

Node.js Backend Module

Functional Design

Billing Automated

Requisition Service

(BARS)



# Table of Contents

[Table of Contents 2](#_Toc98422594)

[Revision History 3](#_Toc98422595)

[1. Overview 4](#_Toc98422596)

[2. Input/Output Design 5](#_Toc98422597)

[2.1. Send Request File and Process File Through Browser/Postman/ARC 5](#_Toc98422598)

[2.2. Error Response When File is Not Processed Successfully 6](#_Toc98422599)

[2.3. View Activity Logs in Console/CLI 6](#_Toc98422600)

[3. Business Rule 7](#_Toc98422601)

[3.1. Send Request File through Postman and Process File 7](#_Toc98422602)

[3.2. Receive response in Postman and Display Activity Logs in CLI 8](#_Toc98422603)

[4. Functional Workflow Diagram 9](#_Toc98422604)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Summary** | **Updated By** |
| 3.0 | 3/22/2022 | Update documentation. | Kenneth Bolima |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

This document defines the Functional Specification for the Billing Automated Requisition Service application. The application has two major Use-Cases:

1. Send Request File
2. View response data

**Billing Automated Requisition Service**

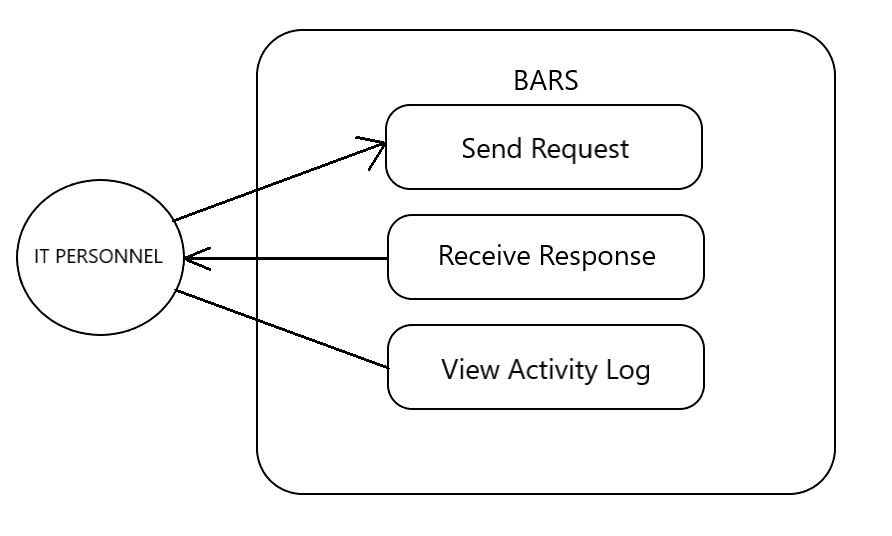


Figure 1: Billing Automated Requisition Service – Use Case

The figure above demonstrates the use case for IT Personnel for the system wherein the IT Personnel can Send Request Files in specific file format such as TXT or CSV. The application allows 3 parameters on the file such as billing cycle, start date, end date. IT Personnel can also Receive Response from CLI based on the information provided from the Send Request File feature/function. The Response contain the billing cycle summary report per account which includes the billing cycle, start date, end date, account name, customer’s first name and last name, and amount.

**IT Personnel**

1. Send Request Files in specific format
2. Receive correct JSON response
3. View the activity logs and errors encountered.

# Input/Output Design

* 1. ***Send Request File and Process File through Browser/Postman/ARC***

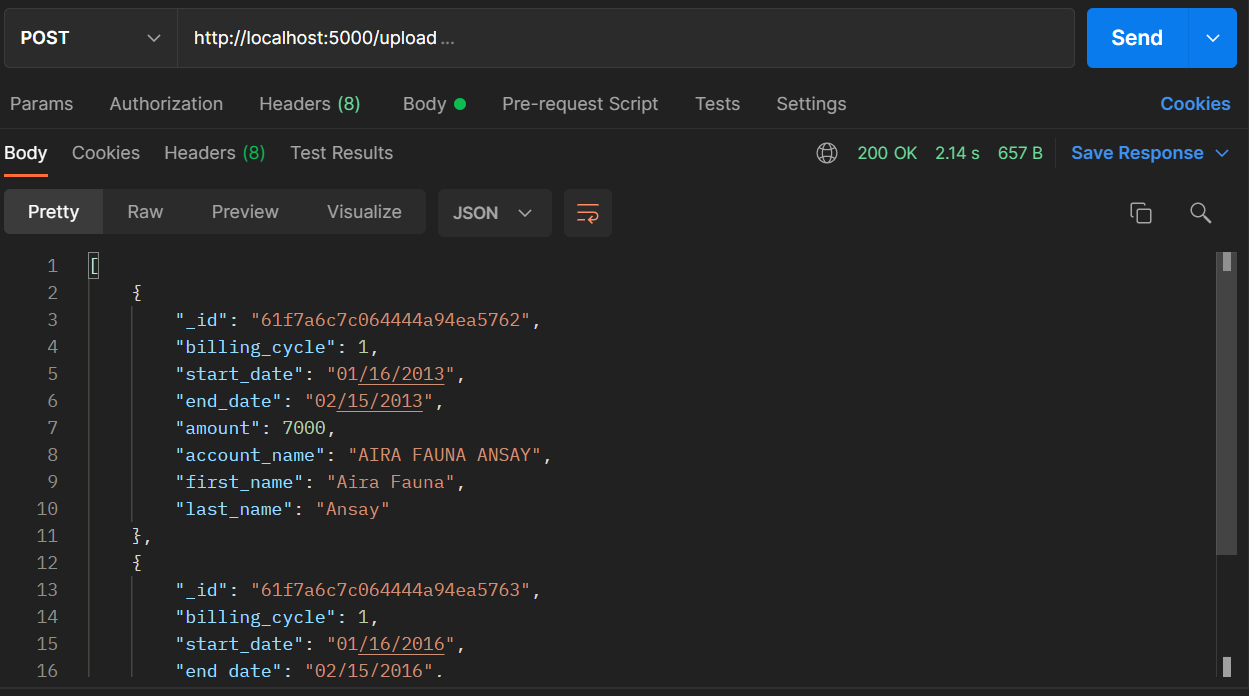


Image 2.1

The image above shows how to send request file to BARS application using any test tool such as Browser, Postman and ARC. The requested file will trigger the application to look in a relative path whether the file is available or not. Once available, the system will process the said request file and return a list of records in JSON format as an HTTP response.

* 1. ***Error Response When File is Not Processed Successfully***

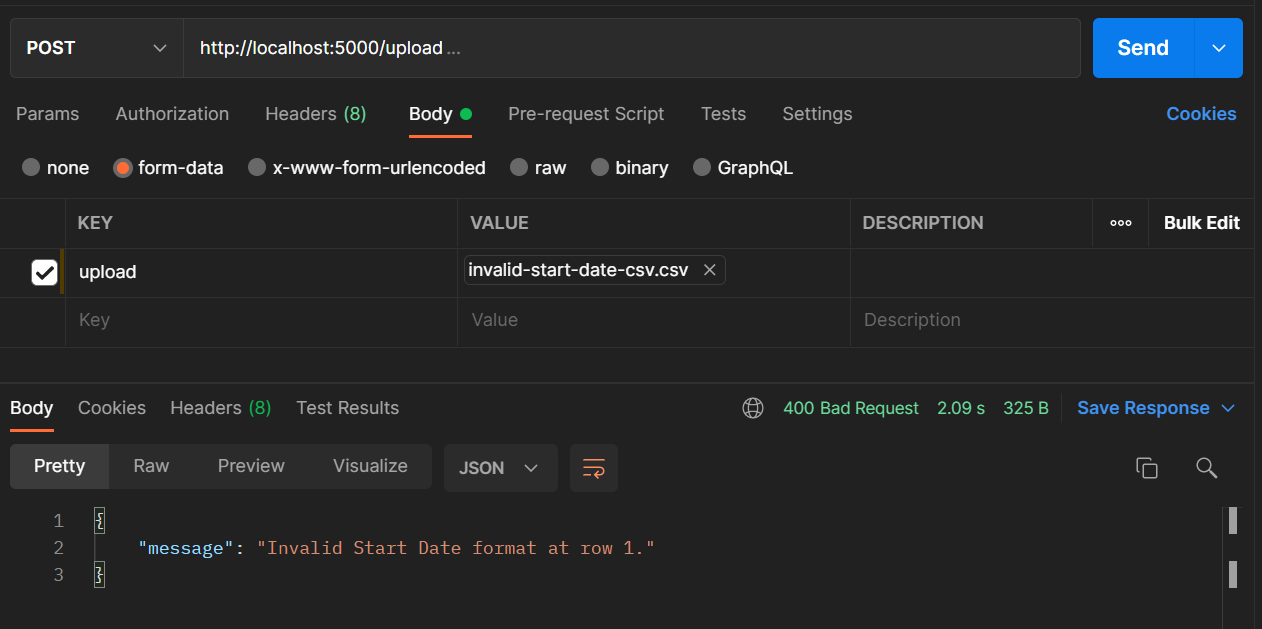


Image 2.2

When the application failed to process the file due to validation errors, the REST endpoint will return an error response with a status code 400.

* 1. ***View Activity Logs in Console/CLI***

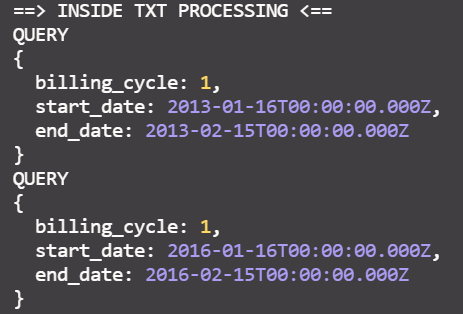


Image 2.3

The image above shows the activity logs of the BARS application from end-to-end process. Any errors encountered by the system will also be displayed. The importance of logs is to serve as a red flag if something unexpected happened during the process of the system as well as check whether the process has been executed successfully.

# Business Rule

## Send Request File through Postman and Process File

Below are the functional rules for the Input Request File Path operation.

* + 1. The user will be able to input the endpoints through postman/web browser
    2. Endpoints will be validated whether it is a single file
    3. The application will check if the request file is in TXT or CSV format. If not, it will display a message in console **“File is not supported for processing” and error response in postman with HTTP status code 400.**
    4. The user will have the option to send a file with Billing Cycle, Start Date and End Date data.
    5. Occurrences of record in file will be validated and will be processed accordingly.
    6. If the user chose to send CSV or TXT files with three parameters (Billing Cycle, Start Date, and End Date), the format must be followed all throughout the records in the file. Consistency must be observed.
    7. The request file should contain the following fields:

Billing Cycle range is 1 – 12

Start Date format depends on the file type

End Date format depends on the file type

* + 1. Contents of the request files will be saved into the database.
    2. If the Billing Cycle is not on range, display an error message: **“ERROR: Billing Cycle not on range at row *<RowNumber>*.”**
    3. If the Start Date has invalid format, display an error message: **“ERROR: Invalid Start Date format at row *<RowNumber>.*”**
    4. If the End Date has invalid format, display an error message: **“ERROR: Invalid End Date format at row *<RowNumber>.*”**
    5. Request Formats:

If uploaded file is in TXT format, the alloted space for the following fields are:

Billing Cycle 2 spaces Ex. 01

Start Date 8 spaces Ex. 01012012

End Date 8 spaces Ex. 01012012

If uploaded file is in CSV format, the alloted space for the following fields are:

Billing Cycle 1st column Ex. 1

Start Date 2nd column Ex. MM/DD/YYYY

End Date 3rd column Ex. MM/DD/YYYY

If the file contains no request(s) (empty file), display a message: **“No request(s) to read from the input file.”**

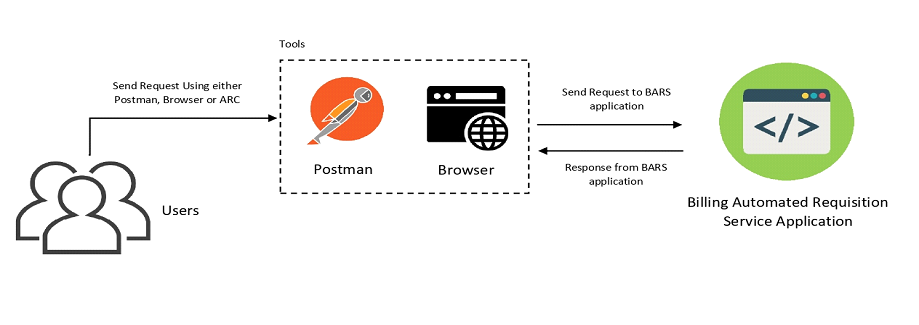
* + 1. If any of the above errors have encountered, stop reading from the current input file.

## Receive response in Postman and Display Activity Logs in CLI

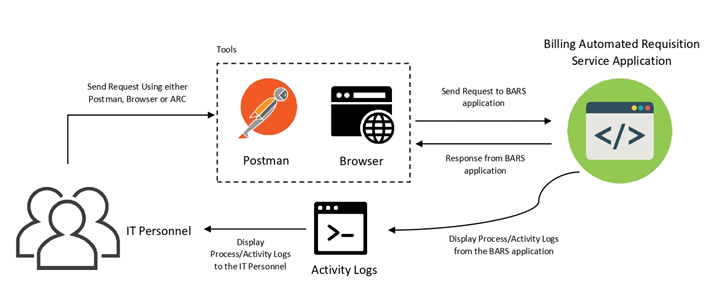
Below are the generic business rules for the Generate Output and Display Activity Logs in CLI operation.

* + 1. User should receive the correct JSON Response output.
    2. Any errors that occur should be handled properly and should return the correct error response.
    3. If there are no resulting record(s) from the database (empty result set), display the message: **“No record(s) to write to the output file.”**
    4. The list of record(s) retrieved from the database will be displayed in the CLI with the message “**Successfully processed Request File**”**.**

# Functional Workflow Diagram



4.1 User Functional Workflow Diagram



4.2 IT Personnel Functional Workflow Diagram