**AI-Powered Demand Forecasting and Inventory Optimization System**

**Abstract**

This project outlines the development of an automated demand forecasting and stock management system. The application will ingest historical sales data, generate SKU-level forecasts using advanced time-series models, and provide prescriptive reorder recommendations. The system will feature a Python backend, a MongoDB database, and a user-friendly dashboard built on a low-code platform.

**1. Problem Statement**

Manual stock planning is inefficient and prone to error, leading to significant operational challenges:

* **Overstocking:** Ties up capital and increases holding costs.
* **Stockouts:** Results in lost sales and poor customer satisfaction.
* **Inefficiency:** Consumes significant manual effort better spent on strategic activities.

This project will engineer an intelligent system to automate and optimize inventory decisions by learning from historical data.

**2. Project Objectives**

1. **Develop an ETL Pipeline** to ingest, clean, and process retail and e-commerce data.
2. **Implement a 3-Tier Modeling Approach** (Descriptive, Predictive, Prescriptive) to analyze data, forecast demand with confidence intervals, and calculate optimal stock levels.
3. **Build a Backend System** with a REST API to manage data, model inference, and business logic.
4. **Create an Interactive Frontend** to display forecasts, KPIs, and actionable alerts.
5. **Evaluate and Benchmark** model performance through rigorous backtesting against baseline methods.

**3. Selected Datasets**

* **Primary Dataset: Retail Store Inventory Forecasting (Kaggle)**
  + A clean, synthetic dataset with daily sales, inventory, pricing, weather, and promotions. Ideal for end-to-end model development.
* **Secondary Dataset: E-Commerce Sales (Kaggle)**
  + A real-world dataset with SKU codes, stock levels, and sales data. Used to validate the model's generalizability to an online retail context.

**4. System Architecture**

The system uses a modern 3-tier architecture for scalability and separation of concerns.

* **Frontend (Presentation Layer)**
  + **Technology:** Typescript, Reacr(Vite.js), AI tools
  + **Purpose:** User interaction, data visualization, and displaying recommendations.
* **Backend (Application Layer)**
  + **Technology:** Python (FastAPI), MongoDB.
  + **Purpose:** Exposing a REST API, data management, and orchestrating model calls.
* **Modeling Engine (Intelligence Layer)**
  + **Technology:** Python
  + **Purpose:** Data processing, model training, and generating forecasts.

**5. Project Methodology**

**Phase 1: ETL and Feature Engineering**

* Load Kaggle datasets into a MongoDB database.
* Clean and preprocess data.
* Engineer features: time-based (day of week, month), lag (sales last week), rolling windows (7-day average), and external factors (holidays, promotions).

**Phase 2: Model Development**

* **Descriptive:** Decompose time-series data into trend, seasonality, and residuals using statsmodels.
* **Predictive:**
  + **Baseline Model:** SARIMA for performance comparison.
  + **Primary Model:** XGBoost for its high performance and ability to handle causal variables.
  + **Confidence Intervals:** Use Quantile Regression to forecast a realistic demand range.
* **Prescriptive:**
  + Calculate **Safety Stock** and **Reorder Points (ROP)** based on forecasts and desired service levels.
  + Recommend **Order Quantities** using a "days of supply" or EOQ model.

**Phase 3: Backend and API Development**

* Develop REST API endpoints for data ingestion, forecasting, and recommendations.
* Implement orchestration logic to connect API requests to the modeling engine.

**Phase 4: Frontend Development**

* Integrate the platform with the backend API.
* Build the user dashboard, SKU-level forecast explorer, and recommendation tables.

**Phase 5: Evaluation and Deployment**

* **Backtest** the model on unseen historical data to validate accuracy.
* **Benchmark** the XGBoost model against the SARIMA baseline using MAPE and RMSE.
* **Deploy** the application using Docker containers.

**6. Expected Deliverables**

1. A fully functional **prototype application**.
2. A documented **REST API**.
3. Versioned and **trained machine learning models**.
4. A final **project report** detailing the methodology, results, and insights.