Below is a refined and directed project documentation tailored for a Quick Commerce niche use case. This documentation outlines a Smart Quick Commerce Inventory and Demand Prediction System that simulates a simple store (without payment integration) for placing orders while providing a robust management dashboard for real-time inventory tracking and demand forecasting.

**Smart Quick Commerce Inventory and Demand Prediction System**

**1. Project Overview**

**Purpose:**  
The system is designed to serve quick commerce businesses by streamlining inventory management and accurately forecasting product demand. It simulates a simple online store where customers can place orders, and it provides a management dashboard that offers real-time insights into inventory levels and predicted trends based on historical and live data.

**Key Objectives:**

* **Inventory Optimization:** Ensure that products are neither overstocked nor understocked.
* **Demand Forecasting:** Use deep learning neural network models to predict future sales and manage inventory proactively.
* **Operational Efficiency:** Enable quick commerce operations to respond dynamically to customer demands.
* **User Simulation:** Provide a straightforward customer interface for ordering products and a comprehensive admin dashboard for operational oversight.

**2. System Architecture**

**2.1. Technology Stack**

* **Frontend:**
  + **Framework:** React
  + **Styling:** Tailwind CSS
* **Backend:**
  + **Framework:** Java Spring Boot
  + **APIs:** RESTful services to handle business logic and data exchange
* **Database:**
  + **Relational DB:** PostgreSQL (for storing product, sales, and order data)
* **Data Processing:**
  + **Engine:** Apache Spark (for large-scale data processing and transformation)
* **Prediction Model:**
  + **Architecture:** Neural Network (for demand forecasting)
  + **Type:** [Suggested] Time-series model (e.g., LSTM or feed-forward network with time-lagged features)
  + **Training:** Use historical sales data to train the model on Apache Spark, then serve predictions through the backend.

**3. Use Case: Quick Commerce**

**3.1. Simulated Store Experience**

* **Customer Interaction:**
  + A simple storefront where users can browse items and place orders.
  + Order process is straightforward—users select items, add to cart, and place orders without payment integration.
* **Data Capture:**
  + Each order is logged in PostgreSQL, capturing details such as Product ID, Quantity, Order Time, and Customer details (if needed for analysis).

**3.2. Management Dashboard**

* **Real-Time Inventory Monitoring:**
  + Dashboard displays current stock levels with alerts for low-stock or overstocked items.
  + Inventory adjustments are updated live as orders are placed.
* **Demand Prediction Display:**
  + Graphs and charts show historical sales trends alongside neural network–based demand forecasts.
  + Enables management to adjust inventory and plan restocking proactively.
* **Reports and Analytics:**
  + Generate customizable reports on sales performance, inventory turnover, and forecast accuracy.
  + Use interactive charts (bar, line, pie) to visualize trends over selectable date ranges.

**4. Detailed System Components**

**4.1. Frontend Design (React + Tailwind CSS)**

**Pages/Components:**

1. **User Interface for Customers:**
   * **Home/Storefront Page:**
     + List of available products with images, descriptions, and stock status.
     + "Add to Cart" and "Place Order" functionalities.
   * **Order Confirmation Page:**
     + Display order summary with order status ("Order Placed").
2. **Admin Dashboard:**
   * **Login/Authentication Screen:**
     + Secure login for store management.
   * **Inventory Management Page:**
     + Real-time charts and tables showing current stock levels.
     + Functions to add, update, or delete products.
   * **Demand Prediction Page:**
     + Interactive charts displaying historical data and forecasted demand.
     + Dropdown filters for product/category selection.
   * **Reports Page:**
     + Customizable reports for sales, inventory, and predictions.
     + Download options (e.g., CSV, PDF).

**4.2. Backend Structure (Java Spring Boot)**

**Modules and Services:**

1. **Authentication Module:**
   * Manage user login and session handling.
2. **Inventory Service:**
   * CRUD operations for product and inventory data.
   * REST endpoints:
     + **GET /inventory:** Fetch current inventory.
     + **POST /inventory/update:** Update inventory levels.
3. **Order Management Service:**
   * Capture and store order details from the storefront.
   * REST endpoint:
     + **POST /orders:** Record a new order.
4. **Prediction Service:**
   * Interface with the neural network model for demand forecasting.
   * REST endpoint:
     + **GET /predictions:** Retrieve forecasted demand for specified products/categories.
5. **Reporting Service:**
   * Generate analytics and downloadable reports.
   * REST endpoint:
     + **GET /reports:** Retrieve report data based on query parameters (e.g., date range).

**4.3. Database Structure (PostgreSQL)**

**Tables:**

1. **Products:**
   * Fields: ProductID, Name, Category, Price, CurrentStock, etc.
2. **Orders:**
   * Fields: OrderID, CustomerID, OrderDate, ProductID, Quantity, OrderStatus.
3. **Sales:**
   * Fields: SalesID, ProductID, Date, QuantitySold (used for historical trend analysis).
4. **Predictions:**
   * (Optional) A table to log forecast outputs with timestamps for auditing model performance.

**4.4. Data Processing Pipeline (Apache Spark)**

1. **Data Ingestion:**
   * Stream real-time order data and batch upload historical sales data from PostgreSQL.
2. **Data Transformation:**
   * Clean and prepare data for model training using Spark jobs.
   * Aggregate and compute necessary features (e.g., moving averages, seasonality factors).
3. **Model Training and Serving:**
   * Train the neural network model on historical data.
   * Schedule periodic retraining using Spark pipelines.
   * Serve the model predictions via the backend’s Prediction Service.

**4.5. Demand Prediction Model (Neural Network)**

**Model Architecture:**

* **Input Layer:**
  + Time-series features such as date, product attributes, promotions, seasonality indices.
* **Hidden Layers:**
  + 2-3 layers (e.g., LSTM layers for sequence modeling or Dense layers for regression).
  + Use dropout layers to prevent overfitting.
* **Output Layer:**
  + A regression node that outputs the predicted sales quantity.

**Training Details:**

* **Dataset:**
  + Use historical sales data from the PostgreSQL Sales table.
* **Training Split:**
  + 70% for training, 20% for validation, 10% for testing.
* **Evaluation Metrics:**
  + Mean Absolute Error (MAE)
  + Root Mean Squared Error (RMSE)

**5. Data Flow and Interaction**

1. **Customer Side:**
   * User visits the storefront → selects items → places an order.
   * Order data is sent to the backend (Spring Boot) and stored in PostgreSQL.
2. **Backend Processing:**
   * Order details update the Inventory Service.
   * Sales data is forwarded to Apache Spark for real-time and batch processing.
3. **Prediction Workflow:**
   * Spark processes updated data and feeds features into the neural network.
   * The Prediction Service retrieves model outputs and provides them via REST endpoints.
4. **Management Dashboard:**
   * Dashboard components fetch real-time inventory, sales, and prediction data from the backend.
   * Admins use this information to make inventory adjustments and strategic decisions.

**6. Future Enhancements**

* **Integration of IoT:**
  + Incorporate IoT sensors for real-time physical inventory tracking.
* **Enhanced Analytics:**
  + Develop more complex dashboards with AI-driven recommendations.
* **Model Improvement:**
  + Experiment with additional deep learning architectures (e.g., transformer-based models) for demand prediction.
* **User Feedback Integration:**
  + Collect and analyze customer feedback to refine product offerings and inventory strategies.

This refined documentation sets a clear direction for your mini project by focusing on a Quick Commerce use case. It outlines the critical components, technology stack, system architecture, and data flow necessary for building a comprehensive Smart Retail Inventory and Demand Prediction System.