Summary for UiPath Project: Amazon Toaster Search with Dispatcher and Performer

Project Overview

Provide a brief overview of the project, its purpose, and the business problem it aims to solve.

This is an RPA project that involves two main modules: dispatcher and performer. These two processes work sequentially. They aim to automate the retrieval and segregation of toaster items from Amazon with respect to their price. This helps increase processing speed, decrease margin of error from moving data by humans and eliminate the need of manual extraction of products.

Dispatcher Module

1. **Purpose**: Extracts toaster details from Amazon and uploads them to Orchestrator
2. **Key Functions**:
    - **Data Collection**: Toaster details (title, price and rating) are retrieved from the search query on Amazon.
    - **Queue Management**: The toaster details are added to the data table and pushed to orchestrator
    - **Error Handling**: In case no price is provided for the product, the item is ignored.
3. **Workflow Summary**: The dispatcher starts by opening the amazon page and searching for toasters. Then extracting the first 100 products (excluding products without “toaster” in their title and no price displayed). Then they are populated into one of the two data tables (under 40 dollars and over 40 dollars). Each data table is uploaded to a queue in Orchestrator.

Performer Module

1. **Purpose**: Retrieves transactions from orchestrator queues and writes each queue to a workbook (csv) file depending on their price category(under or over 40 dollars).
2. **Key Functions**:
    - **Initialization**: Initializes two data tables to hold transaction item details before appending to workbooks
    - **Transaction Processing**: The transaction is analyzed by its price and added to the appropriate workbook.
    - **Transaction Status Update**: If the “Process” workflow executes with no exceptions then it is marked as successful in Orchestrator. Default REFramework status handling have been adapted.

- **Error Handling and Retry**: The retry activity is invokes twice in case of getting transaction item in GetTransactionData.xaml and SetTransactionStatus.xaml

3. **Workflow Summary**: The performer initializes two data tables for each price category. Then retrieves transaction items from each queue in Orchestrator, appending to the appropriate workbook file in the local directory.

Interaction Between Dispatcher and Performer

1. **Queue as a Communication Medium**: The dispatcher the queue items into one of the two queues (Queue1 and Queue2) based on the price category. The performer then proceeds to retrieve the transactions and saves their details locally.
2. **Asynchronous Processing**: This helps modularize functionalities of the entre project. This way there is a clear distinction of which part of the project will prepare the data (upload) to orchestrator, and which part will retrieve the transactions from orchestrator.
3. **Scalability**: Since queues can hold unlimited items, then the dispatcher could be modified to select more than 100 items and the performer shall adapt accordingly by processing in the same manner.

Conclusion

The RPA application improves the performance of the manual process. Also ensuring consistency by logging the success/failure of each module. The scalability of the project can also be proven since the project has been built in a modular manner, enabling modifications without altering the rest of the project.