Full-Stack Intern Assignment

Reference material::https://hackmd.io/@Nxtwaveccbp/BJuE-zTR1l

Database: You may use SQLite or MongoDB as well.

Task: Build a Bulk Vegetable/Fruit Ordering Platform
Timeline: 1 Week (before 28th Nov.)

Objective

Create a web application to facilitate bulk vegetable/fruit orders. Buyers can browse available products, place bulk orders, and track their status, while admins can manage orders and inventory efficiently.

Requirements

1. Features

For Buyers

1. **Browse Vegetables/Fruits**
   - Display a product catalogue with basic details such as the name and price per unit.
   - No stock tracking is required.
2. **Place Orders**
   - Allow buyers to request bulk orders by specifying the vegetable/fruit, quantity, and delivery details (name, contact information, delivery address).
   - Save the order in the database with a unique identifier.
3. **Order Tracking**
   - Enable buyers to check the status of their placed orders, which should update as:
     - **Pending**: Order has been received.
     - **In Progress**: Order is being processed for delivery.
     - **Delivered**: Order has been delivered successfully.

For Admin

1. **Order Management**
    - View all placed orders with buyer details, delivery address, and the list of requested items.
    - Update the order status (Pending → In Progress → Delivered).
2. **Inventory Management**
    - Add, edit, or remove vegetables/fruits from the catalogue.
    - Note: No stock tracking is required. Assume all requested items are always available.

2. Database

Use **Neon.tech** to host the PostgreSQL database. Or if you know docker, dockerize the postgres db. (You can use any SQL engine but postgres is preferred)

3. Frontend

- Use **Next.js or React.js**
- Build a clean, responsive, and user-friendly interface.
- Implement basic pages for:
    - Product catalogue.
    - Order placement form.
    - Order tracking view.
    - Admin dashboard.

4. Backend

- Implement API routes using **Next.js / Express / Flask** for:
    - Fetching the product catalogue.
    - Placing an order.
    - Viewing and updating order statuses.
    - Managing inventory (admin-only).
- Use appropriate HTTP methods (GET, POST, PUT, DELETE) for each API endpoint.

5. Bonus Features (Optional)

- Add authentication for users and admins (e.g., admin login to access the dashboard).
- Deploy the application on **Vercel** for public access.
- Integrate email notifications to notify buyers of order updates.

Submission Guidelines

1. Share a GitHub repository containing:
    - Source code with a `README.md` that includes setup instructions.
    - A brief description of implemented features.
2. Provide database credentials (or mock data) for testing in an `.env` file.
3. Optional: Deploy the application on **Vercel** and share the live URL.

Evaluation Criteria

1. **Functionality:** How well the platform meets the buyer and admin requirements.
2. **Code Quality:** Clean, modular, and maintainable code.
3. **Database Design:** Efficient schema for handling orders and products.
4. **UI/UX:** Intuitive and responsive user interface.
5. **Completion:** Effort put into implementing required and optional features.