You want to build a **Sales Forecasting System** that predicts **future sales** using **historical sales data** — categorized by **Territory, Area, and Region**, with **Year-Month granularity**.  
The system will help management plan inventory, set sales targets, and identify regional growth opportunities.

**🚀 Key Components & Features:**

1. **Data Source:**
   * SQL Server database with historical sales records.
   * Dimensions: Region, Area, Territory, Year, Month, and SalesAmount.
2. **Backend Framework:**
   * **Flask API** for model training and prediction endpoints.
   * RESTful design with routes like /train, /forecast, /health.
3. **Machine Learning Models:**
   * **Prophet (Facebook)** → Best for time-series forecasting per Territory/Region.
   * **XGBoost / LightGBM** → Capture additional seasonal or categorical patterns.
   * **Hybrid Approach:** Combine Prophet’s time-series trend + LightGBM’s boosting power.
4. **Architecture:**
5. /sales\_forecast\_app
6. ├── app.py # Flask main file
7. ├── models/ # ML models (Prophet, LightGBM, etc.)
8. ├── services/ # Training & forecasting logic
9. ├── database/ # SQL Server connection & queries
10. ├── utils/ # Data preprocessing, feature engineering
11. ├── static/ & templates/ # Optional dashboard or visual UI
12. └── requirements.txt
13. **Data Handling Best Practices:**
    * Aggregate sales by Region → Area → Territory hierarchy.
    * Sales by Year Month.
    * Normalize or standardize numerical fields.
    * Handle missing or outlier values before model training.
    * Use rolling averages, lag features, and time-based windows for richer model inputs.
14. **Model Training & Evaluation:**
    * Prophet for forecasting each Territory’s time series.
    * LightGBM/XGBoost trained on engineered features.
    * Evaluate using **RMSE**, **MAE**, and **MAPE**.
    * Cross-validation using **TimeSeriesSplit** (not random split).
15. **API Workflow:**
    * /train → Trains models using data from SQL Server.
    * /forecast → Predicts future months’ sales (e.g., next 3–12 months).
    * /status → Health check for API and model readiness.
16. **Deployment & Scalability:**
    * Use **Uvicorn** for production deployment.
    * Schedule periodic retraining (e.g., monthly) using cron or Airflow.
17. **Visualization (Optional):**
    * Build a simple **Flask Dashboard** to visualize forecasts per Region/Area.