# Automated Job Application Tracker Using Power Automate

**1. Introduction: The Job Application Challenge**

For many recent graduates and job seekers, applying for jobs is a time-consuming and often overwhelming process. With hundreds of applications sent out over a short period, it becomes difficult to track which positions have been applied for, the responses received, and the necessary follow-ups required.

**1.1 The Need for a Job Application Tracker**

Without a structured tracking system, job seekers face several challenges:

* **Missed follow-ups:** Many candidates forget to follow up with recruiters, reducing their chances of securing an interview.
* **Lack of visibility into application status:** Checking emails manually for updates such as "Application Viewed" or "Unfortunately, we have decided to move forward with another candidate" is inefficient.
* **No centralized dashboard:** There is no single platform that consolidates all application-related information, such as the number of applications sent, the number of rejections received, and pending responses.

A well-designed automation system can solve these problems by automatically tracking applications, updating statuses, and scheduling follow-ups without requiring manual intervention.

**2. Existing Solutions and Why This Automation Was Developed**

**2.1 Alternative Online Job Tracking Tools**

There are several existing tools that assist with job tracking, but they have key limitations:

| **Tool** | **Limitations** |
| --- | --- |
| Google Sheets | Requires manual entry for each application. |
| Job Search Platforms (LinkedIn, Indeed) | Tracks only applications submitted through their platform and does not consolidate all job applications. |
| Dedicated Job Tracking Apps | Often require a paid subscription and lack full automation. |
| Trello/Notion | Requires manual updates and does not automate follow-ups or job status updates. |

**2.2 Why This Automation Was Developed**

Existing tools lack complete automation, email integration, and structured follow-up reminders. This Power Automate-based solution addresses these gaps by:

* Automatically **extracting job application data from emails**.
* **Tracking application statuses** in a structured format.
* **Automating follow-up reminders** without manual input.

This system integrates Power Automate, Gmail API, and Excel automation to create an end-to-end job tracking system that operates seamlessly and reduces the time spent manually updating job applications.

**3. Why Power Automate?**

Power Automate was chosen because it:

* Enables automation between multiple services, including Gmail, Excel, and Notifications.
* Provides workflow management with conditional logic to determine the next steps based on the email content.
* Supports API integration, allowing for the use of a custom Gmail connector.
* Allows for low-code/no-code automation but can be extended with advanced configurations if needed.

The plan for this automation involved:

1. Extracting job application updates from LinkedIn emails.
2. Processing the extracted data and storing it in an Excel file.
3. Automatically updating application statuses based on email content.
4. Sending reminders for follow-ups at scheduled intervals.
5. Overcoming Gmail & Power Automate security restrictions using a custom connector.

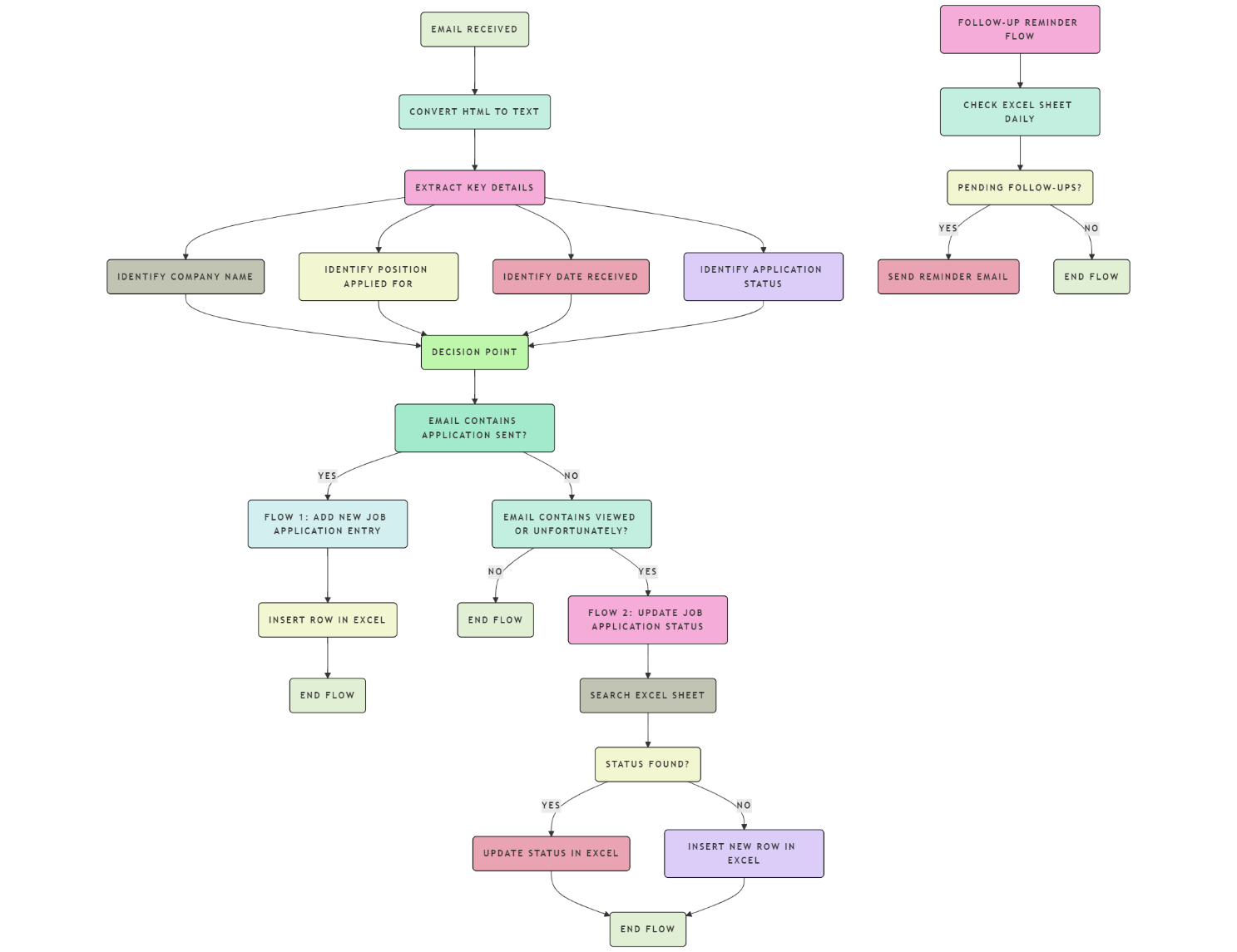
**4. Detailed Explanation of the Process**

The automation consists of three major components:

* Email Processing and Data Extraction
* Decision Logic for Classification
* Updating and Tracking Applications in Excel

**4.1 Flowchart of the Automation Process**

The diagram below represents the automation process in Power Automate.



* **Trigger:** The process starts when an email from LinkedIn is received.
* **Data Extraction:** The email content is converted from HTML to text, and key details such as the **Company Name, Job Title, Date, and Status** are extracted.
* **Decision Point:** The system determines whether the email confirms an application submission or provides an update on the application status.
* **Data Storage & Updates:** If it is a new application, a new row is added to the tracking sheet; if it is an update (e.g., "Application Viewed"), the existing entry is modified.
* **Scheduled Follow-Ups:** The system checks applications at defined intervals and sends reminders accordingly.

**4.2 Step 1: Email Processing & Data Extraction**

**Trigger:**

* The process is initiated when Gmail receives a job application-related email from LinkedIn.

**Actions Taken:**

1. Convert the email from HTML to text to make the content readable.
2. Extract the following information:
   * Company Name
   * Position Applied For
   * Date Received
   * Application Status (e.g., "Applied," "Viewed," "Rejected")

**Challenges Faced:**

* Gmail’s security policies prevent Power Automate from directly retrieving email data due to restricted API access.
* Microsoft Outlook has built-in Power Automate connectors, but Gmail requires additional authentication steps.

**Solution Implemented:**

* A Custom Gmail Connector was developed using Google Cloud Platform (GCP) OAuth 2.0 authentication (I provided a clear explanation and included an alternative at the conclusion of the document).
* This connector allows Power Automate to securely access emails, extract the required data, and process it for tracking.

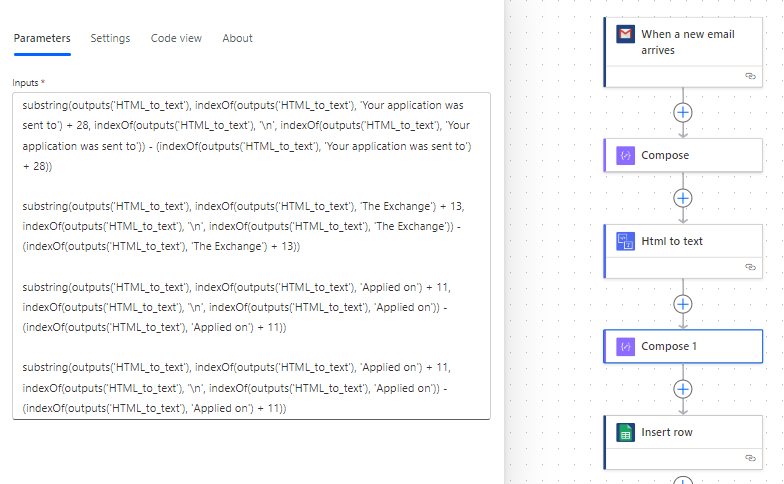
**4.3 Step 2: Decision Logic – Classifying Email Type**

* If the email contains the phrase "Your application was sent to [Company Name]", it is classified as a new application entry (Flow 1).
* If the email contains the phrase "Viewed" or "Unfortunately", it is classified as an application status update (Flow 2).

This ensures that the system accurately differentiates between new applications and status updates, preventing duplicate entries.

**4.4 Step 3: Flow 1 – Adding a New Job Entry to Excel**

* If an application is detected as **new**, the system **inserts a new row** into the job tracking Excel sheet.
* The row includes:
  + **Company Name**
  + **Position Applied For**
  + **Date Applied**
  + **Status: "Applied"**



This step ensures that every new application is automatically logged without manual input, making job tracking easier.

**4.5 Step 4: Flow 2 – Updating Job Application Status**

* If an email indicates an application status update, the system first searches the Excel sheet for the existing entry.
* If the Company Name and Position exist, the Status column is updated accordingly.
* If no existing entry is found, a new row is added with full details.

This prevents unnecessary duplication while ensuring that every job application status is properly tracked.

**4.6 Step 5: Follow-Up Reminder Automation**

**Trigger:**

* The follow-up system runs twice daily (9 AM and 2 PM).

Logic and Actions:

* The automation scans the Excel sheet for applications that were submitted 3 and 7 days ago.
* If an application has not received any response, a reminder notification is sent suggesting a follow-up.
* If an application has already been marked as Viewed or Rejected, no reminder is sent.

This ensures that job seekers can proactively follow up on pending applications without manual tracking.

**5. Conclusion and Future Improvements**

This automation project successfully:

* **Eliminates manual effort** in job application tracking.
* **Ensures real-time status updates** without needing to check emails manually.
* **Improves follow-up efficiency** to increase response rates from recruiters.

### Challenges Faced and Solutions Implemented in the Power Automate Job Tracker

**1. Gmail Custom Connector Issue and Its Alternative (Gmail API)**

One of the biggest challenges faced in this automation project was integrating Gmail with Power Automate. Unlike Microsoft Outlook, which has a built-in Power Automate connector, Gmail restricts direct access due to security and API limitations.

**1.1 Why the Default Power Automate Gmail Connector Was Not Sufficient**

* Power Automate does offer a **Gmail connector**, but it requires **manual sign-in** each time it is used, making automation unreliable.
* **OAuth authentication is mandatory** for API access, adding a layer of complexity to the integration process.
* Power Automate was **unable to extract email contents directly**, requiring an alternative method.

**1.2 Solution: Developing a Custom Gmail Connector**

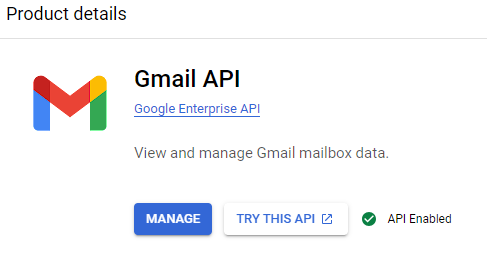
To overcome this limitation, I created a **Custom Gmail Connector using Google Cloud Platform (GCP)**. The steps included:

1. Setting up a Google Cloud project in the Google Cloud Console.
2. Enabling the Gmail API to allow external access.
3. Creating OAuth 2.0 credentials to authenticate the connector securely.
4. Configuring the Power Automate custom connector with authentication tokens.

This custom connector allowed Power Automate to securely retrieve **job application emails**, extract relevant details, and update the job tracking sheet.

**1.3 Exploring an Alternative: Gmail API**

As I continue exploring Google Cloud Console, I discovered that Gmail API can be a better alternative to the Custom Connector approach.



With Gmail API, users can:

* View and manage Gmail mailbox data, including threads, messages, and labels.
* Fetch specific email details programmatically without needing a manual sign-in.
* Use structured API calls instead of relying on Power Automate’s built-in limitations.

Although this is an alternative, integrating Gmail API directly with Power Automate requires additional Google Cloud setup and Azure authentication. In the future, I plan to explore this method to further improve the reliability of this automation.

**2. Power Automate Flow Failures Due to Location Issues**

Another unexpected challenge encountered was flow failures due to location settings in Power Automate.

**2.1 Issue Identified**

* Some Power Automate flows failed instantly, without any clear error messages.
* Upon further research, it was identified that certain cloud services used in Power Automate are region-specific.
* The Gmail integration was running on a different server location than the primary Power Automate environment, causing authentication errors.

**2.2 Solution Implemented**

* Ensured that Power Automate and Google Cloud API services were configured in the same region.
* Updated the Power Automate environment settings to match the GCP API region.
* Added retry logic to handle intermittent failures caused by regional mismatches.

By synchronizing region settings between Power Automate and Google Cloud, the automation became significantly more stable.

**3. Unexpected Power Automate Flow Failures**

While testing various Power Automate workflows, I noticed that some **flows failed instantly** without any proper error messages.

**3.1 Investigation and Findings**

* Microsoft documentation does not clearly specify why certain flows fail immediately.
* Some failures were due to **invalid authentication tokens**, especially when OAuth credentials expired.
* In some cases, Power Automate **failed due to API rate limits**, meaning that too many requests were made in a short period.

**3.2 Solution Implemented**

* Added error handling logic to capture flow failures and send notifications when a flow fails.
* Implemented token refresh mechanisms to prevent authentication failures.
* Monitored Power Automate run history to detect patterns in failures and make necessary adjustments.

This helped reduce the number of failed flows, improving the automation's overall reliability.

**4. Converting HTML to Text – A Major Challenge**

Another significant challenge was extracting text from Gmail emails, which are usually formatted in HTML.

**4.1 Issue Identified**

* Gmail emails contain HTML elements, inline styles, and embedded links, making it difficult to extract clean, structured text.
* Power Automate does not have a built-in HTML to Text converter for Gmail emails.
* The extracted data often contained unwanted metadata, causing issues in job tracking.

**4.2 Solution Implemented**

To resolve this, I used a combination of HTML parsing and text extraction methods, including:

1. **Developing a Power Automate Custom Connector** that:
   * Reads the **email content**.
   * Converts the **HTML to plain text**.
   * Removes **unnecessary elements** such as scripts and styles.
2. Using Power Automate Expressions to refine extracted text and eliminate formatting issues.
3. Testing different Gmail email structures to ensure all job-related emails were processed correctly.

By fine-tuning the HTML-to-text conversion, the automation was able to reliably extract only relevant job details while ignoring extra formatting.

**Conclusion: Lessons Learned and Future Enhancements**

The challenges faced during this project provided valuable insights into:

1. **How Gmail security impacts API integrations** and why OAuth authentication is critical.
2. **Power Automate's limitations** regarding region settings and API failures.
3. **How to improve automation resilience** by implementing error handling and failover mechanisms.
4. **The complexities of handling HTML-based emails** and the importance of structured text extraction.

**Future Enhancements**

* Further exploration of the **Gmail API** as a **direct alternative** to the custom connector.
* Implementing **machine learning-based text parsing** for more **intelligent email classification**.
* Expanding the automation to **track job applications across multiple platforms (LinkedIn, Indeed, etc.)**.

This project is an ongoing effort to enhance workflow automation using Power Automate, API integrations, and structured data processing. The lessons learned will be applied to future improvements and optimizations.

### Final Thoughts and Next Steps

Through this project, I successfully **implemented job application tracking automation using Power Automate**, overcoming key challenges such as **API restrictions, authentication barriers, regional failures, and HTML data extraction**. This experience not only strengthened my understanding of **workflow automation** but also provided valuable insights into troubleshooting and optimizing **real-world automation processes**.

Now, it's time to take this knowledge further—**applying the same problem-solving mindset to AI-driven automation and intelligent agent technology**. As I explore **AI agents and their potential in automation**, I look forward to developing more advanced, intelligent systems that push the boundaries of efficiency and innovation.

Let’s meet again soon, with **new implementations, enhanced automation, and smarter solutions**! 🚀

- **Santhosh Botcha**