**Project Structure**

* **Django Backend:**
  + Django REST Framework (DRF) will be crucial for creating the API.
  + Django models will map to database tables (using SQLite for simplicity in this example).
  + Serializers will handle data conversion between Python objects and JSON for the API.
  + Django Admin will be used for basic administrative tasks.
  + Will implement custom user authentication using token-based approach.
* **React Frontend:**
  + Built with Create React App or Vite.
  + Shadcn UI for pre-built, accessible UI components.
  + Axios or Fetch for making API requests to the Django backend.
  + React Router for handling navigation.
  + State management (e.g., Context API, Redux, Zustand) for managing data flow.

**Updated Project Requirements (Focusing on React and API)**

Here's a breakdown of how the key features will be adapted:

**1. User Management**

* **Django:**
  + Create User models with fields like role (admin, manager, employee, customer).
  + Create API endpoints for:
    - GET /users: List users (with role-based filtering).
    - POST /users: Create a new user.
    - GET /users/{id}: Get a specific user.
    - PUT /users/{id}: Update a user.
    - DELETE /users/{id}: Delete a user.
  + Implement authentication endpoints using token-based strategy.
* **React:**
  + Use forms with input fields to create and edit users.
  + List users in table format.
  + Implement a search functionality.
  + Use useState hooks to manage user data in the frontend.
  + Use a context to hold current user context, for showing role based features.
  + Implement token based authentication flows.

**2. Product Management**

* **Django:**
  + Create Product models.
  + API endpoints for CRUD operations on products.
  + Endpoints to get product prices by role.
* **React:**
  + Display product lists.
  + Allow users to add to the cart (if customer).
  + Admin/employee can add/edit product via forms.

**3. Order Management**

* **Django:**
  + Create Order models (with shop details, employee, payment terms, location, status).
  + API endpoints for:
    - POST /orders: Create a new order.
    - GET /orders: List orders (with filtering by status, user).
    - GET /orders/{id}: Get specific order.
    - PUT /orders/{id}: Update order (status changes, etc).
  + Handle user authentication for roles.
* **React:**
  + Create order forms for customers and field employees.
  + Display order list with status.
  + Implement status update for manager.

**4. Payment Management**

* **Django:**
  + Implement payment tracking through API endpoints (mark payments as done, etc.)
  + Generate QR code via API.
* **React:**
  + Display payment history.
  + Display payment information along with the order.
  + Show QR code or UPI id.

**5. Sales Reporting**

* **Django:**
  + API endpoint for setting monthly sales targets (with employee ID).
  + API endpoint to get monthly sales report
  + Generate data for visualization.
* **React:**
  + Render report data (targets, pending payments, achievement) in tables or graphs.