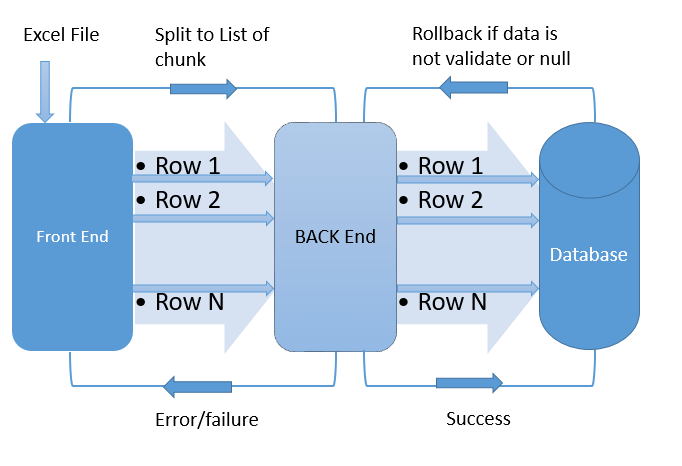
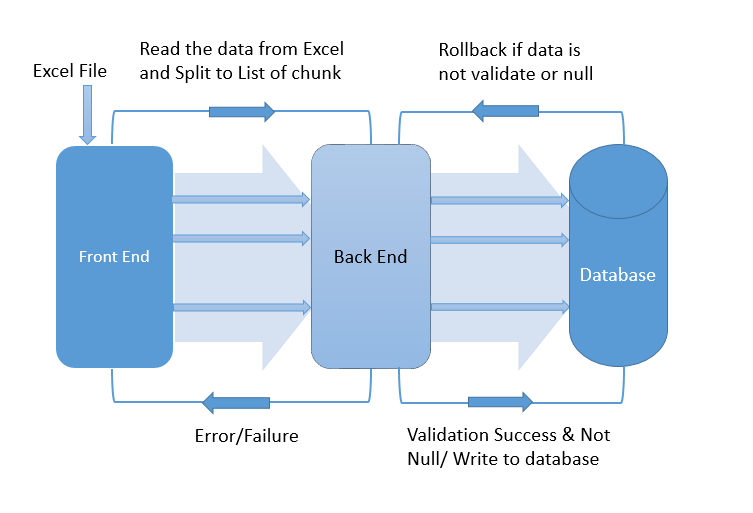
**Architecture for Excel Bulk Upload**

**Figure: 1**



**Figure: 2**



**Flow chart for Excel Bulk Upload (Back End)**

File

Get the data in List of String array & UID

from front end

Upload Controller

**Batch Service Facade**

(Get the chunk list & set UID. Count the row & launch the job according to that)

Spring Batch Excel Configuration

**DB Writer**

Written in desired location

**Processor**

Manipulate & Validation

**List Reader**

Read the data

Database

**Why Not A File:**

In this system, the backend, specifically the Upload Controller, is designed to accommodate input data in a format different from the conventional approach, as it does not accept multipart files (such as Excel or CSV files). Instead, it requires a List of String arrays. This unconventional method was adopted to mitigate the risk of server node overload, especially when dealing with relatively large file uploads. The frontend, implemented using Angular, plays a crucial role in this process by breaking down the uploaded file into smaller, manageable chunks. By receiving these chunks as input, the system can effectively prevent server overload.

**Upload Controller:**

The Upload Controller receives input parameters in the form of a Load Request object, which contains two variables: a List of String arrays and a UID (Unique Identifier). The List of String arrays represents the Excel data that has been parsed and divided into smaller segments by the frontend. The UID serves as a unique identifier for each uploaded file. This UID plays a vital role in distinguishing different sets of data and facilitates the ability to perform database rollback operations in case of file corruption or data integrity issues.

**Job:**

In the context of this system, the term "Job" denotes a specific task that needs to be executed using the Spring Batch framework. A Job must comprise at least one step, each of which includes an Item Reader, Item Processor, and Item Writer. These components can be configured to read and process data extracted from Excel files. The Item Writer is configured to ensure that the data read from the file is accurately stored in the database. This approach not only optimizes server performance by preventing overload during large file uploads but also enhances data integrity and management through the use of UIDs for tracking and potential rollback operations.

**Flowchart**

Job Launcher Job Step List Reader

(Reads the data from list &Convert to data to be processed.)

Excel Processor

(Manipulated & Validation)

DB Writer

(Data stored or written in desired location)