­­­­­

|  |  |
| --- | --- |
|  |  |



Solution Design

Document – Approval Bot

Table of Contents

[I. Purpose 3](#_Toc5787525)

[II. Automated process details 4](#_Toc5787528)

[3 Runtime guide 5](#_Toc5787529)

[3.1 Architectural structure of the Master Project 5](#_Toc5787530)

[3.2 Master Project Runtime Details 5](#_Toc5787531)

[3.3 Project name 1 6](#_Toc5787532)

[3.4 Project(s) workflows 6](#_Toc5787533)

[3.5 Packages 7](#_Toc5787534)

[3.6 Architectural structure of the Master Project 7](#_Toc5787535)

[4 Other Details 8](#_Toc5787536)

[Future Improvements 8](#_Toc5787537)

[Other Remarks 8](#_Toc5787538)

[5 Glossary 9](#_Toc5787539)

# Purpose



Approval bot is an idea to channel various approval requests that an employee in a corporate environment deals with on a day-to-day basis into a one stop solution.

It is most likely that these approvals come from different sources of origin and when concentrated at the top level could overwhelm the employee and their efforts to keep them in track.

The main Idea is to bring the data from multiple applications into a single common data pool and from there it is relayed to the end user through various communication channels. The current version of the approval bot includes an email alert and a MS Teams card alert. The input is sourced from a SQL Server database located in Azure. The different data tables act as independent approval types.

The input connections, query and column mapping are customized in the config excel. The templates for output also reside in the input folder. The process is built on RE Framework and is built to handle exceptions. The main touch points are logged with various log levels and each iteration.

An intermediate Power Automate connection is used to send the MS Teams alert. Alternative to this can be found in UiPath marketplace which requires Azure AD app and respective tenant and client. The Power Automate solution has integrated connectivity to MS Teams which was used and can be triggered by firing an API request.

# Automated process details

|  |  |
| --- | --- |
| Item | Description |
| Master Project Name | ApprovalReminderBot |
| Robot Type | BOR |
| Orchestrator used? | Yes - Assets |
| Scalable | Yes |
| UiPath version used | 2023.8 |

# Runtime guide

## Architectural structure of the Master Project

## A diagram of a system Description automatically generatedMaster Project Runtime Details

Outlines the details of the automated process by filling in the table below.

|  |  |
| --- | --- |
| ITEM NAME | DESCRIPTION  *Fill in each bolded section - empty fields are not allowed. If the section does not apply to your automation then mark as n/a.* |
| Production environment details | ***No specific environment*** |
| Prerequisites to run | ***Database credentials in Assets*** |
| Input Data | ***Data table with Pending Approval requests*** |
| Expected output | ***Email and Teams card sent to individual approver*** |
| How to start the automated process | ***Manual Trigger*** |
| Reporting  (queues reporting, Kibana or another platform) | *Orchestrator logs and jobs dashboards.* |
| How is Orchestrator used? | *Orchestrator used for asset passwords.* |
| Password policies  (mention any specific compliance requests) | ***Database server passwords*** |
| Stored credentials  (Never use hardcoded credentials in the workflow!) | *Stored in Orchestrator Assets* |
| List of queues names  (Naming convention: ProcessName\_QueueName) | ***NA*** |
| Schedule Details | ***TBD*** |
| Multiple Resolutions Supported?  (in case of image automation / Citrix and VDI) | ***NA*** |
| Recommended Resolution | ***NA*** |

## Project name

|  |  |
| --- | --- |
| ITEM NAME | DESCRIPTION  *Fill in each section - empty fields are not allowed. If the section does not apply to your automation then mark as n/a.* |
| Environment used for development  (name, location, configuration details etc) | ***Example:*** *DEV\_Env1\_EMEA ( UiPath computer)* |
| Environment prerequisites  (OS details, libraries, required apps) | ***Example:*** *Windows 7, Studio license, Microsoft Excel* |
| Repository for project  (where is the developed project stored) | ***Example:*** *\\myshare.com\Zendesk* |
| Configuration method  (assets, excel file, Json file) | ***Example:*** *Config.xlsx, SQLConnections.json* |
| List of reused components | ***NA*** |
|
| List of new reusable components | ***NA*** |

**Associated Files**

|  |  |
| --- | --- |
| File Name | Description |
| Data/Config.xlsx | *Main uipath config file* |
| Data/SQLConnections.json | *Sql connections config file* |
| Data/Input/AdaptiveJSON.json | *Template for teams card* |
| Data/Input/EmailTemplateMain.html | *Template for email report* |
| Data/Input/WBZlogo.jpg | *Logo image file* |
| Data/Output/HTMLOut.txt | *Temp file generated by bot* |
| Data/Output/JsonOut | *Temp file generated by bot* |
| Data/Output/Jsonobjecttableplaintext | *Temp file generated by bot* |

## Project(s) workflows

Workflows specific to: Specify Project Name from section above

For the workflow files defined below please specify the input and output parameters.

|  |  |
| --- | --- |
| Workflow Name | Description |
| Main.xaml | *Entry point. Invokes all the other workflows* |
| Framework/CloseAllapplications.xaml | *Soft close all open applications* |
| Framework/GetTransactionData.xaml | *Feed individual transaction item to process* |
| Framework/InitAllApplications.xaml | *Initialization workflow. Identifies the input data and Invokes RunDBQueries.xaml* |
| Framework/InitAllSettings.xaml | *Read config.xlsx and SQLconnections.json and store as variables* |
| Framework/KillAllProcess.xaml | *Force quit applications* |
| Framework/Process.xaml | *Process individual transactions. Get transaction item for the iteration and invokes ConstructEmailHTML.xaml and ConstructTeamsJSON.xaml workflows.* |
| Framework/SetTransactionStatus.xaml | *No status is set as it does not have queue. However, upon exceptions, this will send the error email.* |
| Framework/RetryTransaction.xaml | *Retries the failed transactions* |
| Framework/TakeScreenshot.xaml | *Take screenshot when required.* |
| RunDBQueries.xaml | *Identifies the connection and queries the DB and compiles the result into JSON Object* |
| ConstructEmailHTML.xaml | *Reads the JSON object and generates a HTML file dynamically and will be sent to the end user via Email* |
| ConstructTeamsJSON.xaml | ***Reads the JSON object and generates an adaptive JSON file dynamically and will be sent to the end user via Teams*** |

## Packages

Include the list of packages and high-level description for each of them, to explain their purpose

|  |  |
| --- | --- |
| Package Name | Description |
| *Approval\_Bot\_Teams\_Alert\_20230922074137.zip* | ***Triggers power automate flow upon a POST API request from UiPath. Payload containing the end user email and adaptive Json card information.*** |

## Architectural structure of the Master Project

## 

# Other Details

### Future Improvements

The current version of approval bot can handle inputs from multiple databases simultaneously and can be customized to map only the relevant information.

For all the applications which we do not have an input database, we could call the request details through an API call and input the information into a common Approval bot database which acts as a source.

The current version will only carry a hyperlink that routes to the respective application’s request page. We could configure few additional features using Node.JS to use the hyperlink as a decision itself or could trigger another flow which will take the decision and update the common and source databases.

Current version has 2 communication channels – Email and MS Teams. More channels like Slack/ Discord can also be integrated.

Another scalable option involves the end user who can also opt for the communication channel they would prefer and could schedule at what time they would prefer to receive the alert.