I'm developing a Carbon Footprint Tracking System for coffee industry branches across multiple Egyptian cities. The system has these core components:

1. System Purpose

- *Objective*: Help coffee manufacturers measure, analyze, and reduce carbon emissions from production, packaging, and distribution
- *Key Features*:
    - Real-time emission calculations
    - Reduction strategy ROI forecasting
    - Multi-branch comparative analytics
    - Role-based access control

2. Technical Specifications

- *Database*: MySQL with 10 core tables including:
    - Branch (16 locations in 4 cities)
    - CoffeeProduction (with supplier/bean type tracking)
    - CoffeeDistribution (vehicle-based emission models)
    - ReductionStrategy (cost/profit projections)
- *Key Database Features*:

sql

Copy

Download

```sql
-- Example of generated columns:
V_CarbonEmissions_Kg DECIMAL(10,2) GENERATED ALWAYS AS (
  ROUND((NumberOfVehicles * DistancePerVehicle_KM /
  CASE WHEN VehicleType = 'Minivan' THEN 10 ELSE 15 END) * 2.68, 2)
) STORED
```

3. User Roles

1. **BranchUser** (per-location access)
2. **OPManager** (cross-branch control)
3. **CIO** (read-only analytics)
4. **CEO** (profitability dashboards)

1. Project Setup Tasks

a. Environment Setup

- Install necessary tools:
    - **XAMPP/WAMP/MAMP** for running PHP and MySQL.
    - A code editor like **VS Code** or **PHPStorm**.
    - Install **phpMyAdmin** for managing databases.
- Create a new folder for your project (e.g., CofaktoryTracker) in your local server root directory (htdocs for XAMPP).

b. Database Setup

- Use the provided SQL schema to set up your database:
    i. Open phpMyAdmin.
    ii. Create a database named carbon_footprint_tracker.
    iii. Import the SQL schema and data into the database. You can use the SQL tab in phpMyAdmin to execute scripts.
    iv. Verify that all tables (e.g., User, Branch, CoffeeDistribution) are created and populated.

---

2. Backend Development Tasks (PHP)

a. User Authentication

- **Task**: Implement user login and registration.
- **How**:
    i. Create a login.php page with a form for email and password.
    ii. Use PHP to verify user credentials against the User table.
        - Hash passwords with password_hash() and verify with password_verify().
    iii. Start a session for authenticated users and store their UserID, UserRole, and BranchID.
    iv. Redirect users based on their roles:
        - BranchUser: Branch-specific dashboard.
        - OPManager, CIO, CEO: Different levels of access.

---

b. Role-Based Access Control

- **Task**: Restrict access to pages based on user roles.
- **How**:
    i. Use $_SESSION variables to check user roles.
    ii. Create middleware-like PHP scripts (e.g., authorize.php) to include on pages for role-based restrictions.

---

c. CRUD Operations

- **Task**: Allow users to perform CRUD operations on entities like Branch, CoffeeDistribution, ReductionStrategy, etc.
- **How**:
  i. Use PHP forms for creating and updating records.
     - Example: A form for adding a new branch should insert data into the Branch table.

  ii. Use prepared statements (PDO or mysqli) to prevent SQL injection.

  iii. Display data in tables using SELECT queries.

  iv. Implement delete functionality with confirmation dialogs.

---

d. Audit Logging

- **Task**: Log all database changes.
- **How**:
  i. Create a PHP function to insert logs into the AuditLogging table.

  ii. Call this function whenever INSERT, UPDATE, or DELETE operations occur.

  iii. Display logs in an admin panel.

---

e. Notifications

- **Task**: Send notifications based on defined conditions.
- **How**:
  i. Insert notifications into the Notification table when specific events occur (e.g., high emissions).

  ii. Create a PHP script to fetch unread notifications for the logged-in user.

  iii. Display unread notifications in a Bootstrap dropdown or modal.

---

f. Reporting and Metrics

- **Task**: Generate reports for carbon footprint metrics.
- **How**:
  i. Write PHP scripts to execute SQL queries, such as:
     - Total carbon footprint by city.
     - Reduction strategies and their statuses.

  ii. Use libraries like **Chart.js** or Google Charts to visualize data in charts.

  iii. Allow users to download reports as PDFs using libraries like dompdf.

---

3. Frontend Development Tasks (HTML, CSS, JS, Bootstrap)

a. Design the UI

- **Task**: Create responsive pages for the system.
- **How**:
  1. i. Use Bootstrap for layout and styling.
  2. ii. Create a navigation bar with links to different sections (e.g., Dashboard, Notifications, Reports).
  3. iii. Use modals for forms (e.g., adding a new branch or coffee production record).

---

b. Dashboards

- **Task**: Create role-specific dashboards.
- **How**:
  - i. Use PHP to fetch relevant data for the logged-in user.
  - ii. Display data in cards, tables, or charts.
    - Example: Show a card for total emissions in a branch.
  - iii. Use JavaScript/jQuery for dynamic interactions (e.g., filtering data).

---

c. Notifications Panel

- **Task**: Show unread notifications.
- **How**:
  - i. Use AJAX to fetch unread notifications without refreshing the page.
  - ii. Mark notifications as read when the user clicks on them.

---

d. Forms and Validations

- **Task**: Ensure all forms are user-friendly and validated.
- **How**:
  - i. Use Bootstrap's form controls for layout.
  - ii. Validate input on the client-side using JavaScript and on the server-side using PHP.

---

e. Charts for Reports

- **Task**: Visualize data in charts.
- **How**:
  - i. Use Chart.js to create bar charts, pie charts, etc.

ii. Fetch data using PHP and pass it to JavaScript in JSON format.

---

4. Database Queries and Stored Procedures

- Implement the provided SQL queries and stored procedures as backend functions in PHP.
- Use AJAX to call these functions and display results dynamically.

---

5. Testing and Debugging

- Test all functionalities:
  i. User login and role-based redirection.
  ii. CRUD operations.
  iii. Notifications and audit logs.
  iv. Reports and charts.
- Debug issues using browser developer tools and PHP error logs.

---

6. Deployment

- Deploy the project on a local server (e.g., XAMPP) for testing.
- For production, use a hosting provider that supports PHP and MySQL (e.g., Hostinger, Bluehost).

---

7. Stretch Goals (Optional Enhancements)

- **Multi-Language Support**: Use PHP resource files for language strings.
- **Data Import/Export**: Add functionality to export reports as CSV or Excel files.
- **Advanced Analytics**: Implement machine learning models for predicting emissions.