**File Upload Functionality**

**Component Specification Document**

**V1.0**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** |
| **Name** |  |  |  |
| **Role** |  |  |  |
| **Signature** |  |  |  |
| **Date** |  |  |  |

**Table of Contents**

[1.0 Abstract 3](#_Toc114091527)

[1.1 Technology/Framework for Development 3](#_Toc114091528)

[2.0 Business Scenario 3](#_Toc114091529)

[Component Features 4](#_Toc114091530)

[3.0 Sample Approach 5](#_Toc114091531)

[4.0 Assumptions 5](#_Toc114091532)

[5.0 Dependencies & Constraints 5](#_Toc114091533)

[6.0 Exception Handling 5](#_Toc114091534)

# Abstract

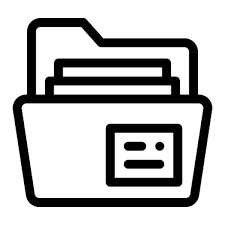
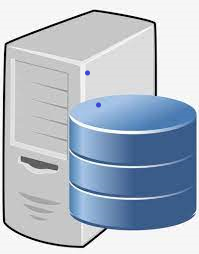
The File Upload component is a class library which helps the developers to reuse the service part to upload the data. This component is built in a generic / reusable way so that it can be integrated with any applications of same tech stack since File Upload functionality is commonly used in almost all application development. Development effort will be reduced by using this component.

## Technology/Framework for Development

Class library of any tech stack.

# Business Scenario

This component is usable when there is scenario of developing file upload functionality in various applications. This component can be used in 2 business scenarios like Reusable within same application and Reusable across applications. In the application, file upload functionality will be present in multiple places as well as this may be having different set of columns and number of columns may vary as per the requirement. So, it will be more useful if we can create this component coming up with all validation criteria’s configured in the config file and once any upload related issues fixed in this component, then it should be fixed across all the places where ever this file upload functionality is used.



**Web Client**

**Web Server**

**DB Server**

**Local File System**

File Upload Service Call - POST / File



If all Validations Passed

Success / Validation Message

Store Uploaded Data

**Users**

User Selects the file

File Stream

# Component Features

To use this component as a developer

* Provision to handle file uploads done by multiple users.
* Provision to upload any number of columns till max. of 20 which can be customized in component’s config file for various modules.
* Provision to do Basic file upload validations which can be customized in component’s config file for various modules.

# Sample Approach

Below is one of the sample approach:

**Front End:** Users selects any file of types Excel, CSV and XML from local file system and click on Upload button. Upon upload button click, file content will be read and converted to JSON data post which File Upload Service post method will be called where the file upload JSON data is sent as parameter.

**File Upload Service:** File Upload post method which in turn calls file upload library / component to perform the validations on the received File Upload JSON data. If all validations passed, then upload the data in DB and return back the Success message to Front End. If validations fails, return back the appropriate validation message to Front End.

# Assumptions

This component supports basic validations of file upload feature mentioned in Dependencies & Constraints section. Developers should customize the code for project specific additional business validations.

# Dependencies & Constraints

* This should be a reusable module which can be easily integrated into any applications of same tech stack.
* The following input parameters need to send by UI during file upload service call.
  + Module Name for which file upload should be done. Ex., Project, User Uploads, etc.,
  + JSON which holds data that are read from the files.
* The file upload functionality should have the following constraints
  + Files must not exceed 10 MB.
  + The allowed file format should be XLSX, XLSM, CSV and XML.
  + File should support maximum 20 columns data.
  + The system should ignore blank columns which are in between the columns having the data.
  + The system should be able to check whether the columns are mandatory or not based on the mandatory configuration done for each column in config file.
  + Unique Key column which is defined in config file should not be duplicated.
  + The system should be able to eliminate exact duplicate records before bulk insert or update.
  + The system should validate the column length which is defined in config file.
  + The system should validate the data type of each column which is defined in config file.
  + The system should be able to validate whether the date / date time column adheres to specific date / date time format (ie., MM/DD/YYYY or DD/MM/YYYY, etc.,) which is configured in config file.

# Exception Handling

* Exception appearing for any Timeout issues or for any failure should be logged into the application logs.
* Exceptions which are appearing due to incorrect data/format which are user specific exception has to be sent back to UI with user friendly messages.

**Roles and Responsibilities**

|  |  |  |
| --- | --- | --- |
| Roles | Member Name |  |
| Scrum Master | Rishika |  |
| Development (Backend) | Sai Srinivas Vara Prasad Korlam |  |
| Development (frontend) | Manideep Reddy Sripathi |  |
| Test Engineer | Gaurav Vitrag |  |
| QA Engineer | Vasanthidevi |  |