Project Specification: Campus Carbon Footprint Analyzer

1. Project Objective

You are to build a full-stack, data-driven web application titled **"Campus Carbon Footprint Analyzer."** The primary goal is to provide the **KIT-KalaignarKarunanidhi Institute of Technology** administration and student body with a tool to quantify, track, analyze, and reduce the campus's carbon footprint, in direct support of **UN Sustainable Development Goal 13 (Climate Action)**.

2. Core Technology Stack

This project MUST be built using the following technologies:

- **Frontend: React JS(pure react js without using tailwind css, typescript use only jSX)**
- **Backend: Python** (using the **Flask** framework)
- **Database: MySQL**
- **Visualization: Chart.js** or **Recharts** for interactive graphs.
- **Styling:** A modern UI library (like **Material-UI** or **Tailwind CSS**) to achieve a professional, dark-themed dashboard.

3. User Personas & Authentication

1. **Admin (Campus Staff):**
   - Requires a secure username/password login.
   - Has full access to all modules.
   - **Can:** View the dashboard, add new data, edit/delete existing data.
2. **Public User (Student/Guest):**
   - Does not require a login.
   - Has read-only access.
   - **Can:** Only view the main interactive dashboard and recommendations.

4. Database Schema (MySQL)

You must create the following tables:

1. **users**
   - id (INT, Primary Key, Auto-Increment)
   - username (VARCHAR, Unique)
   - password_hash (VARCHAR)
2. **activity_data** (This table stores all raw consumption data)
   - id (INT, Primary Key, Auto-Increment)
   - date (DATE) - The date the data was recorded for (e.g., '2025-10-01').
   - source_type (VARCHAR) - e.g., 'electricity', 'bus_diesel', 'canteen_lpg', 'waste_landfill'.
   - raw_value (DECIMAL) - The amount consumed (e.g., 120000).
   - unit (VARCHAR) - The unit of measurement (e.g., 'kWh', 'Liters', 'kg').
3. **emission_factors** (This table stores the conversion factors)
   - id (INT, Primary Key, Auto-Increment)
   - source_type (VARCHAR, Unique) - Must match activity_data.source_type.
   - factor (DECIMAL) - The kg CO_2e per unit (e.g., 0.708).
   - factor_unit (VARCHAR) - e.g., 'kg_co2e_per_kwh'.

You must pre-populate the emission_factors table with values like:

- ('electricity', 0.708, 'kg_co2e_per_kwh')
- ('bus_diesel', 2.68, 'kg_co2e_per_liter')

- ('canteen_lpg', 2.93, 'kg_co2e_per_kg')
- ('waste_landfill', 1.25, 'kg_co2e_per_kg')

5. Backend API (Python/Flask)

You will build a REST API to handle all logic.

- Calculation Logic: The core function of the backend is to fetch data from activity_data, join it with emission_factors, and calculate the final emissions:
- Total Emissions (tonnes) = (raw_value * factor) / 1000
- **API Endpoints:**
  - POST /api/login: Authenticates an admin user (sets a JWT token).
  - POST /api/data (Admin Only): Adds a new entry to the activity_data table.
  - GET /api/dashboard: **This is the main public endpoint.**
    - It must accept query parameters for a date range (e.g., ?start_date=YYYY-MM-DD&end_date=YYYY-MM-DD).
    - It must return a single JSON object containing all data needed for the dashboard (KPIs, chart data, etc.).
  - GET /api/recommendations: Returns a list of suggestions based on simple logic (e.g., if "electricity" is the highest source, return "Focus on LED lighting").

6. Frontend UI/UX (React JS)

The frontend must be a single-page application (SPA) with two main views:

A. Admin Data Input Page (/data-input)

- Must be a protected route.
- Must contain a clean, simple form with fields for:
  - Date
  - Source Type (Dropdown: Electricity, Bus Diesel, etc.)
  - Raw Value
  - Unit (Dropdown: kWh, Liters, etc.)
- (Advanced) Include a file upload for CSV data.

B. Main Dashboard Page (/)

- This is the public-facing view. It must be detailed, interactive, and visually match the "final view" image provided in our chat.
- **Key Components:**
  1. **Date Range Selector:** A dropdown/button group to filter the *entire* dashboard (e.g., "Last 6 Months," "Last Year").
  2. **KPI Cards (4 cards):**
     - **Total Emissions:** (e.g., "150 Tonnes CO_2e").
     - **% Change:** Compares Total Emissions to the *previous* period (e.g., "$\downarrow$ 12% vs. Last Year").
     - **Biggest Source:** (e.g., "Energy (68%)").
     - **Total Energy Saved:** (e.g., "25,000 kWh").
  3. **Line Chart (Historical Trend):**
     - **Title:** "Total Emissions Trend"
     - **Data:** Shows the total CO_2e (Y-axis) for each month (X-axis) in the selected date range.
  4. **Donut Chart (Emissions Breakdown):**
     - **Title:** "Emissions Breakdown by Source"
     - **Data:** Shows the percentage split of total emissions (e.g., Energy: 68%, Transport: 25%, Waste: 7%).
  5. **Grouped Bar Chart (Year-over-Year Comparison):**

    - **Title:** "Monthly Emissions (Current vs. Last Year)"
    - **Data:** For each month (Jan, Feb, Mar...), it must show two bars side-by-side: Emissions 2024 and Emissions 2025.
6. **Recommendation Panel:**
    - A simple card that displays the suggestions fetched from the /api/recommendations endpoint.

7. (Optional - Phase 2) AI/ML Integration

If time permits, you will add the following AI features:

1. **Predictive Forecasting:**
    - **Backend:** Create a new endpoint /api/forecast.
    - **Logic:** Use the **SARIMA** time-series model (from the statsmodels Python library) to train on the existing historical data.
    - **Frontend:** The Line Chart ("Total Emissions Trend") should be updated to show a second, *dotted line* representing the forecasted emissions for the next 6 months.
2. **Automated Data Entry (OCR):**
    - **Backend:** Create a new endpoint /api/upload-bill.
    - **Logic:** Use the **Tesseract** OCR library (via pytesseract) to scan an uploaded image (e.g., an electricity bill) and automatically extract the raw_value (kWh) and date.

8. Final Deliverables

1. A complete, working full-stack application.
2. The full source code in a GitHub repository.
3. A README.md file with clear instructions on how to set up the database, install dependencies (for both backend and frontend), and run the project.
4. A schema.sql file to create the database structure.