DIGITAL AUTOMATION CHALLENGES 2025

Name: Mohamad Azafri bin Abd Majid

Number Matrics: B23CSS0044

Background Of The Solution

Traditionally, submitting a form application for each individual row in a spreadsheet would require multiple manual entries, which is time-consuming and error-prone. To overcome this limitation, my solution introduces a **bulk submission approach**.

Instead of submitting data row by row, the user only needs to provide a Google Sheets link, and the entire process is automated. Once the link is submitted, the system triggers a UiPath workflow via an API call from the Django REST API. UiPath then handles the entire automation pipeline—from downloading and processing the sheet to identifying and separating the data.

The UiPath workflow processes the spreadsheet and categorizes the entries into two groups:

- Valid Data: Entries without duplication.
- Duplicate Data: Entries that have matching values for specific fields (e.g., Staff ID and Start Date).

Once the workflow completes, it sends a response back to the Django backend containing both categories. The backend then stores the data directly into the database, eliminating the need for further user intervention.

This solution streamlines the workflow, reduces manual effort, and ensures data is accurately validated and submitted in bulk, resulting in a more efficient and scalable process.

Process Flow For The Solution

There are 2 workflow to solve this challenge Main.xml and AutomationBulkUpload

1. **Main.xml**

   The process of scraping the data from the Google Sheet link is being carried out using it. Additionally, any duplicate data will be found by this workflow and returned as a JSON object. The Uipath Orchestrator API will be triggered to initiate this procedure.

   a. **Assign Spreadsheet ID:**
   Obtain the Google Sheet link and extract the `spreadsheetId`, which will be used for accessing the file.

   b. **Connect to Google Spreadsheet:**
   Authenticate with the Google account and access the spreadsheet using the `spreadsheetId`.

   c. **Download Spreadsheet:**
   Once the spreadsheet is accessed, download it in Microsoft Excel format (`.xlsx`) to ensure uniformity across sheets. The downloaded file path is saved to a variable named `tempPath` for later use.

   d. **Read Data Range:**
   Read the content of the downloaded workbook into a `DataTable` object, which is stored in a variable called `dtSheet`. This data is also assigned to a logical reference named “Sheet5”.

   e. **Convert DataTable to Text (Log Output):**
   Convert the contents of `dtSheet` to text format and store it in the variable `dtSheetText`. This enables easy inspection via log messages.

   f. **Log Message - Initial Data Check:**
   Log the value of `dtSheetText` to verify that the correct data has been read from the sheet.

g. **Identify Duplicate Entries:**
Perform a check for duplicate rows based on `Staff ID` and `Start Date` .This expression identifies rows that share the same `Staff ID` and `Start Date`, and stores them in a new `DataTable` called `dtDuplicateSheet`. We will use the following LINQ expression:

```
(From p In dtSheet.Select()
 Where (From q In dtSheet.Select()
        Where q("Staff ID").Equals(p("Staff ID")) And q("Start
Date").Equals(p("Start Date"))
        Select q).ToArray.Count > 1
 Select p).ToArray.CopyToDataTable()
```

h. **Convert Duplicate DataTable to Text (Log Output):**
Convert the contents of `dtDuplicateSheet` into text format for logging, and store it in the variable `dtDuplicateSheetText`.

i. **Log Message - Duplicate Check:**
Log the value of `dtDuplicateSheetText` to review and confirm the duplicate entries.

j. **Cleanup:**
After processing is complete, delete the downloaded Excel file to conserve storage space.

2. AutomationBulkUpload.xaml

   This is a workflow to run the process of login in into the system, navigate to the submission form and submit the form

   a. Input Dialog: user will enter the google sheets link inside the dialog. The input will be assigned to 'googleSheetsLink' variable
   b. Try :
      i. Use Application/Browser: We will navigate to 'browserURL' value
      Note: By default, it is "http://localhost:8000/login/". Thus, you have to run the Django project first before running this UIPath Automation.
      ii. Type Input: We will enter user's username into the text field
      Note: Use **'admin'** as username
      iii. Type Input: We will enter the password into the text field
      Note: Use **'admin'** as password
      iv. Click Sign In: Click sign in to login
      v. Click OK: Normally google will prompt regarding the safety about password. Will click 'OK' to continue
      vi. Click bulk upload: On the sidebar we will click the bulk upload to navigate to the form
      vii. Click 'Google Sheets Link': This will select an option to submit the form by giving the google sheets link
      viii. Type into 'Google Sheets URL': We will pass the value of 'googleSheetsLink' variable into here
      ix. Click 'Process Sheet': We will submit the form by clicking on the 'Process Sheet' button.
      Note: Here, the backend will trigger an API in UIPath Orchestrator. It will run the Main.xaml workflow to read the sheet.
      x. Get text for successful entries: Once the process is finish, we will get the value of successful entries and will be stored in 'successfulCountText'

xi. Assign: We will parse the string value from ‘successfulCountText’ into Int32 and store it in ‘successfulCountInt’

xii. Get text for failed entries : We will also get the value of failed(duplicates) entries and will be stored in ‘failedCountText’

xiii. Assign: We will parse the string value from failed‘CountText’ into Int32 and store it in ‘failedCountInt’

xiv. Click on ‘Failed Entries’ tab: There will be 2 tabs showing table for successful and failed. We will click on failed entries tab since we want to take a screenshot later

xv. Assign: We will calculate the total entries by adding total of successful entries and failed entries

xvi. Send SMTP Email: We will send the email to sivanesan.letchumanan@dhl.com regarding the new data entries. In the body of the email, we will tell the total of successful, failed and total entries. The screenshot of the successful page also will be given

xvii. Click ‘Go to dashboard’: Once all done, we will go back to dashboard

c. Catch

i. Take screenshot: We will get the screenshot of UIPath to show the error that has been occurred

ii. Send SMTP email: We will send the email regarding the error to sivanesan.letchumanan@dhl.com with an attachment of the screenshot of the error

High-Level Explanation of Key Functions Used in Web Solution & UiPath

The **web solution** is built using the Django framework, which serves both the frontend and backend components of the application. On the frontend, users interact with a form interface designed to accept a Google Sheets link for bulk data submission. The **backend** is responsible for handling form submissions and coordinating with external services.

A key function of the backend is its integration with **UiPath Orchestrator**. Specifically, one of the core APIs is designed to **trigger a UiPath workflow** upon receiving a valid submission. This API call initiates the execution of a pre-defined automation process (`Main.xaml`) through UiPath Orchestrator. The workflow processes the submitted Google Sheet, checks for duplicate entries, and returns a response indicating the number of successful and failed (duplicate) entries.

On the **UiPath** side, several essential activities are used to support this automation process:

- Reading and downloading spreadsheets from Google Sheets.
- Parsing data into `DataTable` format.
- Performing duplicate detection logic using LINQ queries.
- Logging outputs for traceability.
- Sending automated email notifications summarizing the results.
- Handling errors with proper logging and screenshots.

Users can execute `AutomationBulkUpload.xaml` if they choose not to manually navigate and submit the form. This will execute all of the automation on its own, including form submission and login.

How To Execute The Web Solution & Script

MongoDB

Requirements

- You must have MongoDB service running in your machine

Setup Steps

1. If you already have MongoDB in your machine, you only need to run the Django project since everything will be automated.

Information

1. The database is called “dhl_leave_management”
2. It will be running on mongodb://localhost:27017

Django

Requirements

- Python 3.x installed (preferably Python 3.8+)
- pip (Python package manager)
- virtual environment support (venv or virtualenv)

Setup Steps

1. Create a virtual environment
    a. Open Command Prompt (Windows) or Terminal (Mac/Linux)
    b. Navigate to the project directory
    c. Create a virtual environment

```
python -m venv venv
```

    d. Activate the virtual environment
    On Windows:

```
venv\Scripts\activate
```

    On Mac/Linux:

```
source venv/bin/activate
```

2. Install dependencies

```
pip install -r requirements.txt
```

3. Run the development server
    a. Make sure you're in the project directory where manage.py is located
    b. Run the following command

```
python manage.py runserver
```

c. The server will start at http://127.0.0.1:8000/ or http://localhost:8000/

4. Access the development
    a. Open your web browser
    b. Go to http://127.0.0.1:8000/ or http://localhost:8000/