Problem Statement:

CSV Upload and Display Application

Objective:

Develop a full-stack application that allows users to upload a CSV file using a ReactJS frontend, processes the file on a Node.js backend, and displays the contents in a tabular format. The application should integrate with a database for storing and retrieving the CSV data.

Task Details:

1. **Frontend Development (ReactJS):**
    - Develop a ReactJS component that enables users to upload CSV files.
    - After the file is uploaded, send it to the backend server.
    - Retrieve the CSV data from the backend and display it in a table format. The table should dynamically adjust to the column headers and rows based on the CSV file's content.
2. **Backend Development (Node.js):**
    - Implement an API endpoint to handle CSV file uploads.
    - Parse the CSV file to extract its content.
    - Store the extracted data in a database (MongoDB or MySQL).
    - Provide an API endpoint to retrieve the stored data for display on the frontend.
3. **Database:**
    - Use either MongoDB or MySQL to store the CSV data.

Constraints:

- You may use any npm library to assist with CSV parsing, file handling, or table rendering.
- You can refer to the library's documentation only. Googling or any other online help is not allowed.
- The time limit for completing this assignment is 3 hours.

Evaluation Criteria:

1. **Functionality:** The application should correctly handle CSV file uploads and display the data in a unified table format, incorporating data from multiple CSV files, including handling common columns.
2. **Code Quality:** Code should be well-organized, readable, and follow best practices.
3. **Data Management:** Proper handling and storage of CSV data in the chosen database.
4. **User Interface:** The frontend should provide a user-friendly experience for uploading and viewing CSV data.

Please ensure that your solution is complete and functional, with a focus on both frontend and backend integration. If you have any questions or need further clarification, feel free to ask.

Examples:

Here are examples of CSV files with some common columns, and how their data should be represented in a unified table format:

CSV File 1 (file1.csv):

Name	Age	Country
Alice	25	USA
Bob	30	Canada

CSV File 2 (file2.csv):

Product	Price	Quantity	Country
Laptop	1000	2	USA
Mouse	20	10	Canada

CSV File 3 (file3.csv):

Employee	Salary	Department	Country
John	50000	HR	USA
Sarah	60000	IT	Canada

Unified Table View:

Name	Age	Country	Product	Price	Quantity	Employee	Salary	Department
Alice	25	USA						
Bob	30	Canada						
		USA	Laptop	1000	2			
		Canada	Mouse	20	10			
		USA				John	50000	HR
		Canada				Sarah	60000	IT

In this table:

- Rows where only the `Name`, `Age`, and `Country` columns are populated correspond to data from `file1.csv`.
- Rows where `Product`, `Price`, `Quantity`, and `Country` are populated come from `file2.csv`.
- Rows where `Employee`, `Salary`, `Department`, and `Country` are populated are from `file3.csv`.
- Common columns like `Country` align the data for related entries across files.

This unified view helps to illustrate how data with shared columns can be presented together, even when coming from different CSV files.