# Roster Data Management Project Documentation

## Introduction

The Roster Data Management system is designed to handle large Excel file data efficiently. The system allows users to upload Excel files, store them in a database, compare newly uploaded files with existing database records, remove matching rows, and replace them with updated data. This ensures data consistency and reduces redundancy.

## Features of the Roster Data Management System

### Excel File Handling

* Supports uploading of large Excel files.
* Extracts relevant data from Excel and stores it in a database.

### Database Management

* Stores Excel data in a structured database format.
* Uses MongoDB for data storage due to its scalability and flexibility.

### Data Comparison Algorithm

* Compares new Excel file data with existing records based on a unique identifier (regCode).
* Deletes matching records from the database.
* Inserts updated records from the new Excel file.

### User Interface (Frontend)

* Built using React.js for dynamic and responsive interactions.
* Provides options for file uploads, viewing processed data, and downloading results.

### Backend Processing

* Developed using Node.js and Express.js.
* Implements APIs to handle file uploads, data processing, and database interactions.

### Deployment

* Hosted on Hostinger Cloud Professional for optimal performance.
* Uses a subdomain (roster1.sigvitas.com) for access.

### Security and Authentication

* Implements JWT authentication for secure API access.
* Uses validation mechanisms to prevent unauthorized access.

## Architecture Overview

**The project follows the MERN stack architecture:**

* Frontend: React.js
* Backend: Node.js with Express.js
* Database: MongoDB
* Hosting: Hostinger Web Hosting[Single web hosting] and VPS

## Workflow of the System

### Step 1: Uploading an Excel File

* Users select and upload an Excel file.
* The system reads the file, extracts relevant data, and stores it in MongoDB.

### Step 2: Data Comparison and Update

* The system compares the new file data with the existing records using the regCode column.
* Matching records in the database are deleted.
* The updated rows from the new Excel file are inserted into the database.

### Step 3: Download Processed Data

* Users can download the updated data as an Excel file for further analysis.

### Step 4: Error Handling and Logs

* The system validates data formats before processing.
* Logs errors if the file format is incorrect or missing mandatory fields.

## Algorithm for Data Comparison and Update

1. Start
2. Upload Excel File

* Extract data from the uploaded file.
* Convert it into JSON format.

1. Fetch Existing Data from MongoDB
2. Compare regCode Values

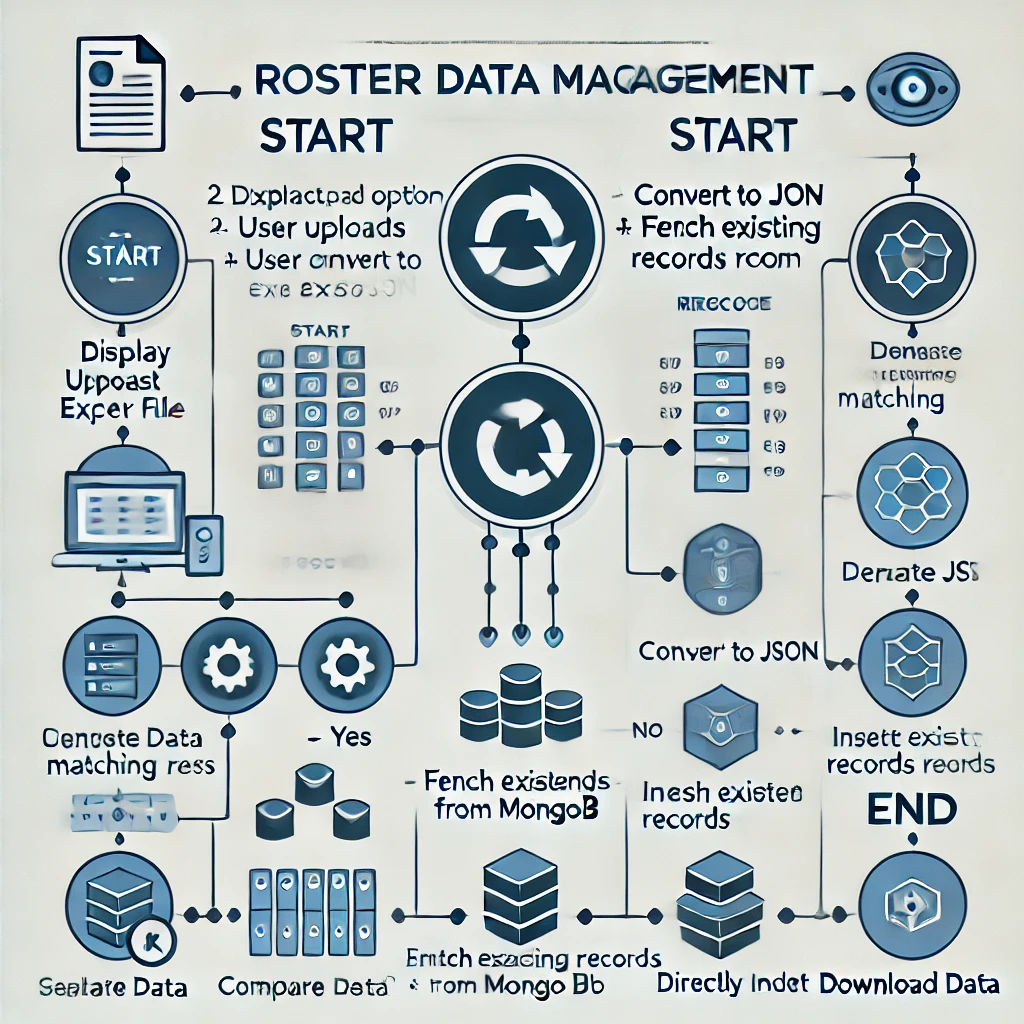
* Identify matching records.
* Delete matched records from the database.

1. Insert New Data into MongoDB
2. Send a Response to the User

* Indicate the number of records added/removed.

1. Enable Data Download Option
2. End

**Flowchart of the System:**



## Deployment Details

* Frontend Hosted on: roster1.sigvitas.com
* Backend Hosted on: Hostinger VPS
* Database: MongoDB Atlas
* Security: JWT Authentication, Secure API Endpoints

## Conclusion

The Roster Data Management system provides an efficient and scalable solution for handling large Excel file data, maintaining updated records, and ensuring data integrity. The project is deployed with a robust MERN stack setup and follows best practices for security and performance optimization.