could use Microsoft.Office.Interop.Outlook to retrieve email accounts, get folders, and create folders in Outlook. Using what we have learned, we successfully created Outlook folders for the listed job positions in the JobPostings.xlsx file. Moreover, we also used this knowledge to solve inefficiencies and potential errors when users manually enter their company Outlook email account when prompted at the start of our program. Instead, we modified our program to retrieve all email accounts configured in the device's Outlook to create a dropdown list for users to choose from.

Secondly, another challenge we experienced was using OpenAI's API to classify the content of the email application into an available job position. Initially, we used OpenAI's Generate Chat Completion activity in UiPath Studio. However, on 9 January 2024, the activity stopped working as OpenAI [deprecated the endpoint and the available models for the activity](https://platform.openai.com/docs/deprecations). To solve this issue, we applied our knowledge of making HTTP requests to OpenAI's API from our Artificial Intelligence and Machine Learning module in UiPath Studio. After setting up the HTTP request with OpenAI's GPT 3.5 Turbo model, we realised it sometimes identified unlisted roles not found in the available job positions instead of classifying it as 'Unknown'. To solve this issue, we added an 'If' condition to set the final applied job position as 'Unknown' if it is not in the available job positions. This ensured accurate classification of unavailable job positions as 'Unknown'.

Thirdly, another challenge we encountered was passing arguments from UiPath's Invoke PowerShell activity into our Python script to assign match scores of applicants' resumes based on the description for their applied job position. Before passing the arguments, we learned that we had to preprocess the resume content and job description to remove all newline characters to ensure the script would run properly. We also learned that we can access positional arguments from PowerShell within Python using sys.argv. Applying the knowledge learned, we successfully ran our resume scorer Python script using UiPath's Invoke PowerShell activity.

Fourthly, another challenge we experienced was getting accurate extraction of details from an applicant's resume, as every resume is formatted differently. However, after conducting our research, we discovered that we could use the generative classifier in UiPath's Extract Document Data activity to obtain applicant details from resumes precisely and reliably. By referring to UiPath's guide on crafting precise prompts for optimal AI responses, we could consistently extract applicant details correctly from resumes.

Fifthly, another challenge we faced was the loss of data during application processing. Initially, our program only writes the applicant details to an Excel file after completing the application processing for every applicant. Since we used the free tier for the generative AI activities, we were limited to 50 API calls per hour. During testing, when we repeatedly ran the program and exceeded the limits, we realised that we lost applicant data if we exceeded the limit while the program ran halfway. To solve this issue, we switched to batch processing by modifying our program to initialise a job application Excel file with the date and time in the beginning, write and sort data in this file after processing each applicant's email before moving it to their respective applied job position folder in Outlook. This modification ensures that our program does not lose applicant data when exceeding the API call limits while running halfway. If our updated program exceeds the API call limits, our users are still able to continue processing the other email applications after the limits have cooled down.

Lastly, the challenge we faced was activating our Excel hyperlink formula. The purpose of the hyperlink is to allow company staff assessing shortlisted applicants to conveniently access the applicant's resume by clicking the link. If we immediately activate the Excel hyperlink formula after processing each applicant, we will lose the formula when updating the Excel file. This issue happens because UiPath's read range activity only reads the displayed cell value instead of the entire formula. To solve this issue, we used the write range workbook activity after processing each applicant, as it maintains the formula value by writing it as a string. After processing all applicants, we activated the Excel formula using the Excel write cell formula. This approach enabled the link to every applicant's resume in the Excel file to work after the program ends.

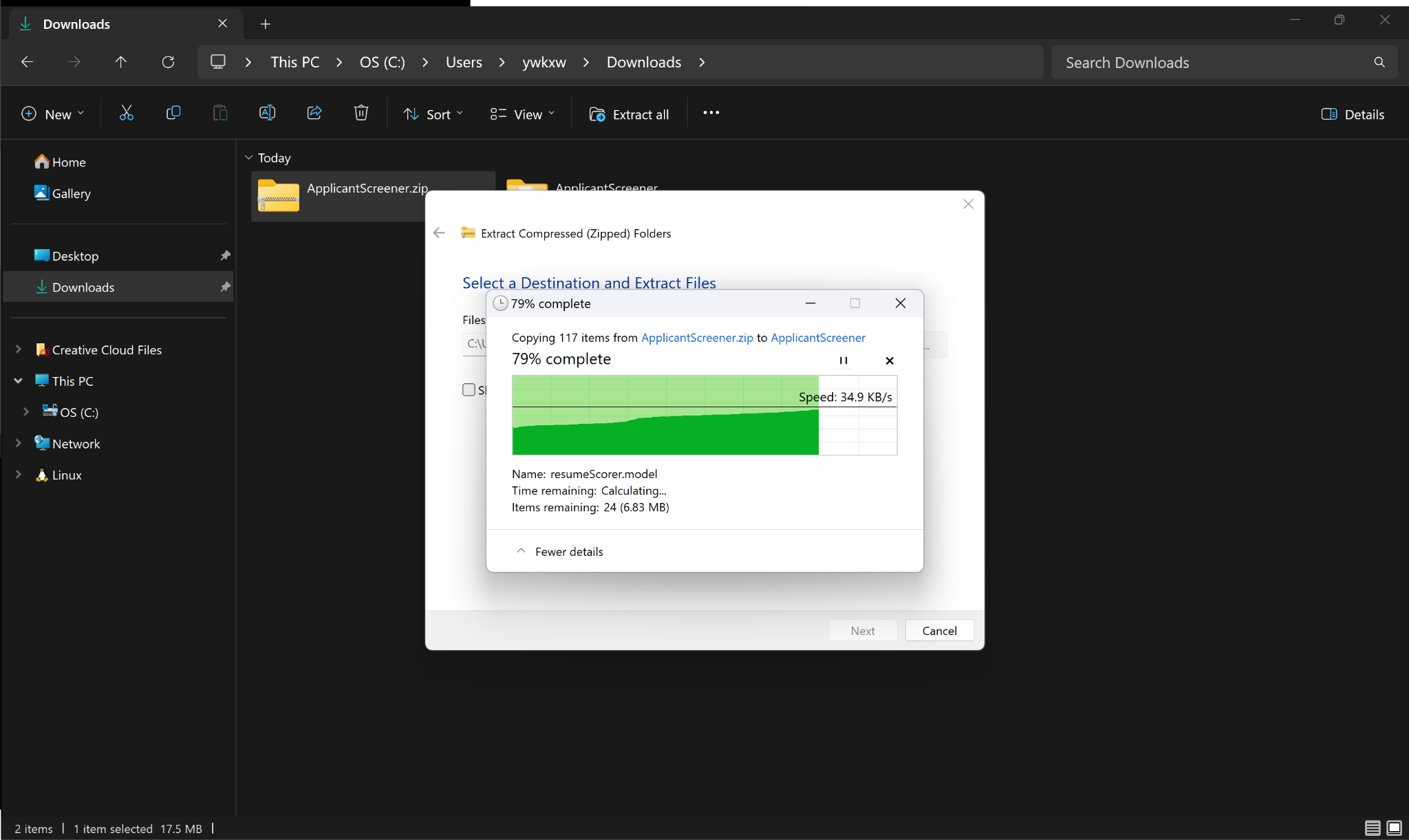
**Conclusion**

In conclusion, our excellent teamwork, adaptability, and problem-solving abilities allowed us to successfully integrate AI with automation to develop the Applicant Screener app in UiPath Studio. By going through an iterative development process to reduce errors and increase our app's reliability, we have created a dependable applicant screening tool that benefits organisations by allowing them to experience greater efficiency, lower costs, reduced hiring time, and better-informed decisions in hiring applicants.

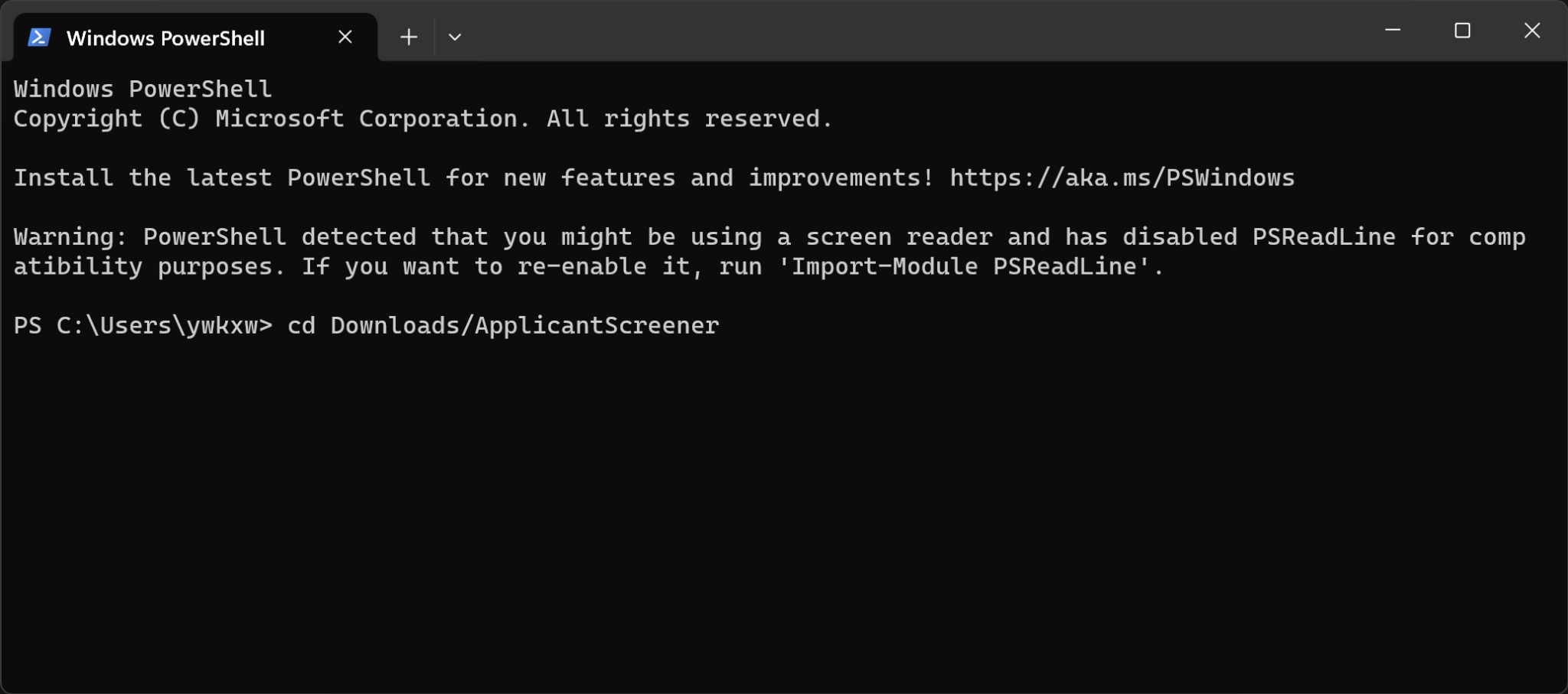
**Steps for Testing the Applicant Screener App**

*Note: Ensure that python and pip are installed on the device running the app, and scripts are also allowed to run on the device.*

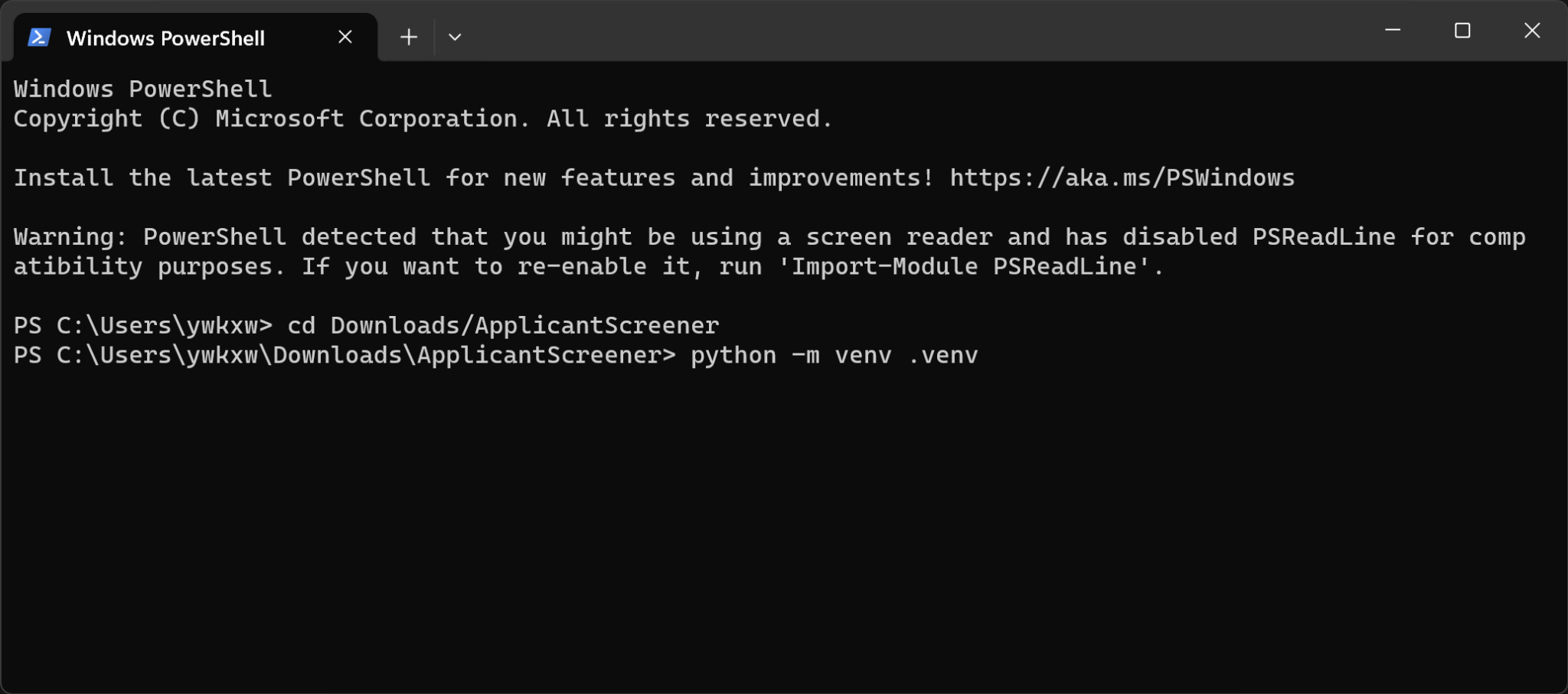
1. Download and unzip the zip file for the Applicant Screener App.



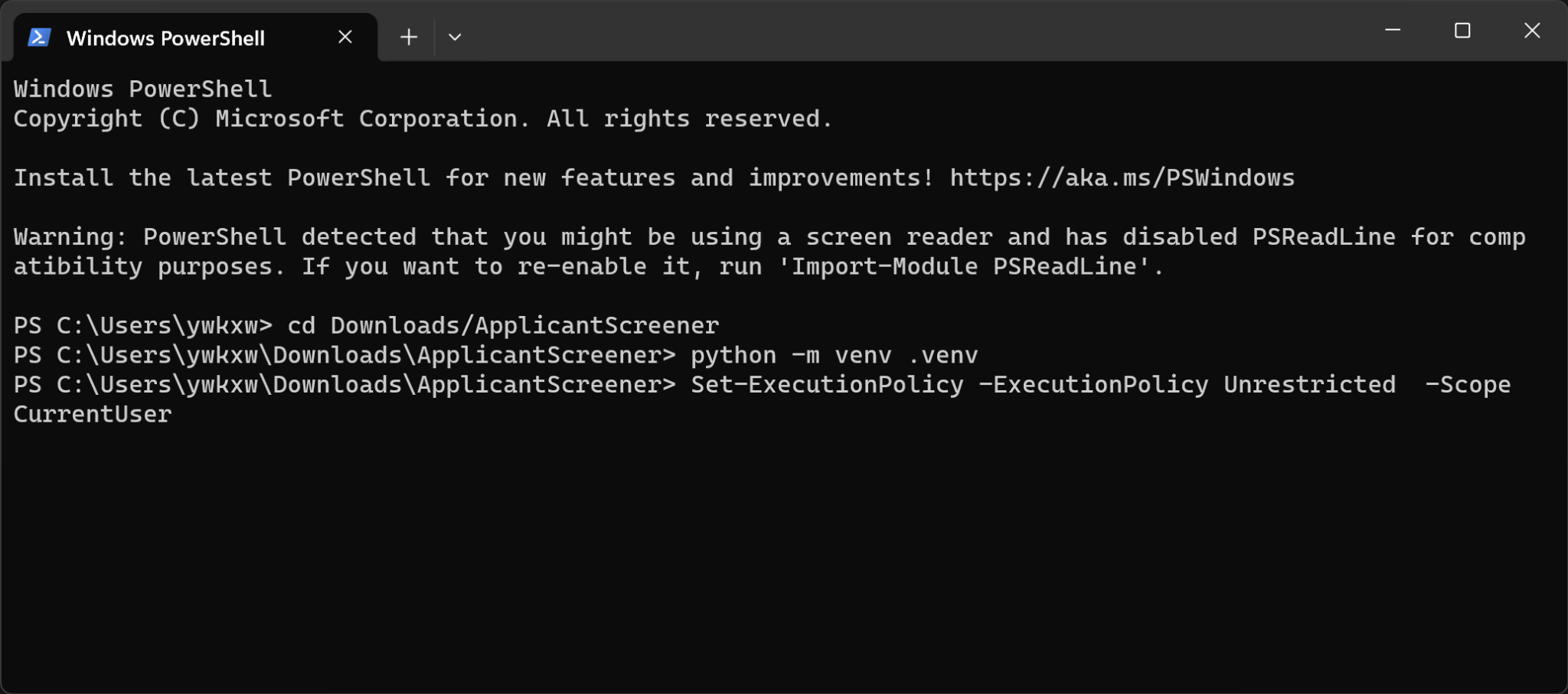
1. Open a PowerShell terminal and change the working directory to the Applicant Screener App directory using “cd [Applicant Screener App Directory]”.



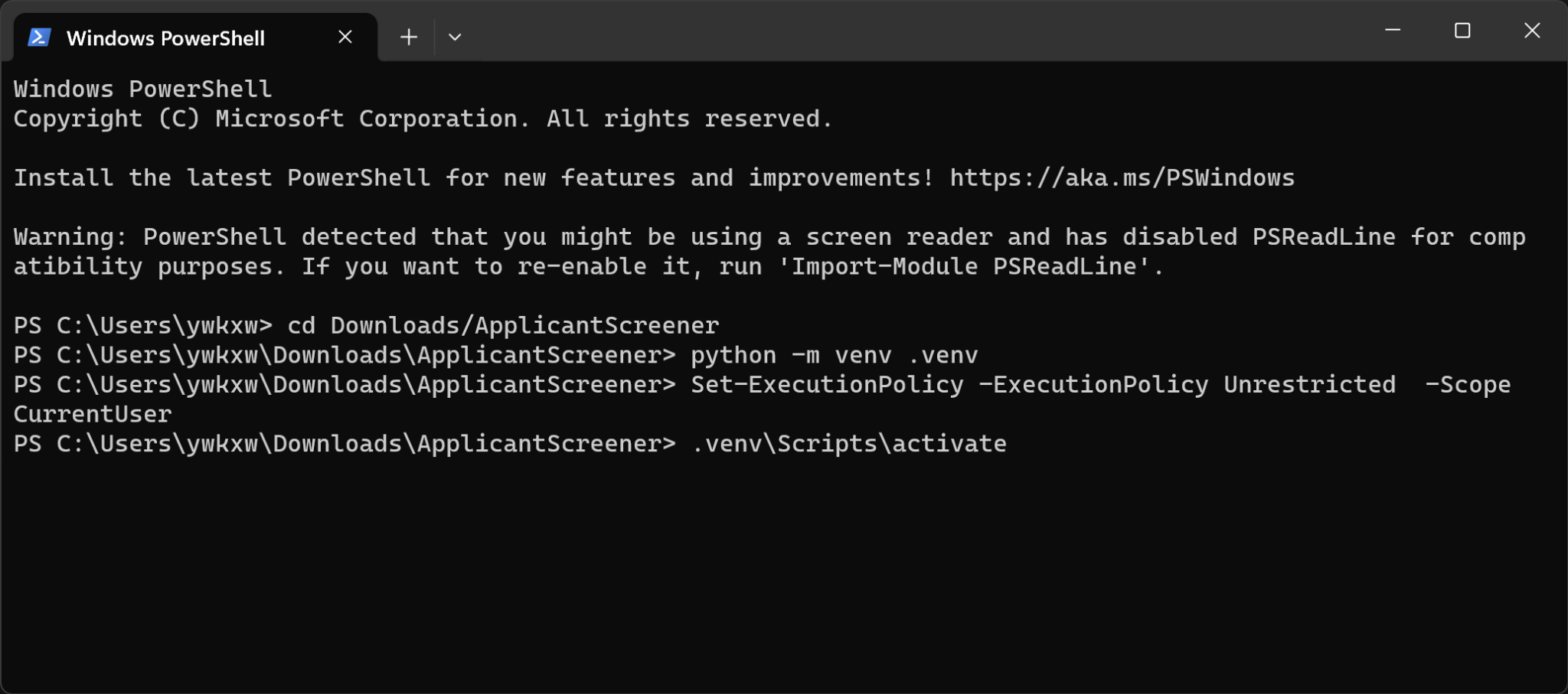
1. Create the Python virtual environment using “python -m venv .venv”.



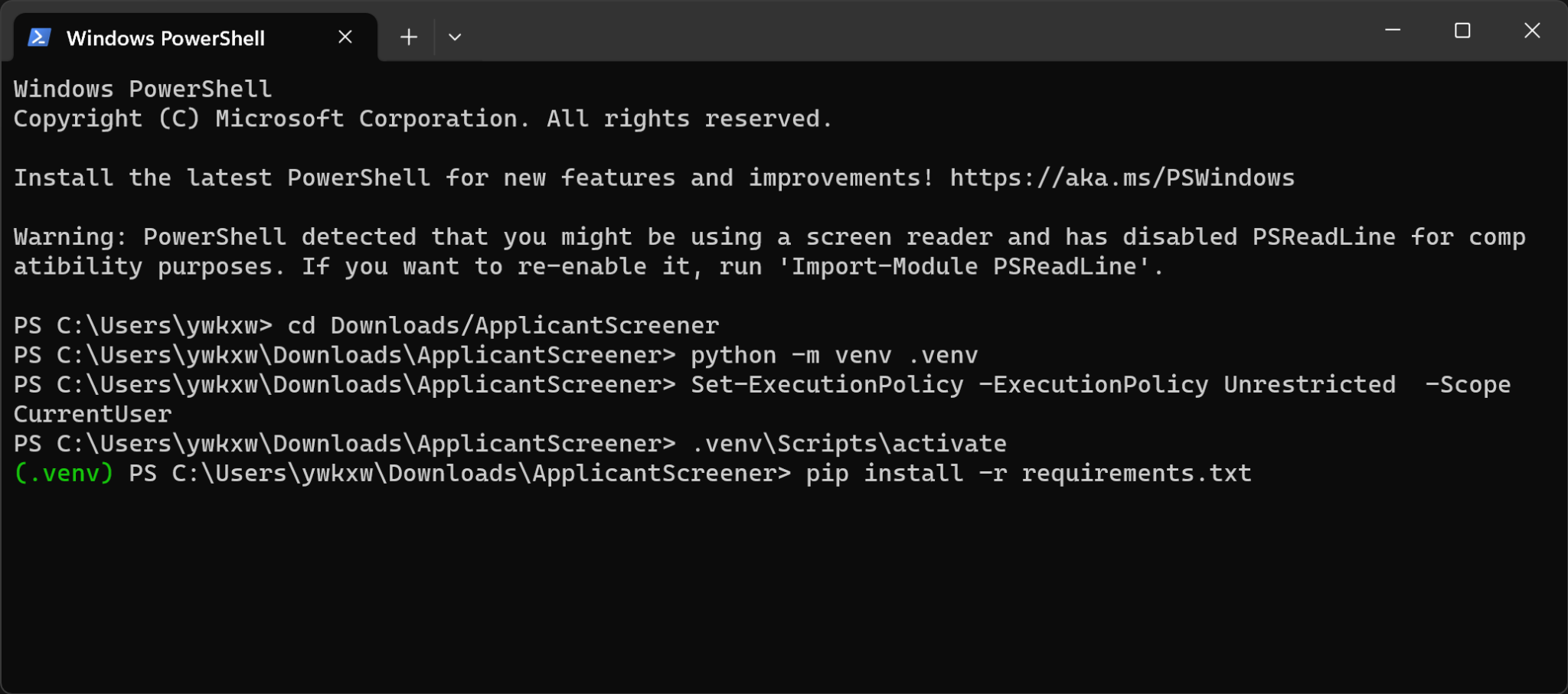
1. Set the execution policy of the current user to unrestricted using “Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser”.



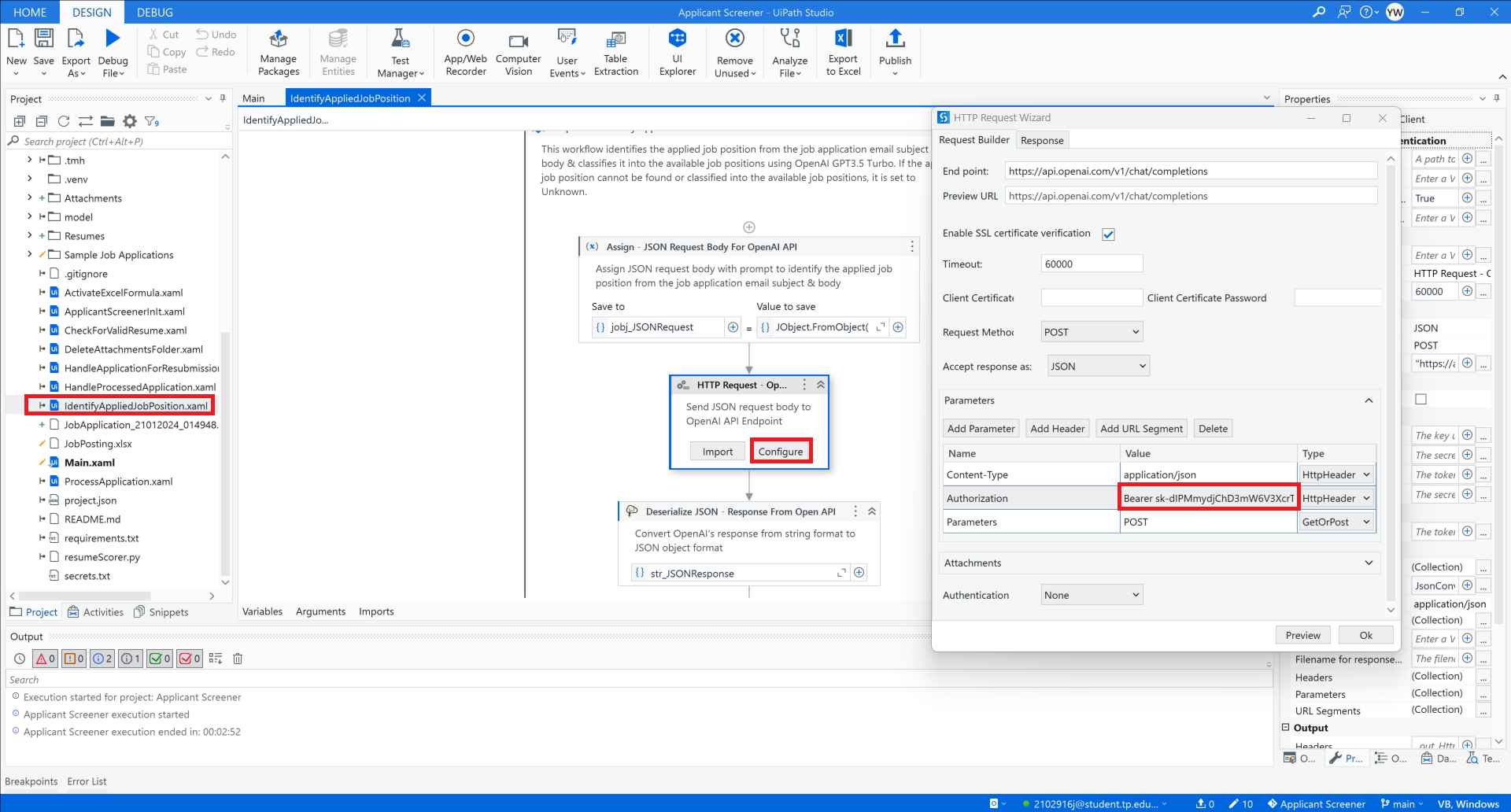
1. Activate the virtual environment using “.venv\Scripts\activate”.



1. Install the required libraries into the virtual environment “pip install -r requirements.txt”.



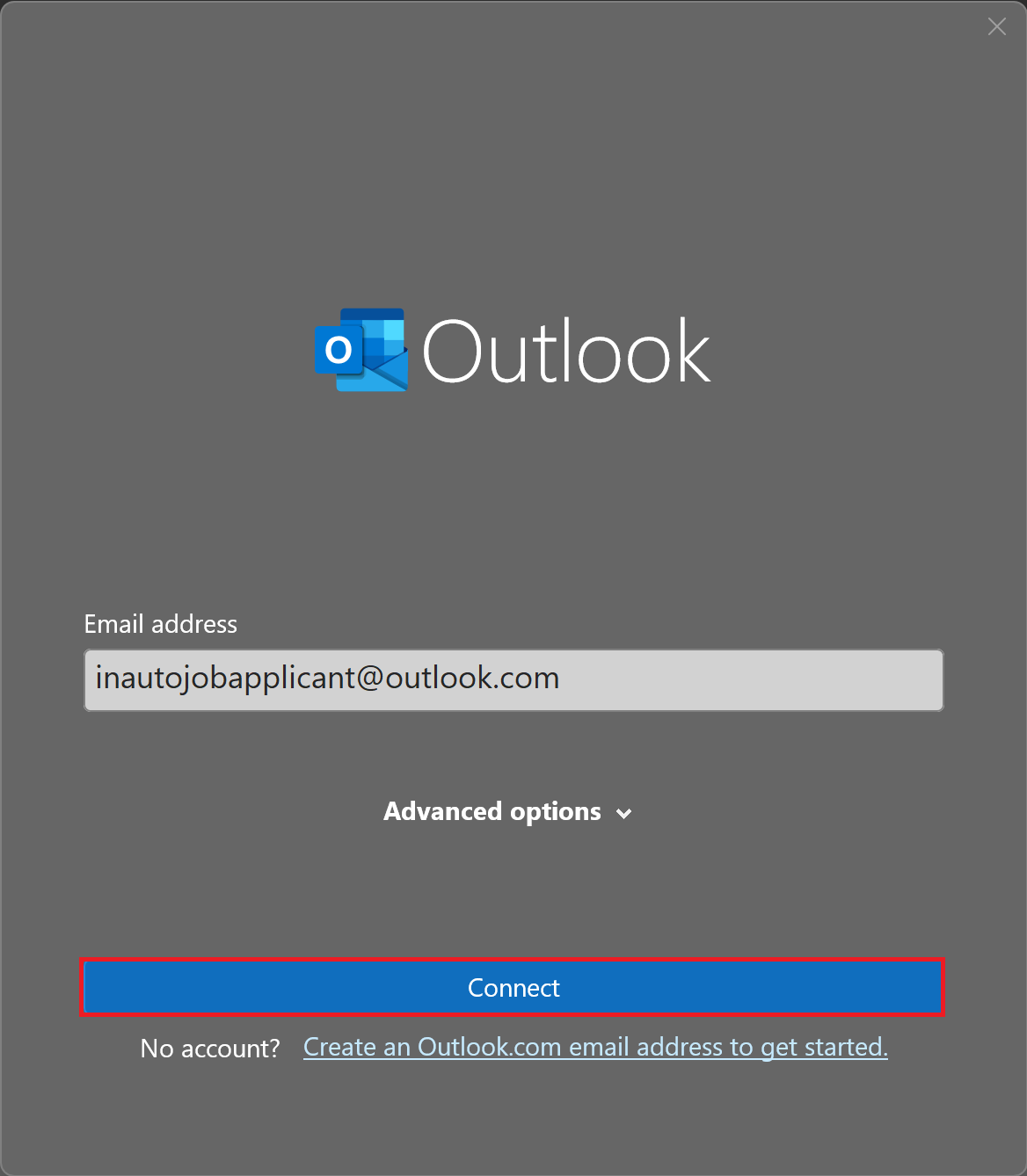
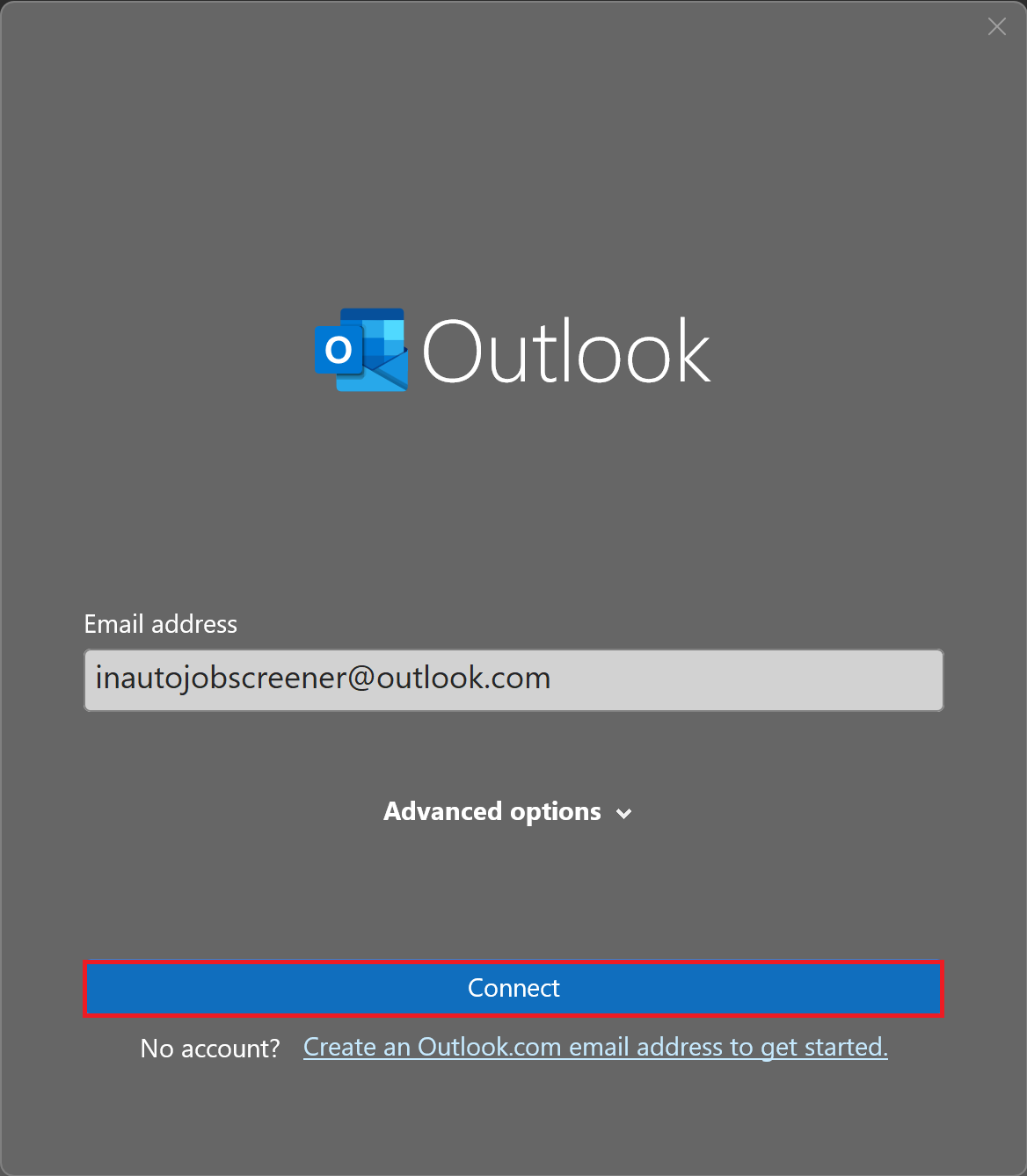
1. **[Only Needed After March 2024 When the OpenAI API Key Expires]** Open the 'IdentifyAppliedJobPosition.xaml' workflow in UiPath Studio, click on the configure option of the HTTP Request activity, and replace the Authorization parameter with “Bearer [Your OpenAI API Key]”.



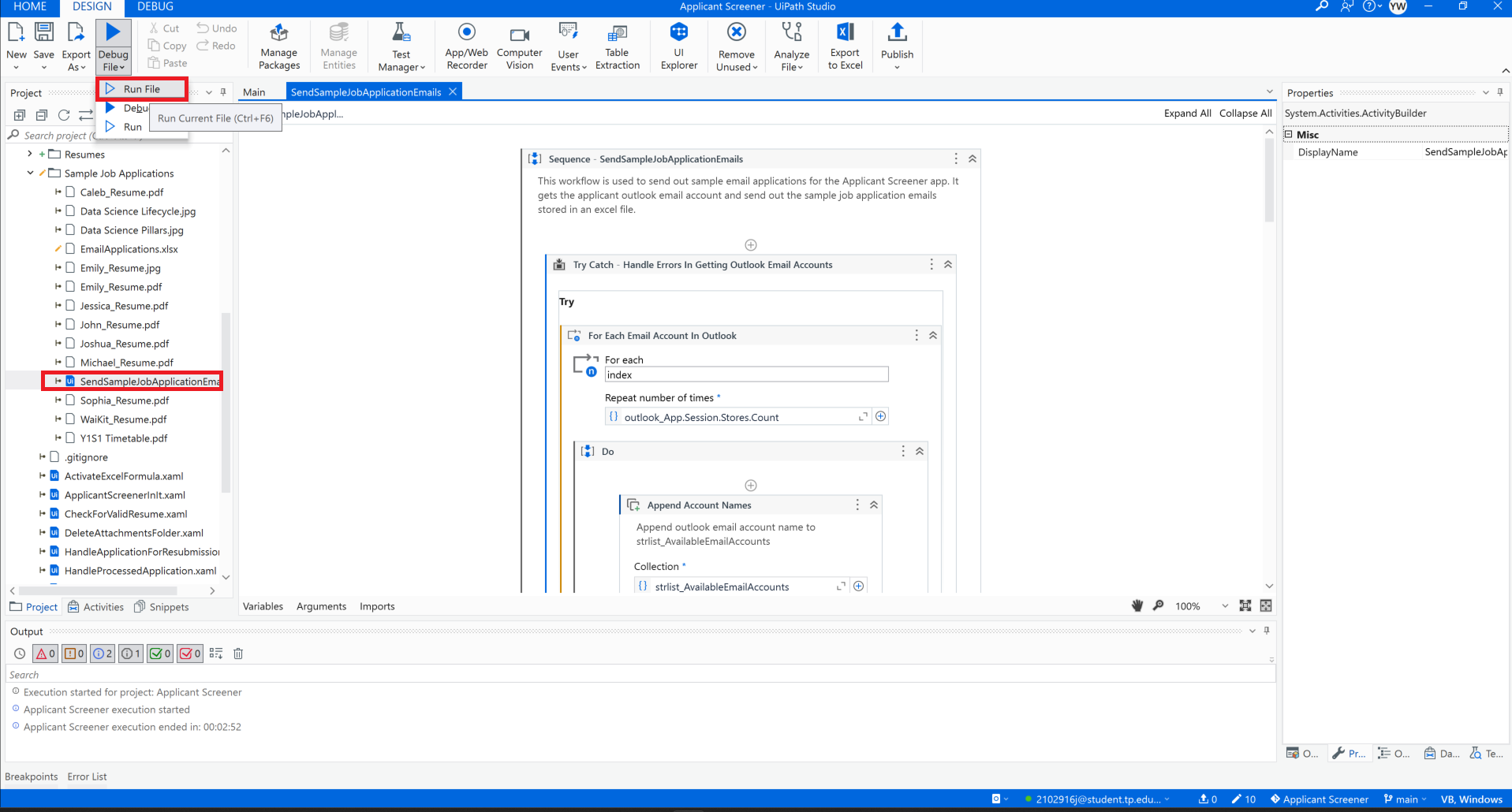
1. Sign in to Orchestrator in UiPath Studio



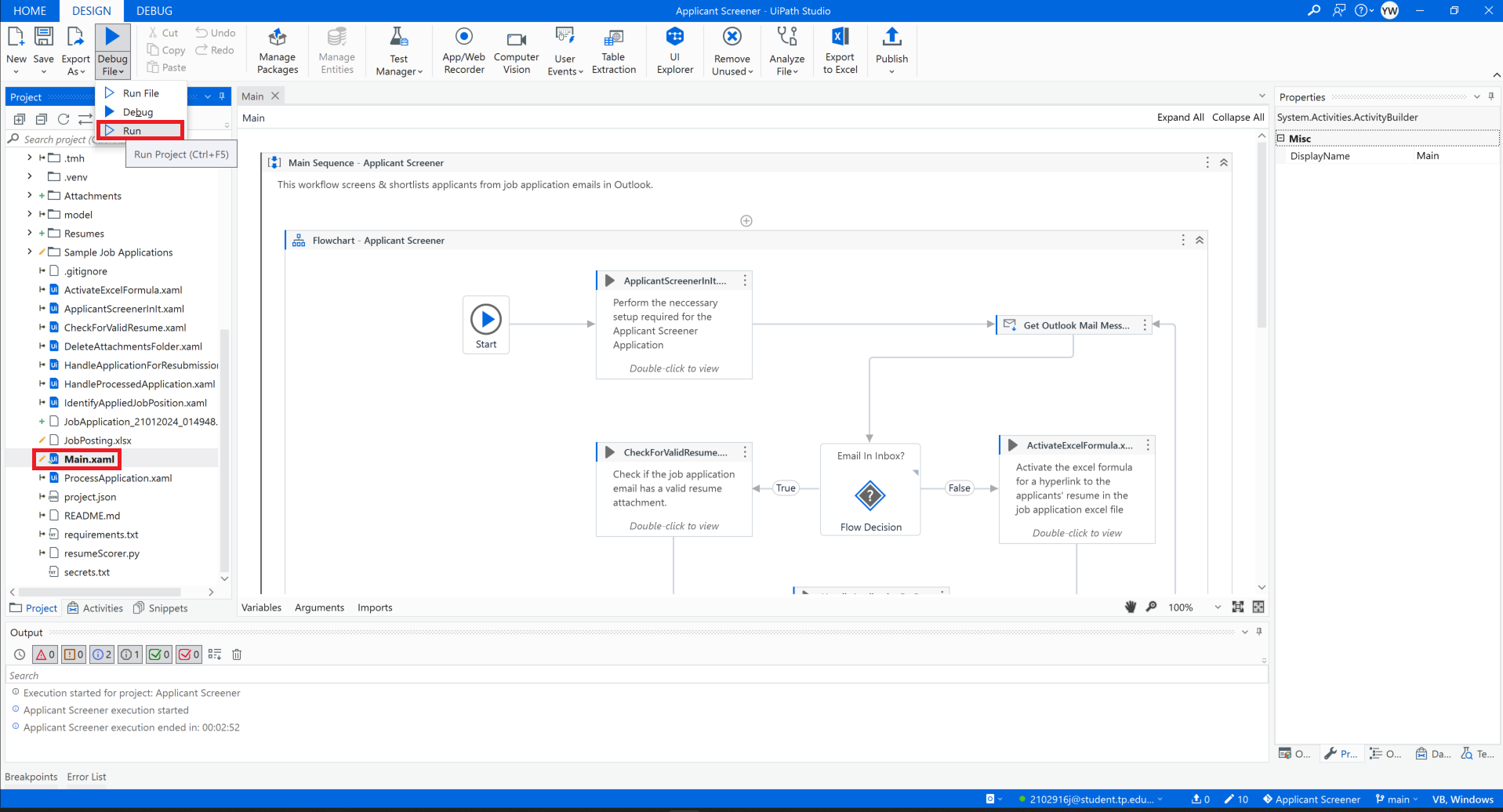
1. Add the [inautojobapplicant@outlook.com](mailto:inautojobapplicant@outlook.com) and [inautojobscreener@outlook.com](mailto:inautojobscreener@outlook.com) email account to the device’s Microsoft 365 Outlook app. The password can be found in ‘secrets.txt’.

1. Run the 'SendSampleJobApplicationEmails.xaml' workflow in in the 'Sample Job Applications' folder in UiPath Studio



1. Run the 'Main.xaml' workflow in UiPath Studio



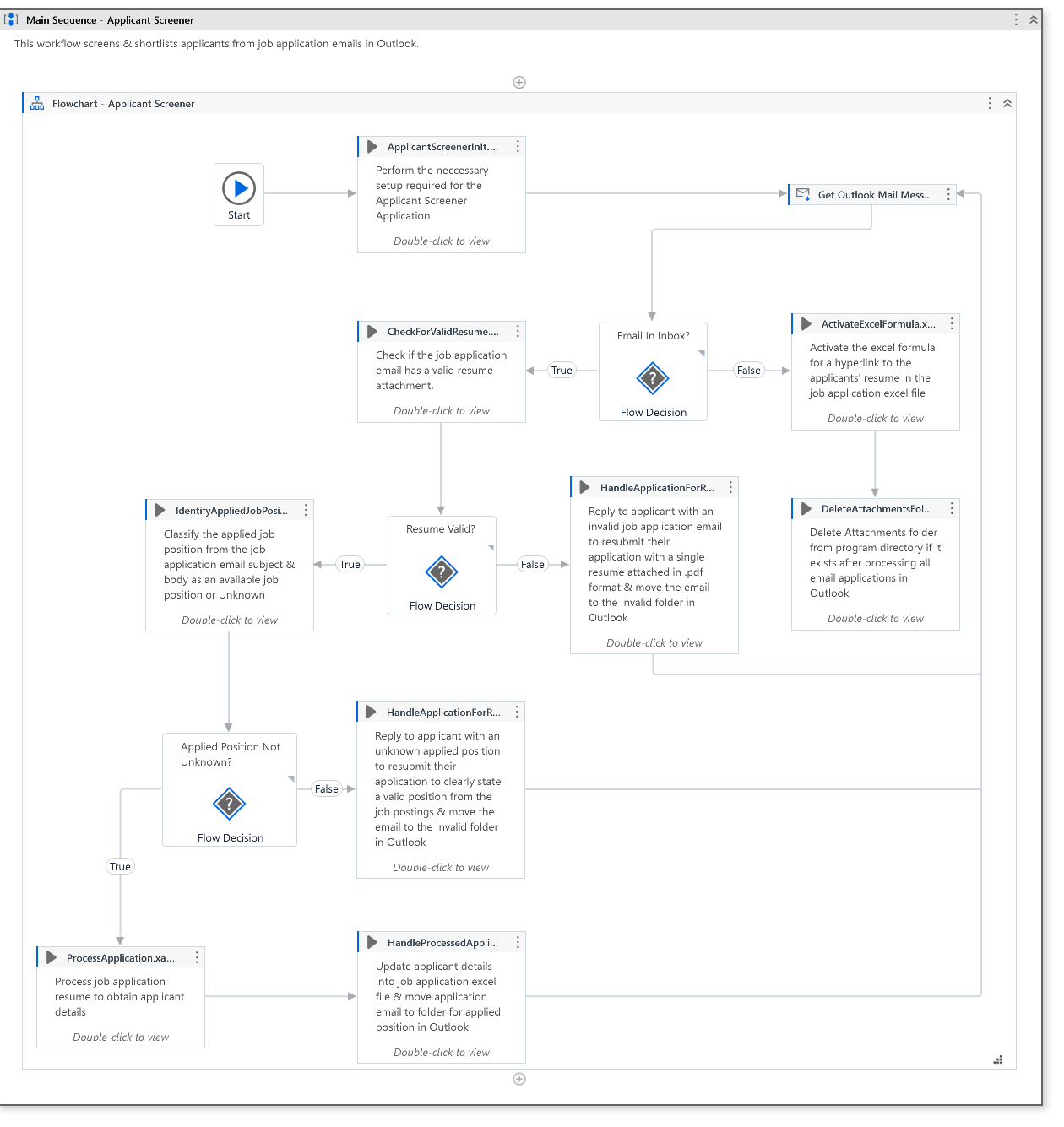
*Please refer to the demonstration video on how the Applicant Screener App should run.*

**Github Repository Link for the Applicant Screener App:**

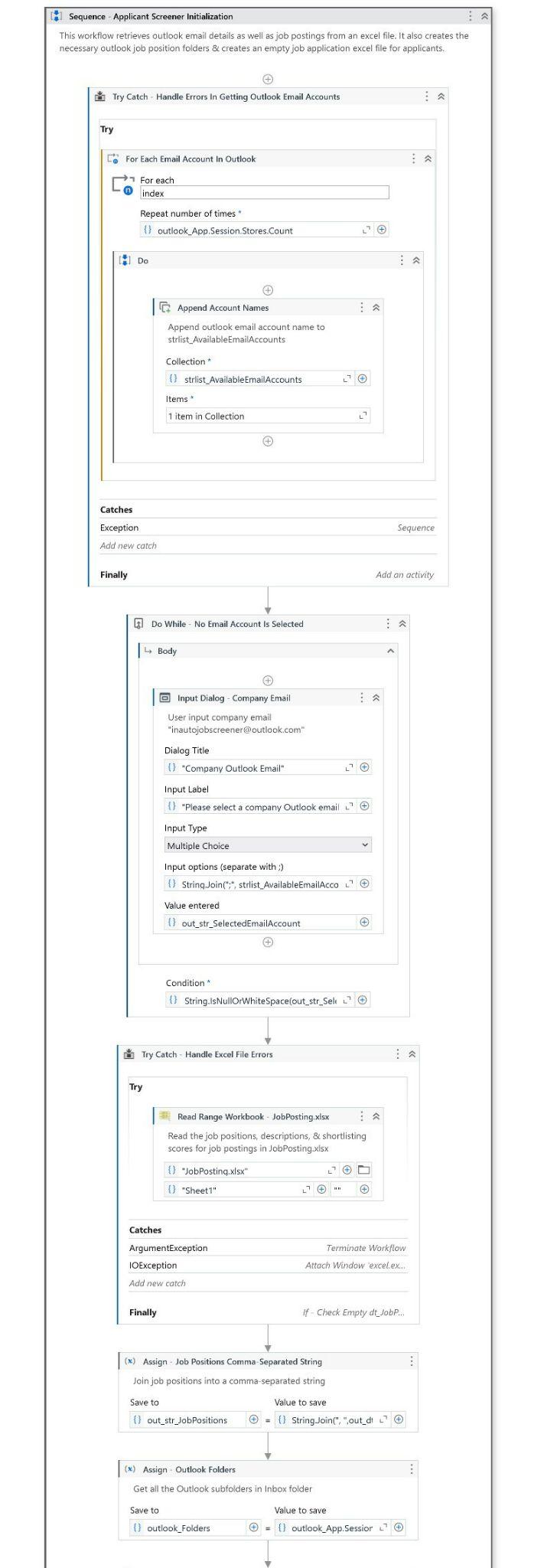
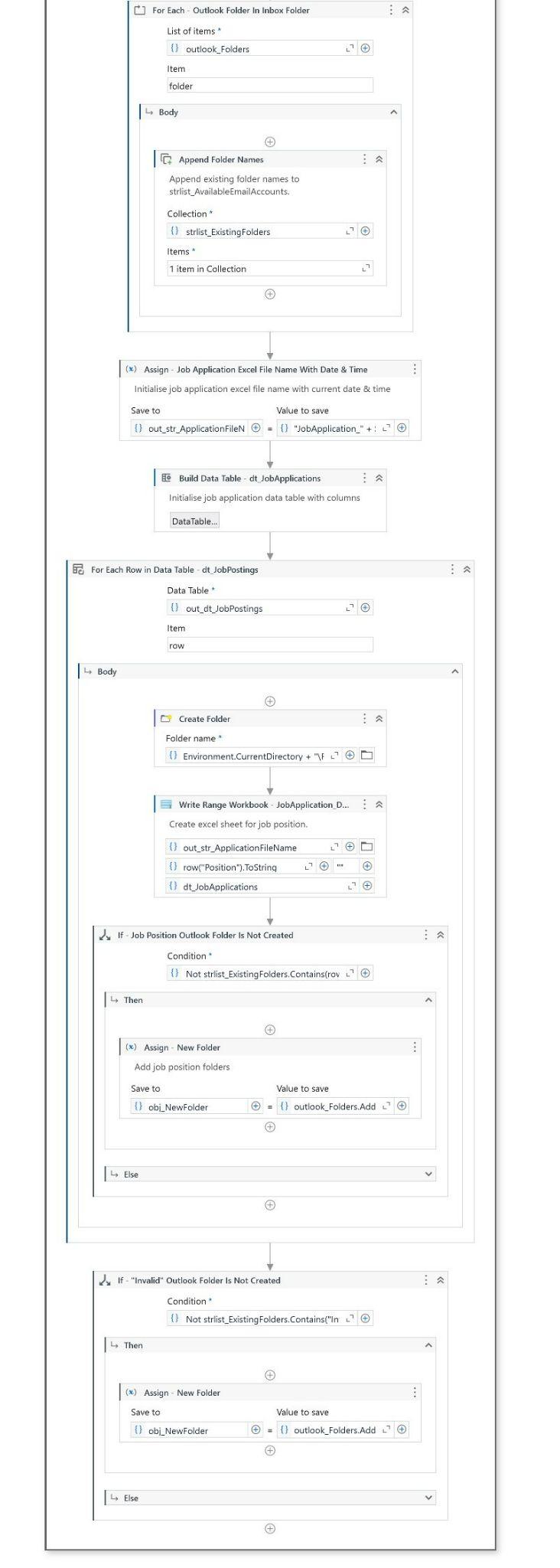
<https://github.com/YoongWK/ApplicantScreener>

**Appendix**

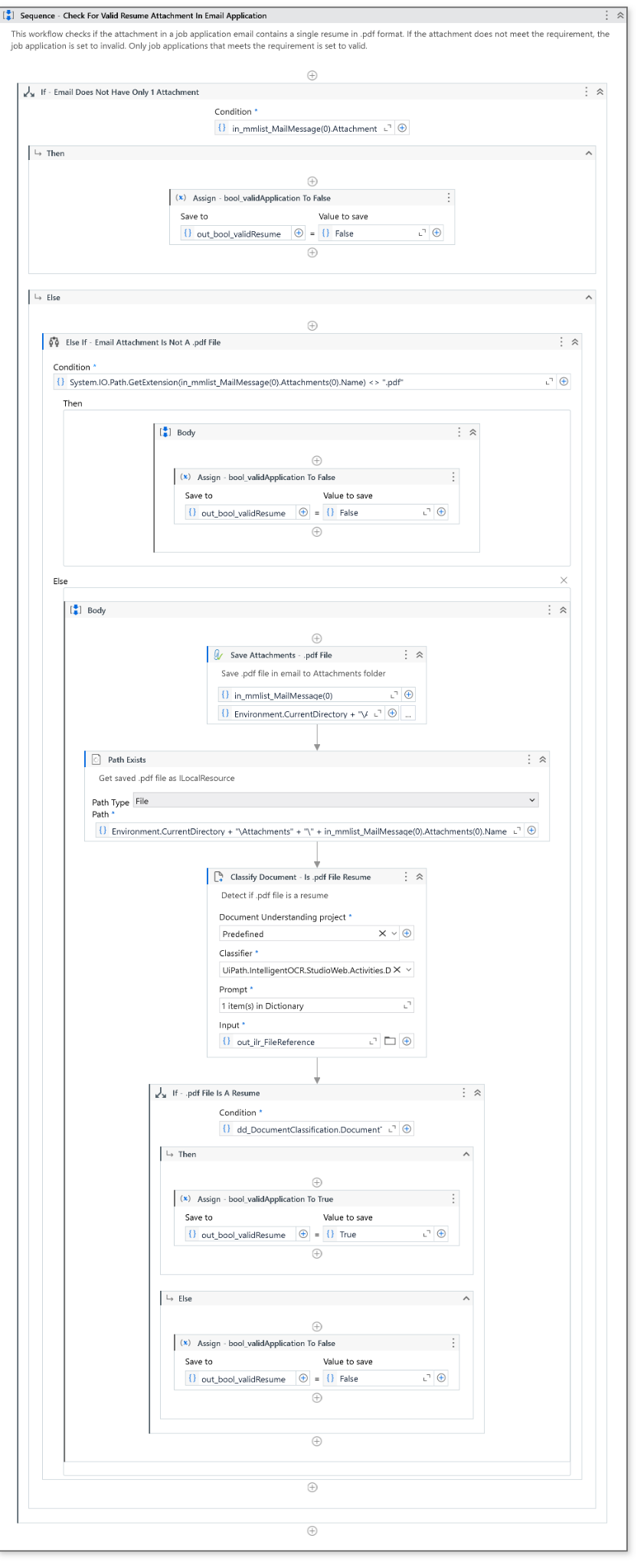
**Main.xaml Workflow**



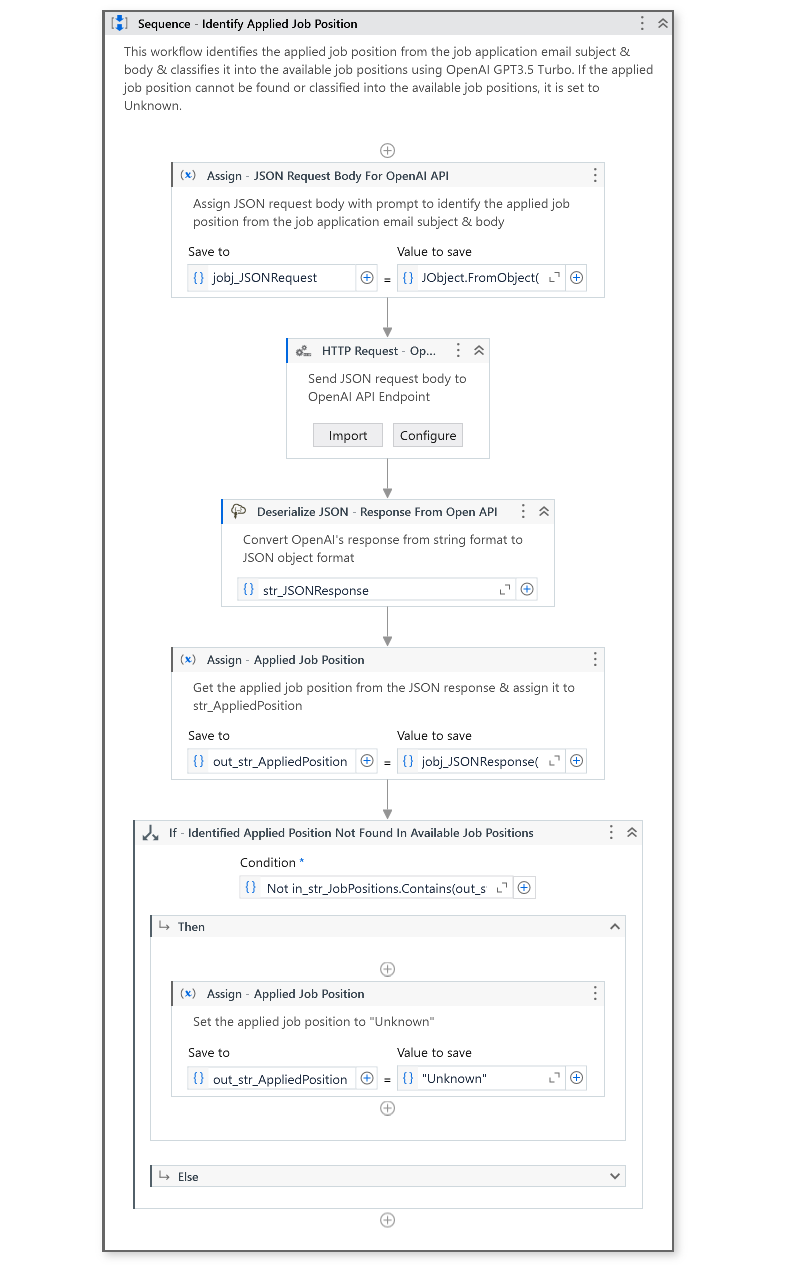
**ApplicantScreenerInIt.xaml Workflow**

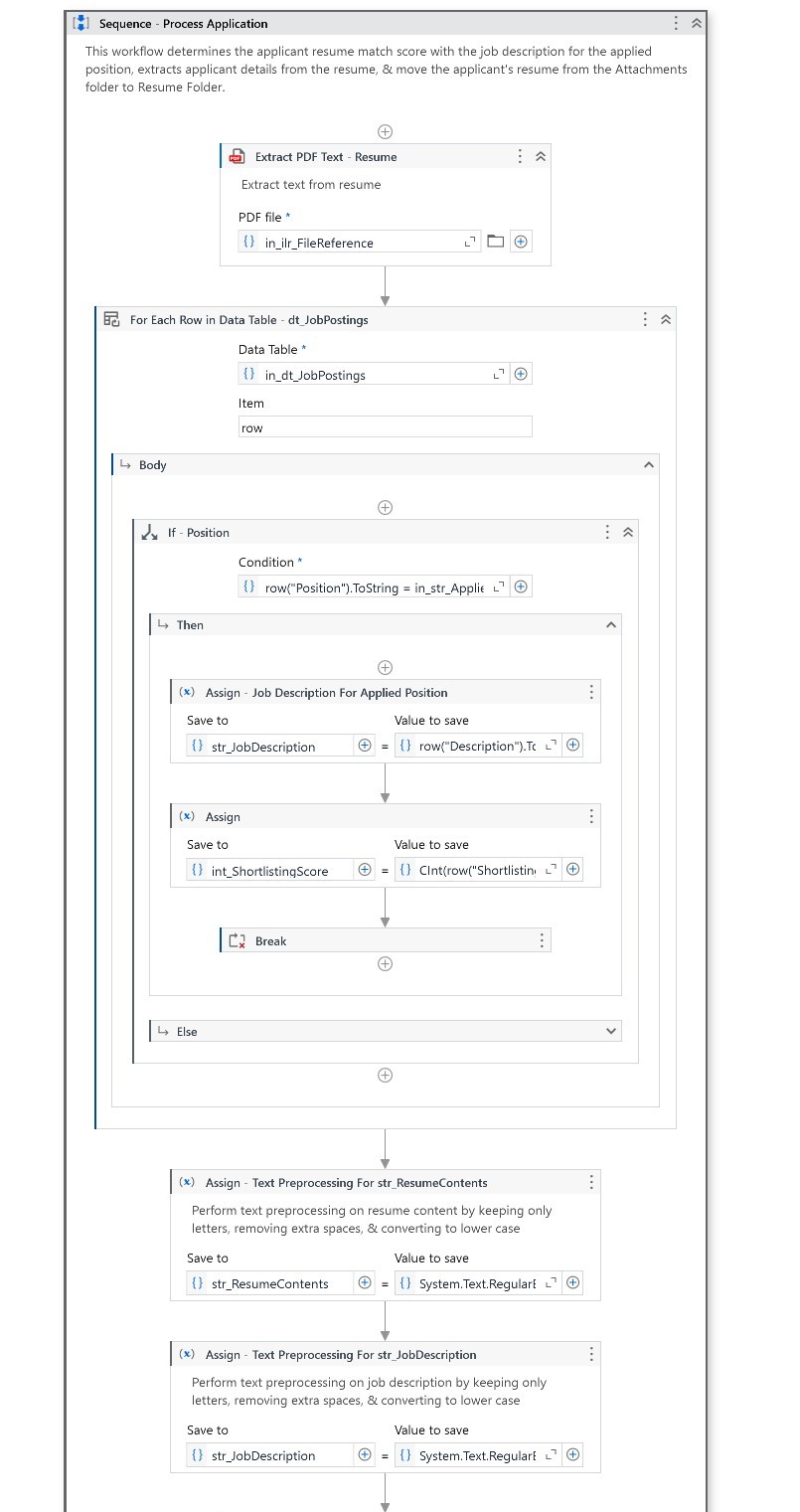
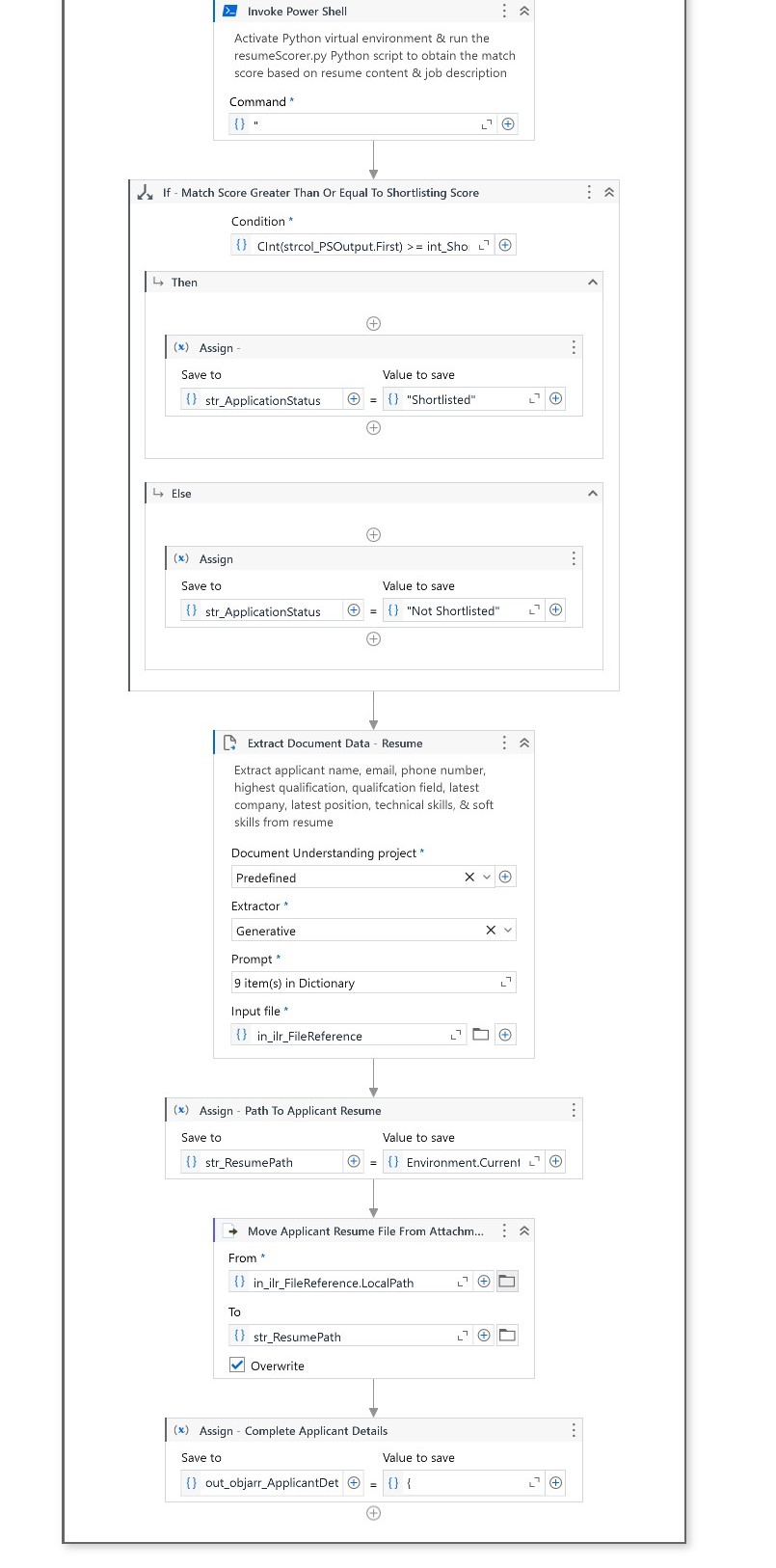
**CheckForValidResume.xaml Workflow**



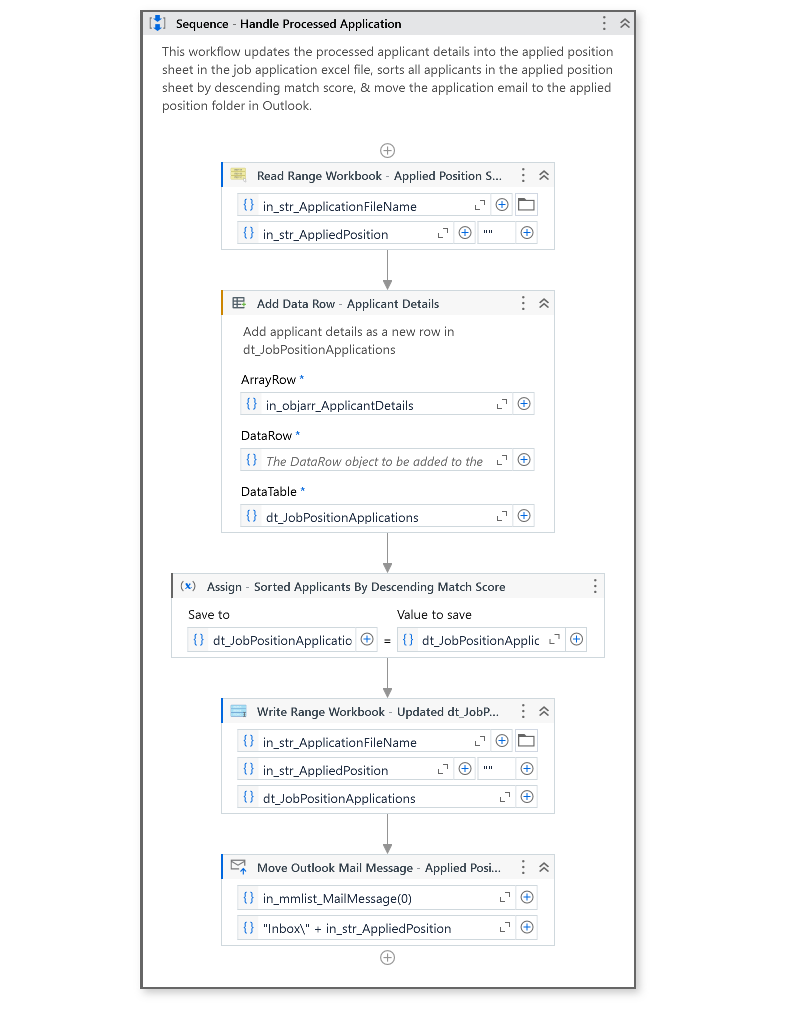
**IdentifyAppliedJobPosition.xaml Workflow**



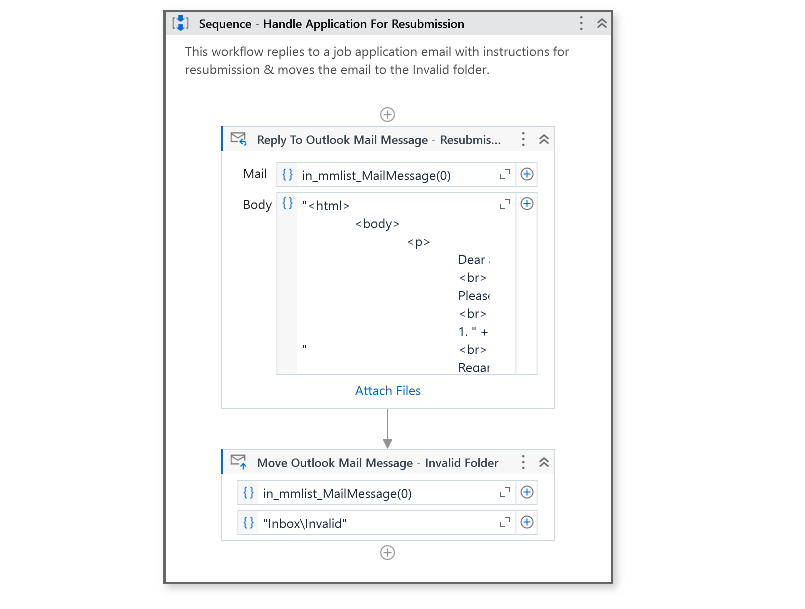
**ProcessApplication.xaml Workflow**

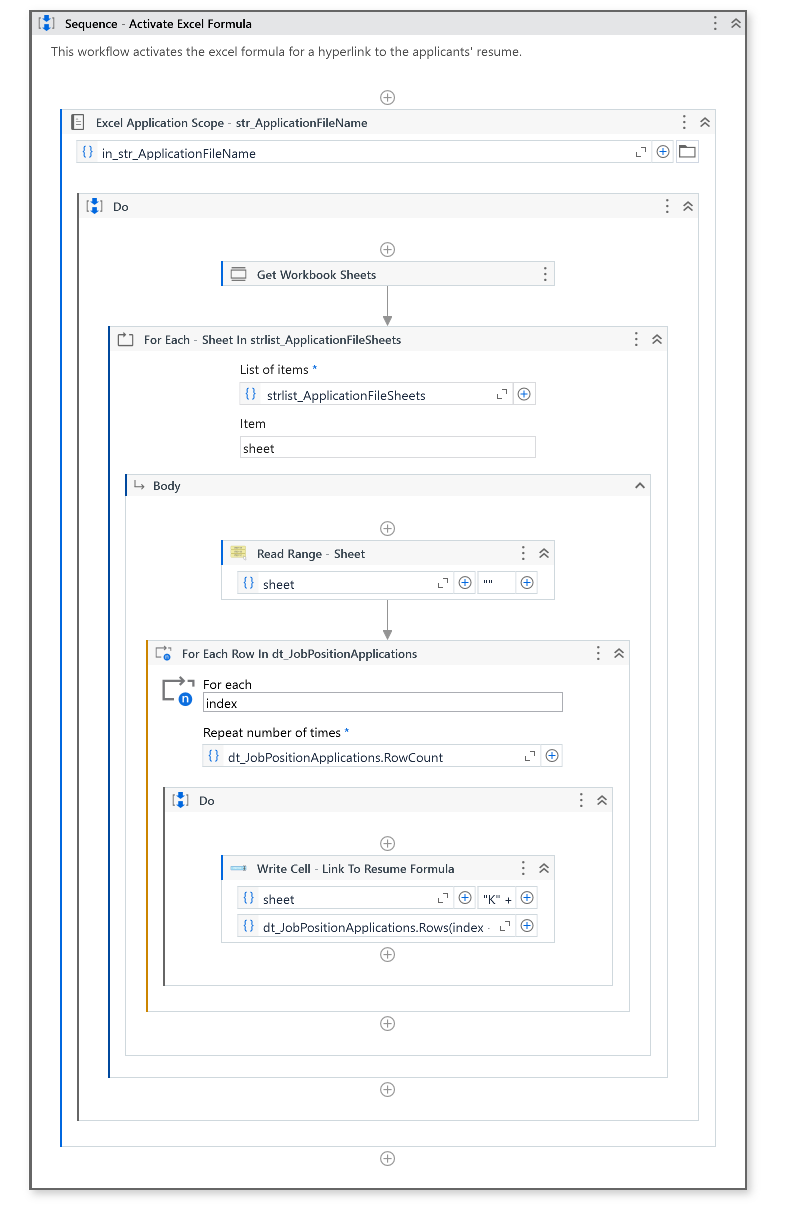
**HandleProcessedApplication.xaml Workflow**



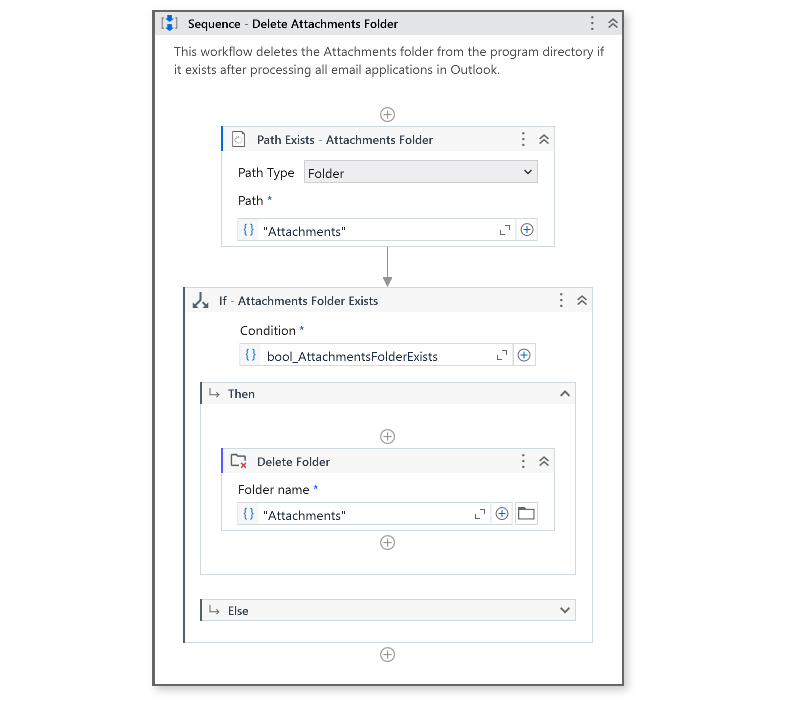
**HandleApplicationForResubmission.xaml Workflow**



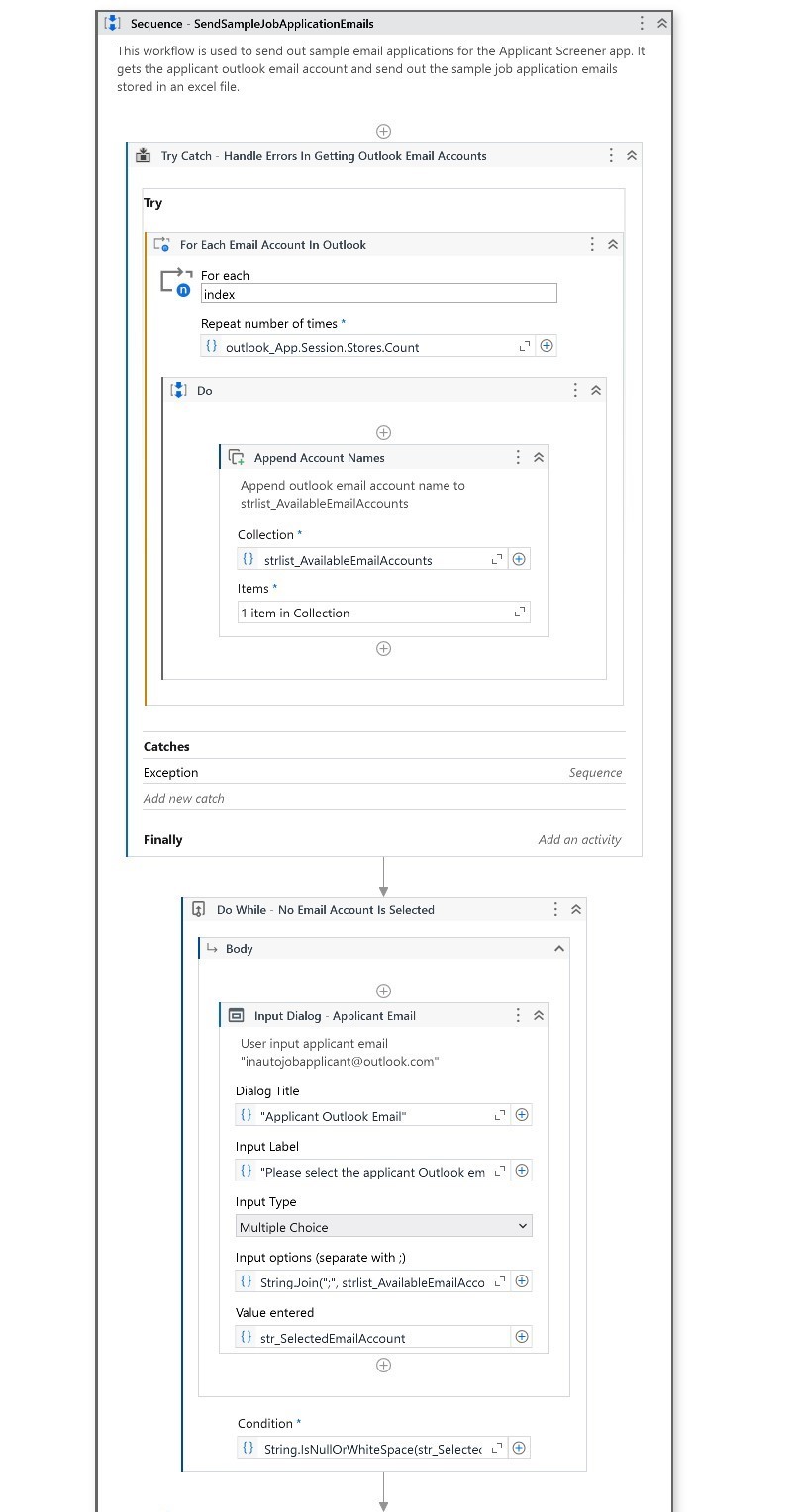
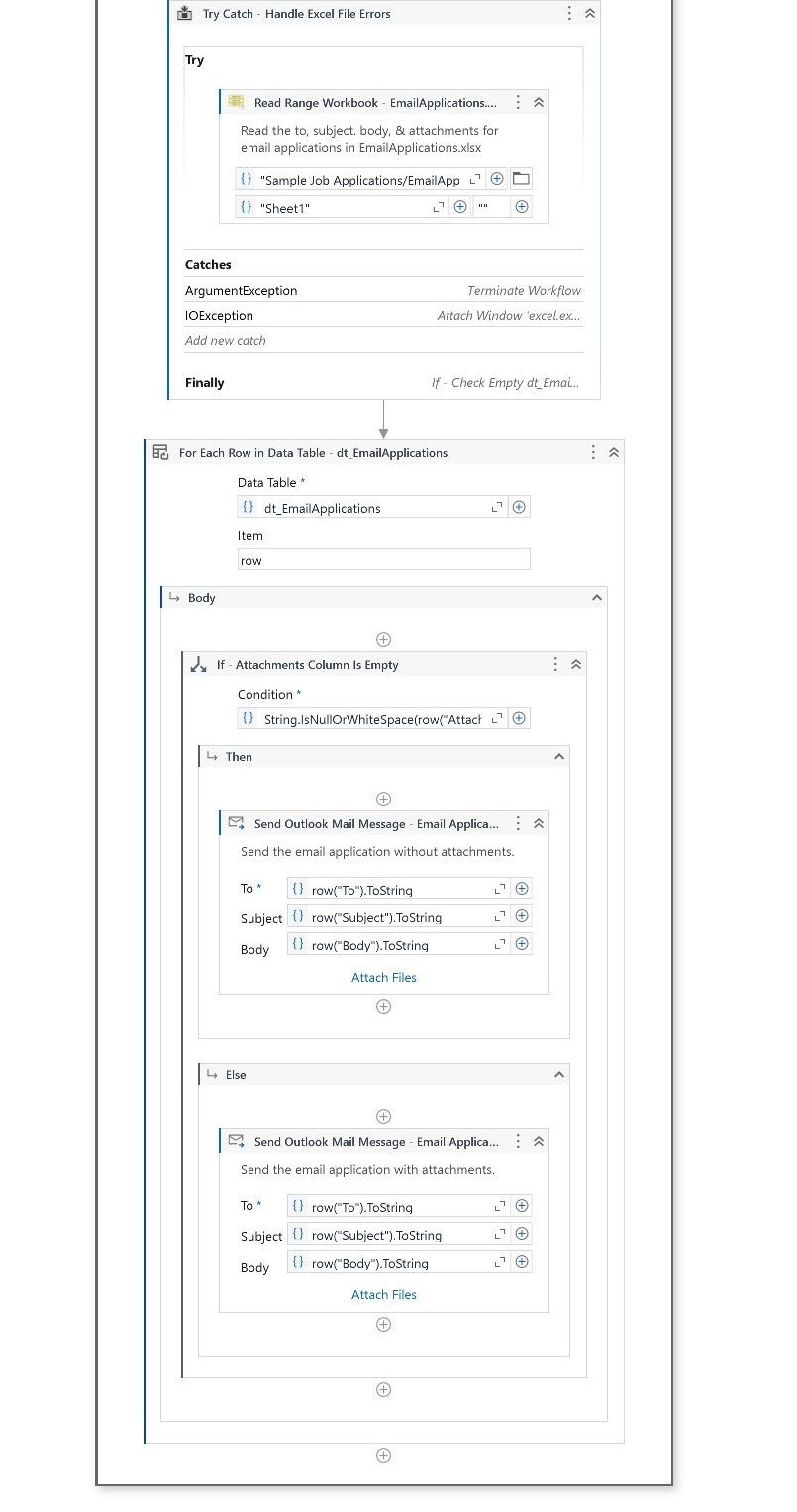
**ActivateExcelFormula.xaml Workflow**



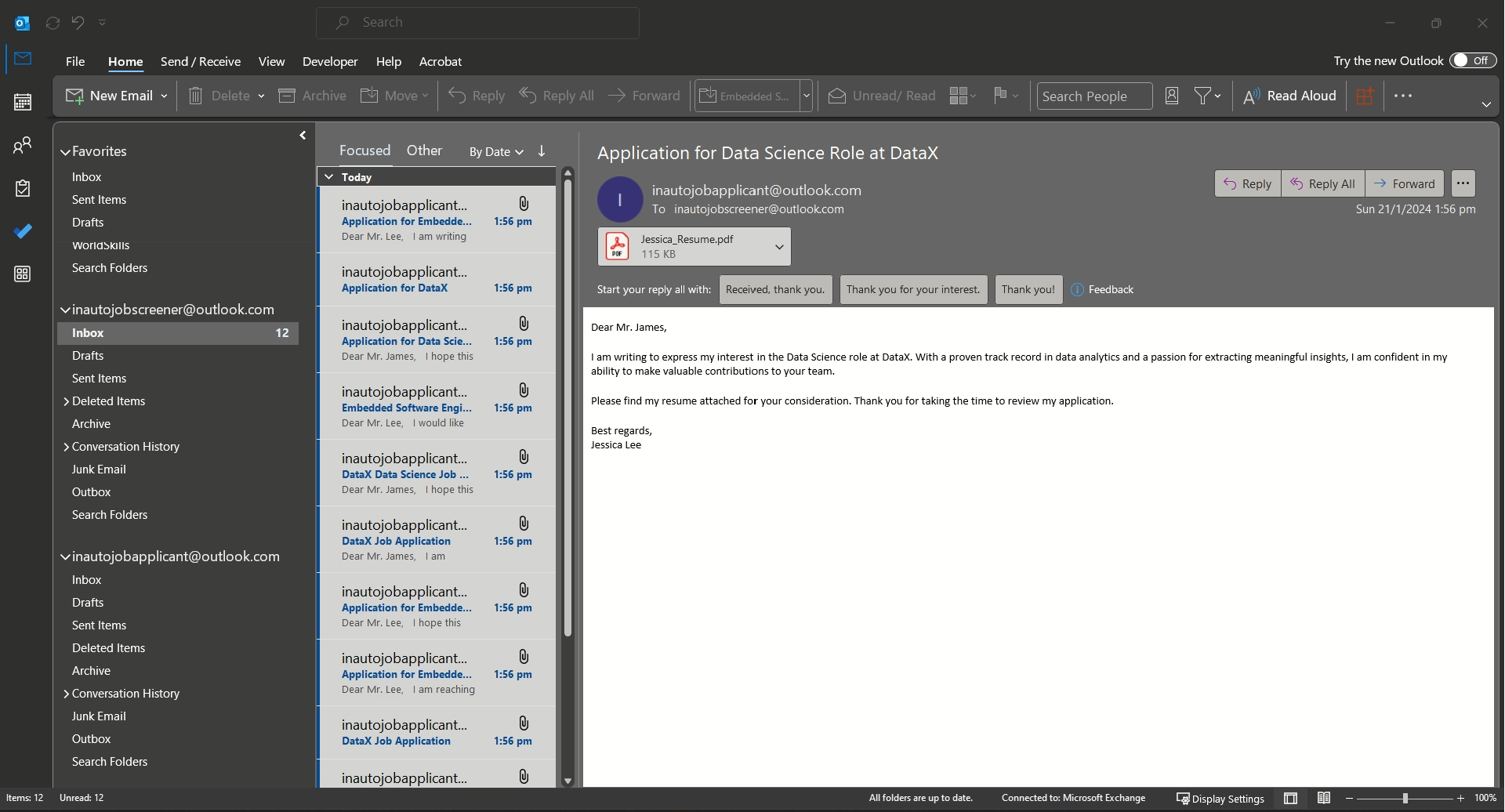
**DeleteAttachmentsFolder.xaml Workflow**



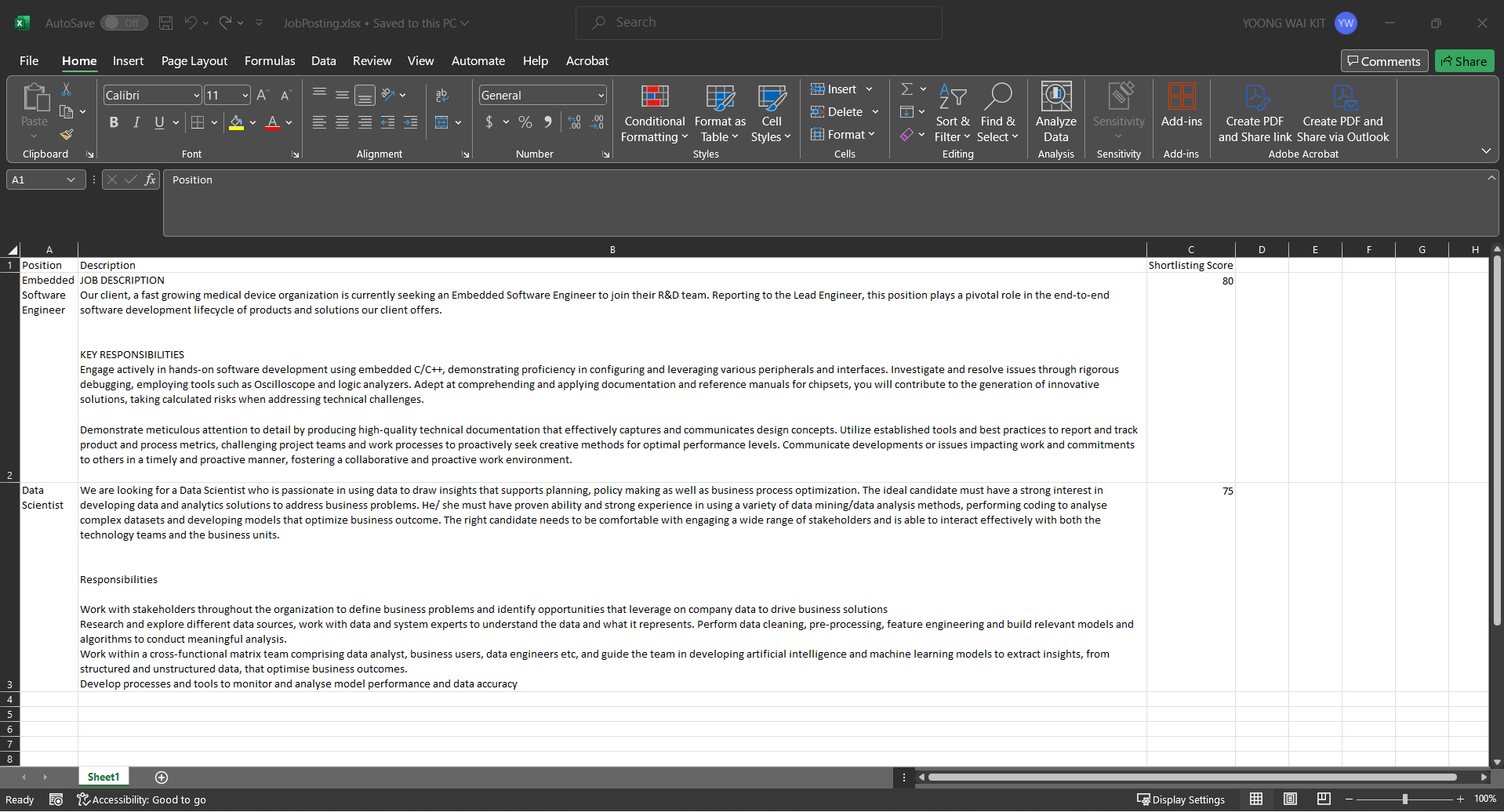
**SendSampleJobApplicationEmails.xaml Workflow (For Easy Testing)**

**Job Application Emails in Company Outlook Account (Input)**



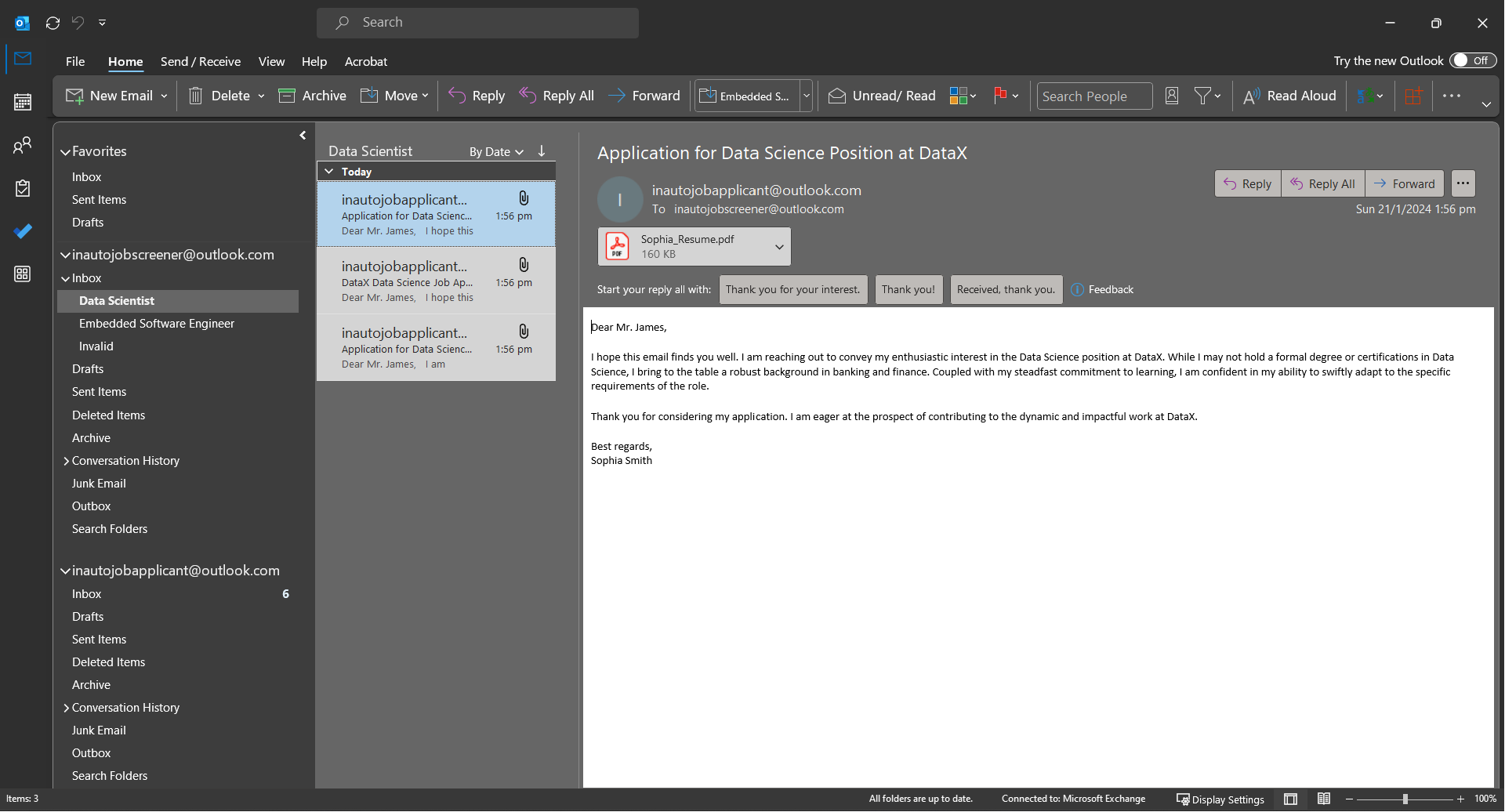
**JobPosting.xlsx File with Job Posting Details of the Company (Input)**



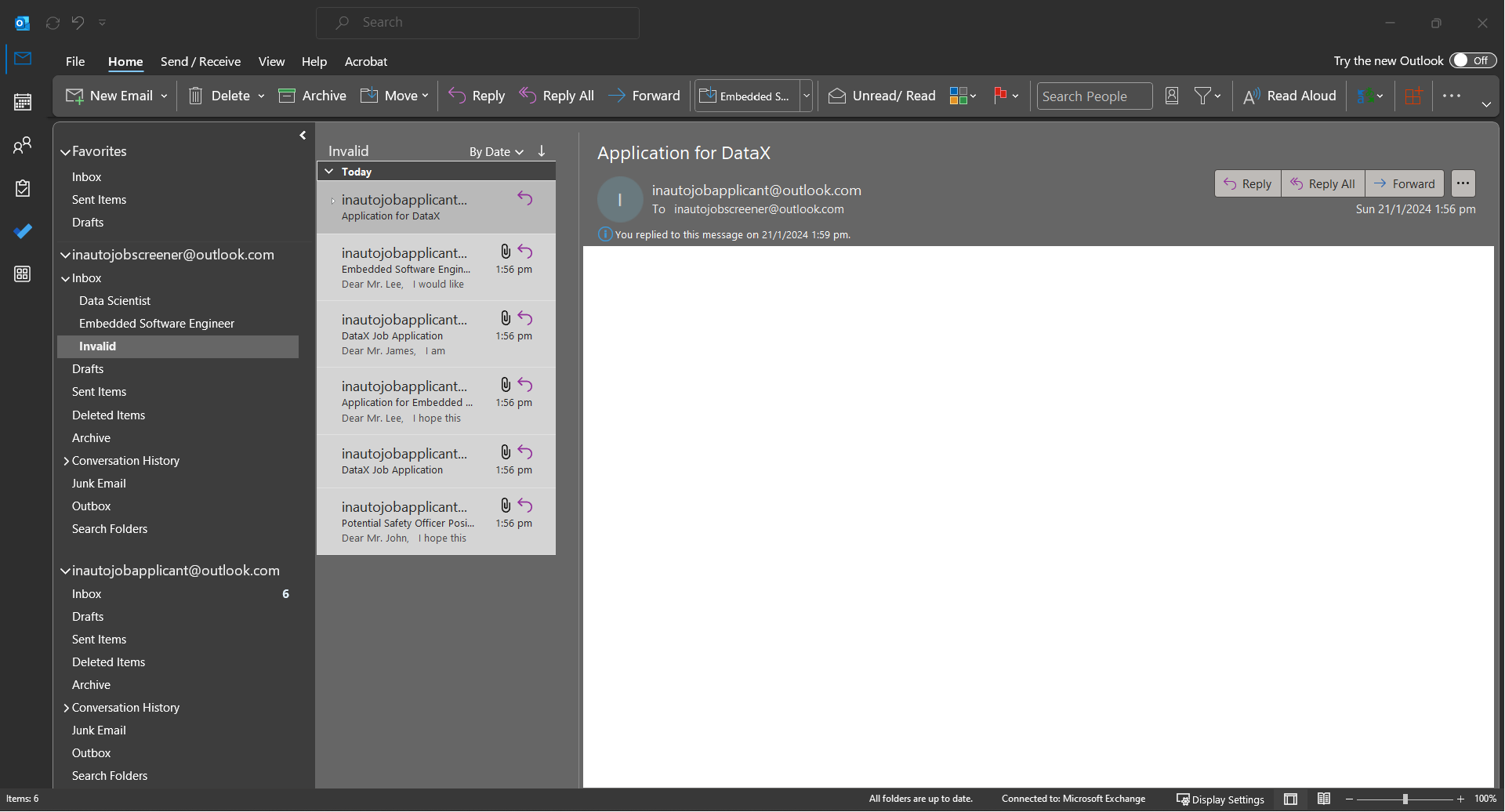
**Company Outlook Account Selection (Input)**



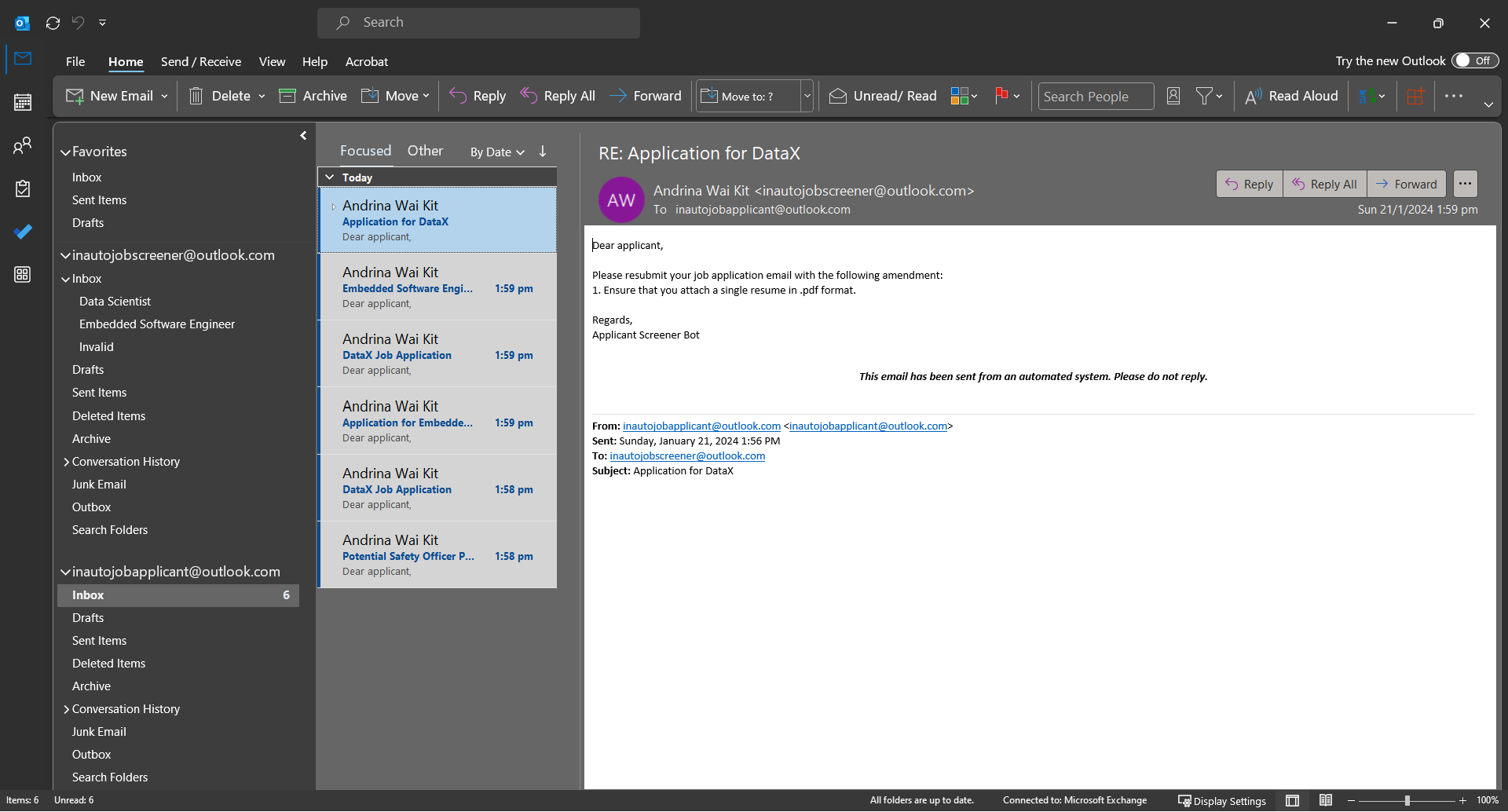
**Processed Job Application Emails Sorted Into the Created Job Position & Invalid Folders in the Company Outlook Account (Output)**



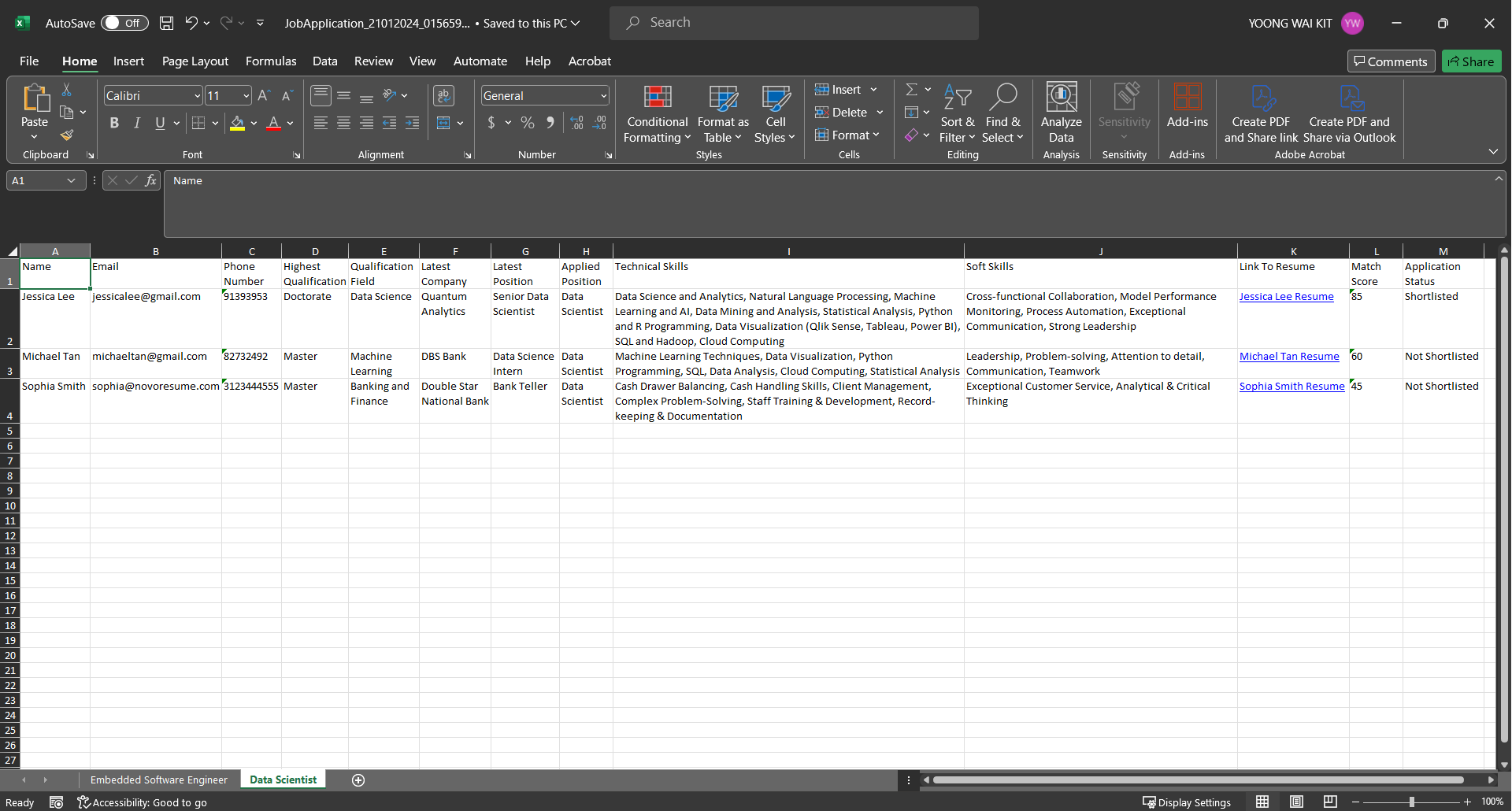




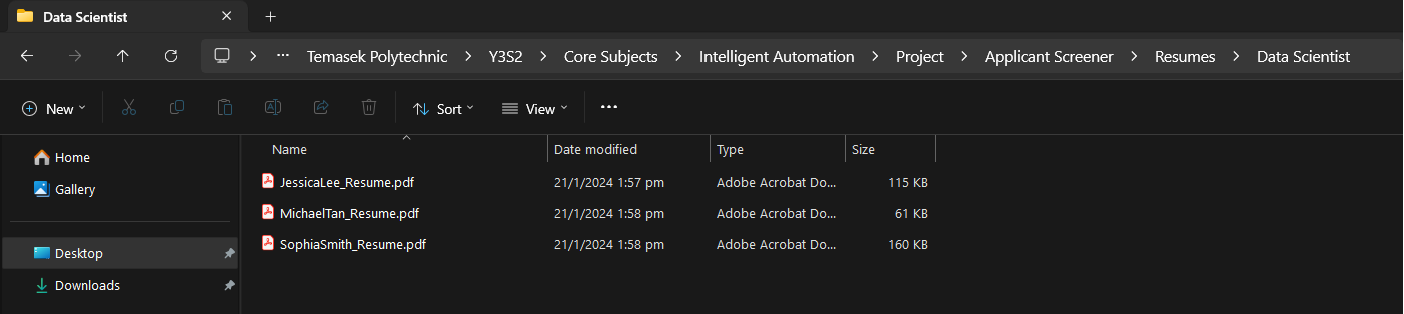
**Invalid Job Application Emails Replied With Instructions for Resubmission (Output)**

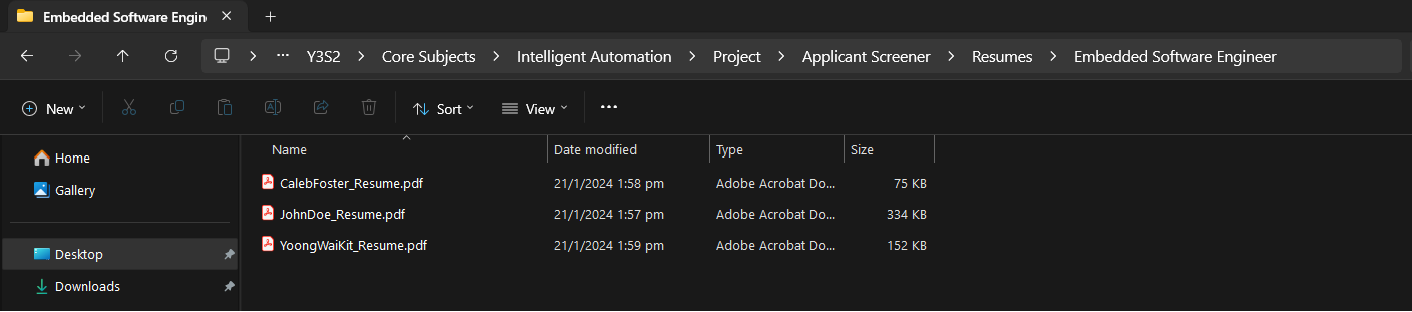


**JobApplication\_[Date]\_[Time].xlsx File with Applicant Details Grouped Into Their Applied Job Position Sheet & Sorted By Descending Match Score (Output)**



**Resumes Stored By Their Applied Job Position Subfolders In the 'Resumes' Folder (Output)**





**~ End of this Document ~**