**Product Management System**

Following are few assumptions which I have considered while implementing this POC.

Functional Requirements:

* **Upload and persist pricing feeds from retail stores using CSV files which contain Store ID, SKU, Product Name, Price, Date**
  + Upload button and file control having the functionality of uploading CSV file
  + Put Validation of the file type for uploading only csv files. Other files should not be allowed to uploaded.
  + CSV file needs to checked for blank / Invalid data. If csv file is having incorrect fields it should not be allowed to upload.
  + Show notification of the successful upload.
  + On successful record insert the data available in the csv to the database.
  + Check for existing records in the CSV files before dump it to database.
  + **Query needs to be asked to client:** Should we overwrite the same records or add multiple Products with same product name but different price?
  + Display the data in the grid with paginations.
* **Search for pricing records using various criteria and be able to edit/save changes to any record**
  + Provide search option to search the data based on the pricing range. Provide the filter for Price column.
    - Here the search can be done in 3 ways.
      1. The client-side filter check on price column.
      2. Separate search for Price and other columns and Price comparisons.
      3. Server-side Search Implementation.
  + Provide Edit button to edit the record. Implement the edit functionality for existing records
    - Edit button in the Grid for Each product record.
    - On clicking of the button edit popup will open.
    - Data would be prefilled with the existing values.
    - User can edit the Product Name and Price and data would be submitted to update the record in the database.
    - User would be notified once the record successfully updated or an error occur while update.

Non-Functional Requirements:

* **Standard set of non-functional requirements you would expect a retail stores chain**

**with 3000 stores across multiple countries**

Understanding: There would be product data which can have different StoreId and different StoreId can be of different Stores situated in different countries.

So, in this case,

* there would be one database table named as Countries having different Countries.
* There would be one Products table where in all the products will be stored with their StoreIds
* There will be one more table having the columns StoreId and CountryId which will be used to relate the stores with particular Countries.

**Assumptions:**

* If user tries to upload CSV having already existing products in the Database, They will be ignored and User will be notified for the same.
* Here 2 fields will be checked for existence of the data, StoreId and SKU.
* There will be different stores of multiple countries
* All stores will be related to a country.
* As of now as per the understanding there can be four roles in the system based on the given requirements.
  + Super Admin
  + Country Admin
  + Store Admin
  + Store Manager

**Source for the implementation:**

* ReactJS: https://react.dev/
* NodeJS: https://www.w3schools.com/nodejs/
* MongoDB: https://www.mongodb.com/docs/v3.4/