MIB BOT



PROJECT MANAGER

Varun Gupta

DOCUMENTED BY

Gunjan Khatter

Contents

1. Overview……………………………………………………………3
2. Tech Stack………………………………………………………….3
3. Steps to execute…………………………………………………...4
4. App Service Backend………………………………………….…..8
5. Logic Apps Workflow…………………………………………….12
6. OVERVIEW

The BOT application is implemented with a vision to support all PMOs

Within Mc Kinsey to pull imputation reports and analyze the data for the employees’ timesheets.

The application has been designed to illustrate the differences between the nagarro and client (WAND) timesheet data so that defaulters list can be obtained through output generated.

There is also a provision to depict the resource billing in the form of excel report which can serve as an parameter of managing budget flows for the designated project.

1. TECH STACK
2. Programming languages - Power FX and java 11.
3. Platform as a Service – powerspps
4. Cloud provider – Azure
5. Infrastructure resources/services

* Logic Apps
* Storage Containers
* App Services

1. Integration applications

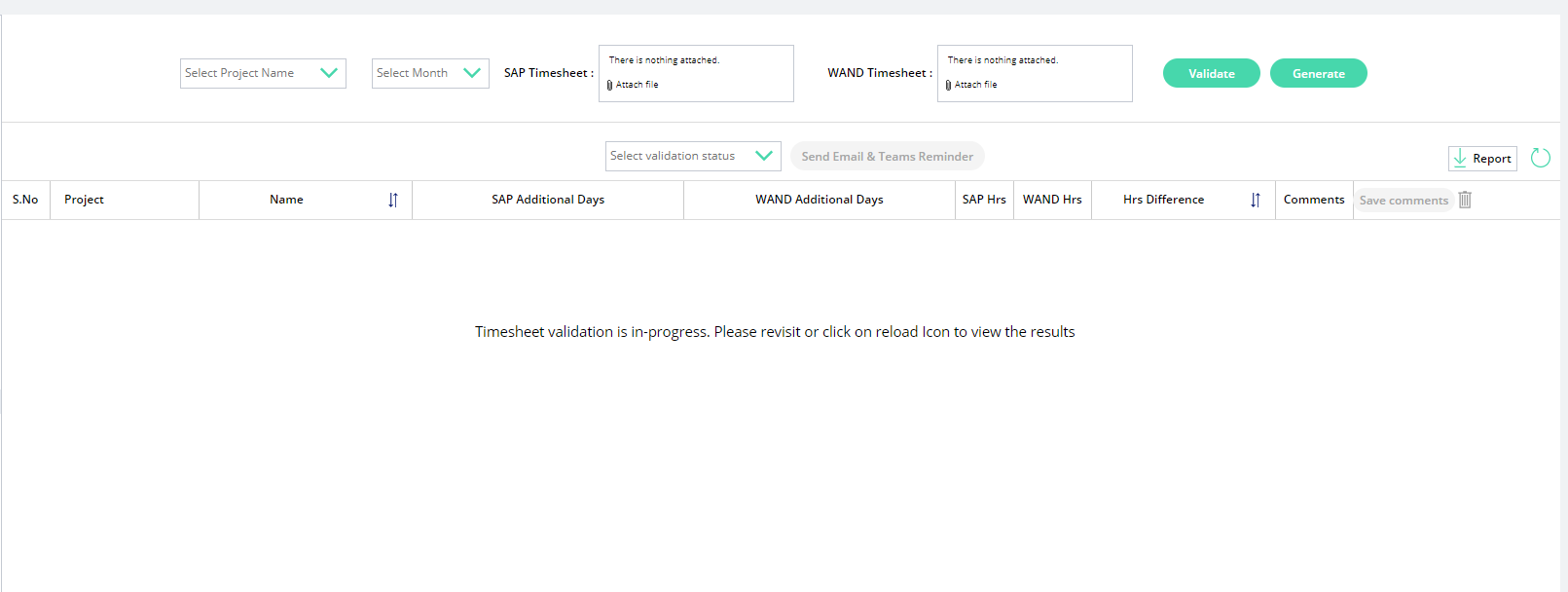
* Outlook
* Sharepoint.

1. EXECUTION STEPS
2. Prepare WAND timesheet in a pivot table format where names are listed vertically and the daily imputation for the entire month is present date wise.
3. Similarly download SAP timesheet. Make sure its in .xlsx format.
4. Go to powerapps application.
5. For better experience , you may download the config file from left pane so that availability of all employee names can be confirmed to be present against each project. If the names are not present in the master config , there will be no potential comparison between the nagarro and WAND data; the missing names will be reflected as error in the output excel.
6. Click on Global Actions -> verify timesheets.

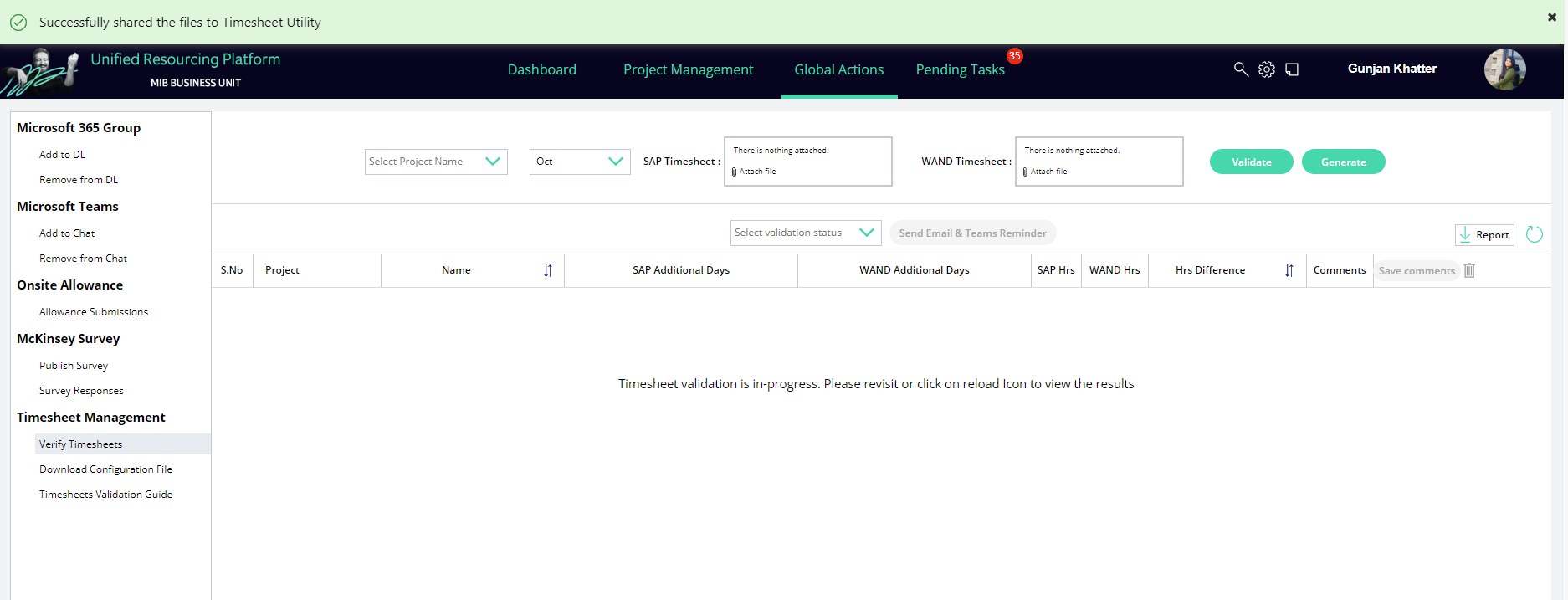
Graphical user interface

Description automatically generated

1. Please select the project name and upload the excel files. You will see timesheet validation in progress as shown.



1. Click on Validate. You will receive a notification on the top of the screen as shown.



1. The user will be able to see the acknowledgement of sending all three files to the BOT via sent items in mailbox as shown.

Graphical user interface

Description automatically generated

1. After waiting for a couple of minutes, you will receive the output generated in your inbox with subject as “Timesheet\_defaulters”.

Chart

Description automatically generated with low confidence

1. In order to get the billing data alongwith the defaulters list, in step vii go to Generate option instead of validate button.
2. Also , the response will be reflected in the sharepoint location:

https://nagarro.sharepoint.com/sites/mckinsey-resourcingmanagement-dev/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2Fmckinsey%2Dresourcingmanagement%2Ddev%2FShared%20Documents&viewpath=%2Fsites%2Fmckinsey%2Dresourcingmanagement%2Ddev%2FShared%20Documents%2FForms%2FAllItems%2Easpx&id=%2Fsites%2Fmckinsey%2Dresourcingmanagement%2Ddev%2FShared%20Documents%2Ftimesheet&viewid=f21cd3b4%2D1225%2D4fb9%2D8c71%2Dc6a0bae87726

1. If automation is failed due to some error in file processing, and output file is not generated, you will receive an automated reply with subject "Timesheet automation failure".

Text, letter

Description automatically generated

1. In case there is some issue related to execution based on outlook mails, please post all the required files on below URL using any rest client such as Postman :

<https://timesheet-mgmt-mckinsey.azurewebsites.net/mcKinsey/upload>

Graphical user interface, text, application, email

Description automatically generated

THINGS TO NOTE

- Download config file from powerapps and ensure that contains data in excel comma separated values (.csv) emailAddress,proWandName,projectName should be the header value.

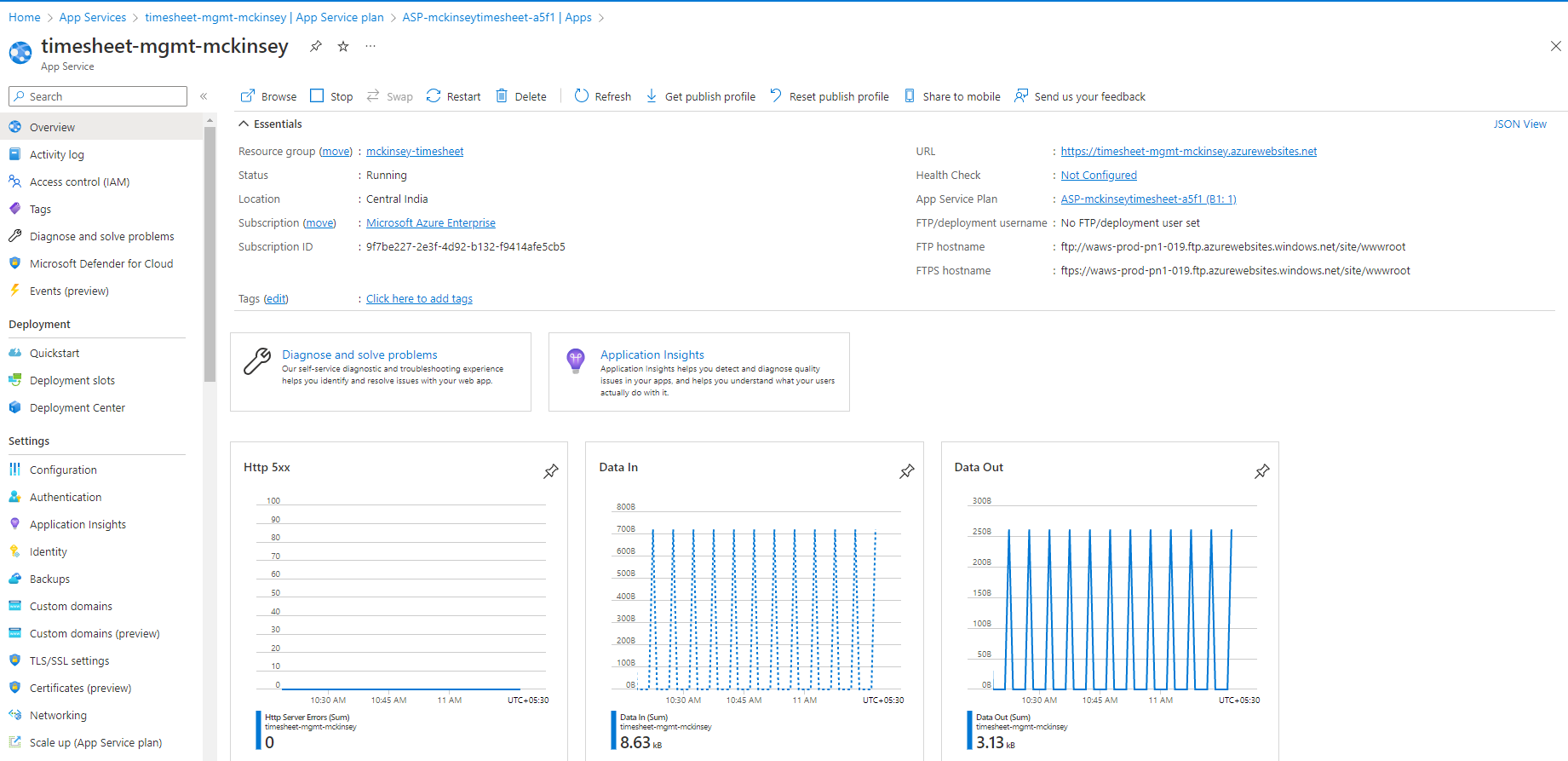
- For Pro Wand timesheet, please add "WAND" keyword on the .xls filename so that program identifies the file for client imputation.

- In files key, please upload required files from your local system.

1. APP SERVICE BACKEND

In azure , the app service is backed up by java springboot application that has been hosted on HTTP endpoint & configuration settings as given –

<https://timesheet-mgmt-mckinsey.azurewebsites.net>



Graphical user interface, application

Description automatically generated

Project structure –

Graphical user interface

Description automatically generated with low confidence

Source Code Repo - https://github.com/gunjank2611/timesheet

API specifications –

1. AzureFileUploadConfig – this configures a handle to file upload specs in order to have connection with Blob storage container.
2. AzureFileUploadConfigurationProperties – this bean autowires all the blob related properties mentioned in resources yaml.
3. MultipartFile - this file designates spring level capacity of file upload , currently max file size is 309600KB
4. SwaggerConfig – Swagger configuration with endpoint metadata resides in this file.
5. AttendanceController – this is the main controller which has post mapping for upload and accepts list of multipart files – i.e. sap xlsx, wand xls and config .csv. It returns 200 OK with the path to the azure blob container in case of successful run and returns 500 in case of failure such as incorrect file formats, improper data in timesheets.
6. ConfigDTO – It contains project name mapping to process in the main logic.
7. MapUtility – Map utility contains entries for entire data read after parsing the excel files and config files.
8. MonthEnum – Enum defines the month and its order mapping so that regex can be matched for date values in the excel headers.
9. ReadNagarroCSV – This file contains the logic for reading sap data. In this we have the logic to map header values as maps (employee names, dates) and their corresponding values in arraylist and the data for a particular employee can be found at a particular index in the list backed by date key.
10. ReadWANDexcel – WAND excel is also read as in step ix, keeping handling for regex for dates as DD-MON.
11. UploadOutputFileService – This is the service based on file upload config i.e. i , ii steps and helps to upload file in azure container.
12. WriteToExcel

* writeEmployeeData holds the main logic such as pre check validation , parameters like if billing excel needs to be generated , and behaves as the caller for all children methods listed below to generate the final output.
* getExceptionRowNum filters out the names of the employees which do not have mapped info in config file.
* writeHeaderData helps to write headers in output file as required.
* writeEmployeeNames writes the data for all the potential employees present in config as well as for those names in SAP or WAND where mapping is not found in config.
* getDifferenceInTimesheets calculates the difference in the filled hours by an employee for each day of the month.
* calculateSAPtotal prints the summation of SAP hous total for an employee.
* calculateWANDtotal prints the summation of WAND hours for an employee.
* getAdditionalDataRows prints the records where either we have No data found in SAP/No data found in WAND for a employee.
* createBillingWorksheet creates the billing worksheet next to defaulters worksheet and prints SAP employee names against the currency, project name, billing rate and calculated sum as per imputed hours.

1. Exceptions:

|  |  |  |
| --- | --- | --- |
| **Exception** | **Reason** | **HTTP status** |
| IncorrectSAPDataException | Incorrect SAP data | BAD\_REQUEST |
| IncorrectWandDataException | Incorrect WAND data | BAD\_REQUEST |
| InvalidFileFormatException | Incorrect file extension(s) found. | BAD\_REQUEST |
| OutputUploadException | Incorrect blob configuration or Output File size exceeds than permissible limits. | INTERNAL\_SERVER\_ERROR |
| BillingCannotBeGeneratedException | Exception found in generating billing worksheet due to parsing errors. | INTERNAL\_SERVER\_ERROR |

1. LOGIC APPS WORKFLOW





Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated