Inventory Demand Forecasting Tool

One-line project description:
A SKU-level demand forecasting web application that enables small and medium businesses to make data-driven inventory and restocking decisions using time-series modeling.

---

1. Problem Statement

Problem Title

Inventory Demand Forecasting Tool

Problem Description

Small and medium businesses typically rely on Excel averages or manual estimation for inventory planning. These approaches ignore trend shifts, seasonal patterns, and demand variability, leading to inaccurate forecasts and inefficient stock management.

Target Users

- Small/Medium retail store owners
- E-commerce sellers
- Warehouse managers
- Inventory planners in SMEs

Existing Gaps

- No structured time-series modeling
- No SKU-level forecasting automation
- No uncertainty estimation (confidence intervals)
- No scientific restocking logic
- Over-reliance on spreadsheet averages

- Doesn’t detects trends (sales increasing/decreasing over time)
- Can’t detects trends (sales increasing/decreasing over time)

---

2. Problem Understanding & Approach

Root Cause Analysis

- Lack of technical forecasting tools
- No trend and seasonality decomposition
- No safety stock calculation
- Reactive decision-making instead of proactive planning

Solution Strategy

- Accept structured historical sales data via CSV
- Perform SKU-level time-series modeling
- Decompose demand into trend, seasonality, and residual
- Forecast future demand using Holt-Winters
- Provide restocking recommendations using statistical logic

---

3. Proposed Solution

Solution Overview

A web-based Inventory Demand Forecasting Tool that processes historical sales data and generates SKU-level forecasts with confidence intervals and automated restocking recommendations.

Core Idea

Combine interpretable time-series forecasting with practical inventory decision logic in a simple and accessible web interface.

Key Features

- CSV-based data upload
- SKU-level demand forecasting
- Holt-Winters time-series modeling
- Trend and seasonality capture
- Forecast confidence intervals
- Reorder point calculation
- Restock recommendation engine
- Interactive dashboard visualization

---

4. System Architecture

High-Level Flow

User → Frontend (React) → Backend (FastAPI) → Forecast Model (Statsmodels) → Response → Dashboard Visualization

Architecture Description

A[User]

B[Frontend
(React / Next.js)]

C[Backend API
(FastAPI / Node)]

D[Data Validation Layer]

E[Preprocessing Engine]

F[Time-Series Decomposition]

G[Forecasting Engine
(Holt-Winters / Moving Average)]

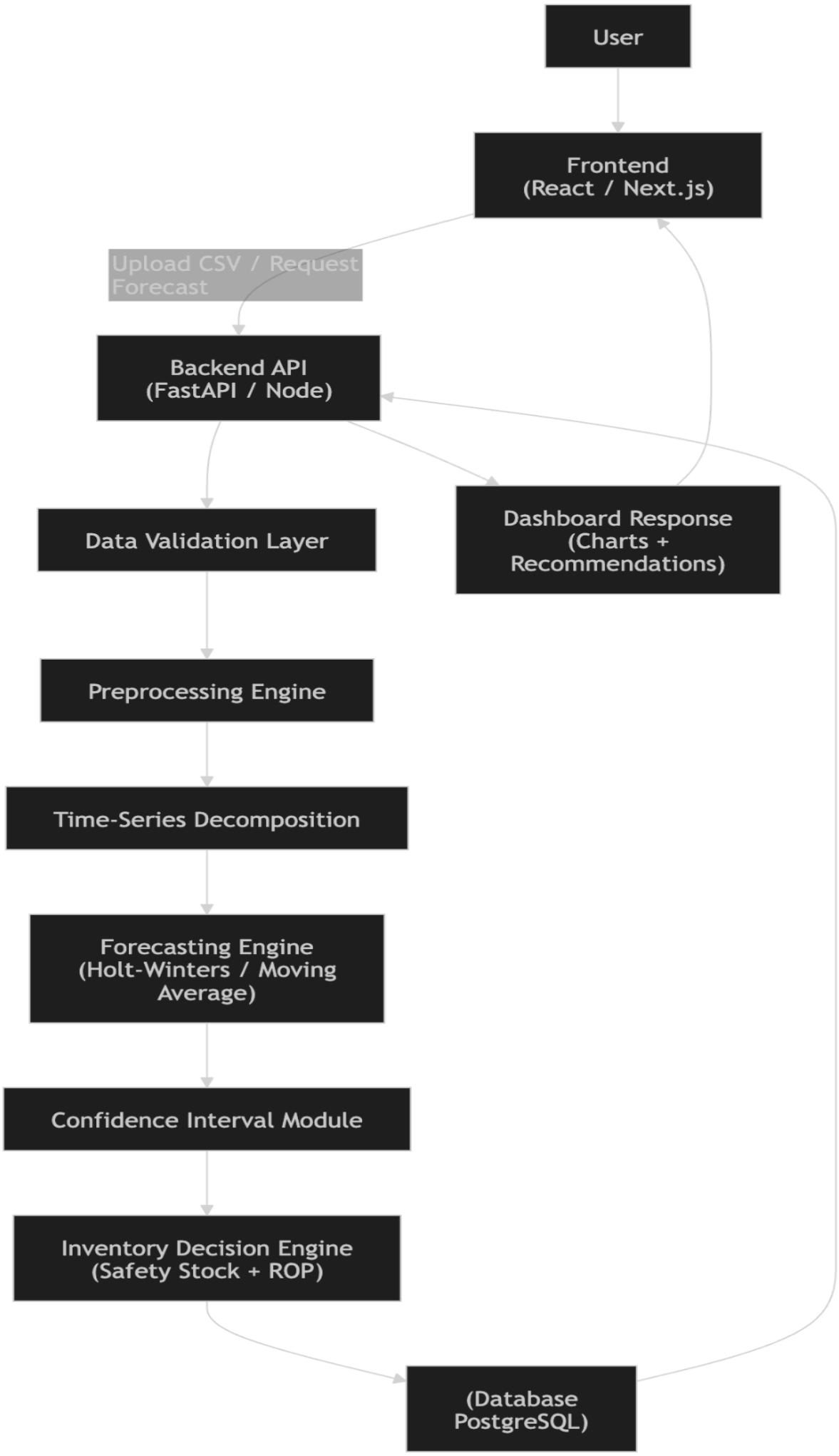
H[Confidence Interval Module]

I[Inventory Decision Engine
(Safety Stock + ROP)]

J[(Database
PostgreSQL)]

```
K[Dashboard Response<br/>(Charts + Recommendations)]


A --> B

B -->|Upload CSV / Request Forecast| C

C --> D

D --> E

E --> F

F --> G

G --> H

H --> I

I --> J

J --> C

C --> K

K --> B
```

Architecture Diagram


User
Frontend
(React / Next.js)
Upload CSV / Request Forecast
Backend API
(FastAPI / Node)
Data Validation Layer
Dashboard Response
(Charts + Recommendations)
Preprocessing Engine
Time-Series Decomposition
Forecasting Engine
(Holt-Winters / Moving Average)
Confidence Interval Module
Inventory Decision Engine
(Safety Stock + ROP)
(Database
PostgreSQL)

---

5. Database Design

ER Diagram

```
erDiagram

    SKUS {
        int sku_id PK
        string product_name
        int lead_time
        float service_level
        float holding_cost
        datetime created_at
    }

    SALES {
        int sale_id PK
        int sku_id FK
        date sale_date
        int quantity
        datetime created_at
    }

    FORECASTS {
        int forecast_id PK
        int sku_id FK
```

```
    date forecast_date
    float predicted_value
    float lower_bound
    float upper_bound
    string model_used
    datetime generated_at
}

INVENTORY_RECOMMENDATIONS {
    int recommendation_id PK
    int sku_id FK
    float safety_stock
    float reorder_point
    float recommended_order_qty
    date recommended_order_date
    datetime generated_at
}

SKUS ||--o{ SALES : "has"
SKUS ||--o{ FORECASTS : "generates"
SKUS ||--o{ INVENTORY_RECOMMENDATIONS : "produces"
```

ER Diagram Description

Entities:

1. SKU

    - sku_id
    - sku_name
2. Sales
    - date
    - sku_id (foreign key)
    - units_sold
3. Forecast
    - sku_id
    - forecast_date
    - forecast_value
    - upper_ci
    - lower_ci

Relationships:

- One SKU → Many Sales Records
- One SKU → Many Forecast Records

---

6. Dataset Selected

Dataset Name

Store Item Demand Forecasting Dataset

Source

Kaggle

Data Type

Daily retail sales data (time-series format)

Selection Reason

- Clean structured retail dataset
- Multiple SKUs
- Suitable for seasonal modeling
- Ideal for hackathon MVP

Preprocessing Steps

- Parse Date column to datetime
- Sort by SKU and Date
- Handle missing values
- Fill missing dates where required
- Remove anomalies if extreme

---

7. Model Selected

Model Name

Holt-Winters Exponential Smoothing

Selection Reasoning

- Captures trend and seasonality
- Works well on retail demand data
- Interpretable and lightweight
- Suitable for small datasets

Alternatives Considered

- Moving Average (too simplistic)

- ARIMA (more complex tuning required)
- Prophet (external dependency and heavier model)

Evaluation Metrics

- MAE (Mean Absolute Error)
- RMSE (Root Mean Squared Error)
- Visual inspection of forecast trend

---

8. Technology Stack

Frontend

- React (Vite)
- HTML
- CSS
- JavaScript

Backend

- Python
- FastAPI
- Pandas
- NUM.py

ML/AI

- Statsmodels

Database

- In-memory (Pandas)
- Extendable to PostgreSQL
- CSV (for row data)

Deployment

- Backend: Render / Railway / aws
- Frontend: Vercel

---

9. API Documentation & Testing

API Endpoints List

Endpoint 1: Upload CSV

POST /upload

Request:
Multipart CSV file

Response:
List of available SKUs

Endpoint 2: Forecast

POST /forecast

Request Body:
{
"sku": "A101",
"forecast_days": 30
}

Response:
{

"dates": [],
"actual": [],
"forecast": [],
"upper_ci": [],
"lower_ci": [],
"reorder_point": number,
"restock_recommended": true/false
}

API Testing Screenshots

---

10. Module-wise Development & Deliverables

Checkpoint 1: Research & Planning

Deliverables:

- Problem definition
- Dataset selection
- Architecture design

Checkpoint 2: Backend Development

Deliverables:

- FastAPI server
- CSV upload endpoint
- Forecast endpoint
- Restock logic implementation

Checkpoint 3: Frontend Development

Deliverables:

- Upload UI
- SKU selector
- Forecast dashboard
- Chart integration

Checkpoint 4: Model Training

Deliverables:

- Holt-Winters implementation
- Confidence interval calculation

Checkpoint 5: Model Integration

Deliverables:

- API integration
- End-to-end testing
- Error handling

Checkpoint 6: Deployment

Deliverables:

- Backend deployed
- Frontend deployed
- Final testing

---

11. End-to-End Workflow

Upload CSV

↓

Data Validation

↓

Preprocessing

↓

Decomposition

↓

Model Selection

↓

Forecast Generation

↓

Confidence Interval

↓

Inventory Recommendation

↓

Dashboard Visualization

---

12. Demo & Video

Live Demo Link:

Demo Video Link:

GitHub Repository:

---

13. Hackathon Deliverables Summary

- Working web application
- SKU-level forecasting
- Confidence interval visualization
- Automated restock logic
- Clean UI dashboard
- Deployed version
- Complete documentation

---

14. Team Roles & Responsibilities

Member Name	Role	Responsibilities
Kshitiz Raj	Backend Developer	API design, model implementation
Yash Raj	Frontend Developer	UI development, chart integration
Vaibhav Singh	ML Engineer	Forecast tuning, evaluation

---

15. Future Scope & Scalability

Short-Term

- Auto model selection
- Multi-SKU batch forecasting
- Downloadable PDF reports
- Email alerts

Long-Term

- Multi-store forecasting
- Real-time POS integration
- Advanced ML models (ARIMA, Prophet, LSTM)
- Cloud-based scalable infrastructure
- Role-based access control

---

16. Known Limitations

- Assumes additive seasonality
- Limited performance on highly irregular demand
- Requires minimum historical data
- No real-time database in MVP
- Basic safety stock logic

---

17. Impact

Business Impact:

- Reduced stockouts
- Reduced overstock costs
- Improved cash flow management
- Data-driven decision making
- Scalable inventory planning framework

Technical Impact:

- Demonstrates applied time-series modeling

- Combines forecasting with decision engine
- Practical ML deployment for SMEs