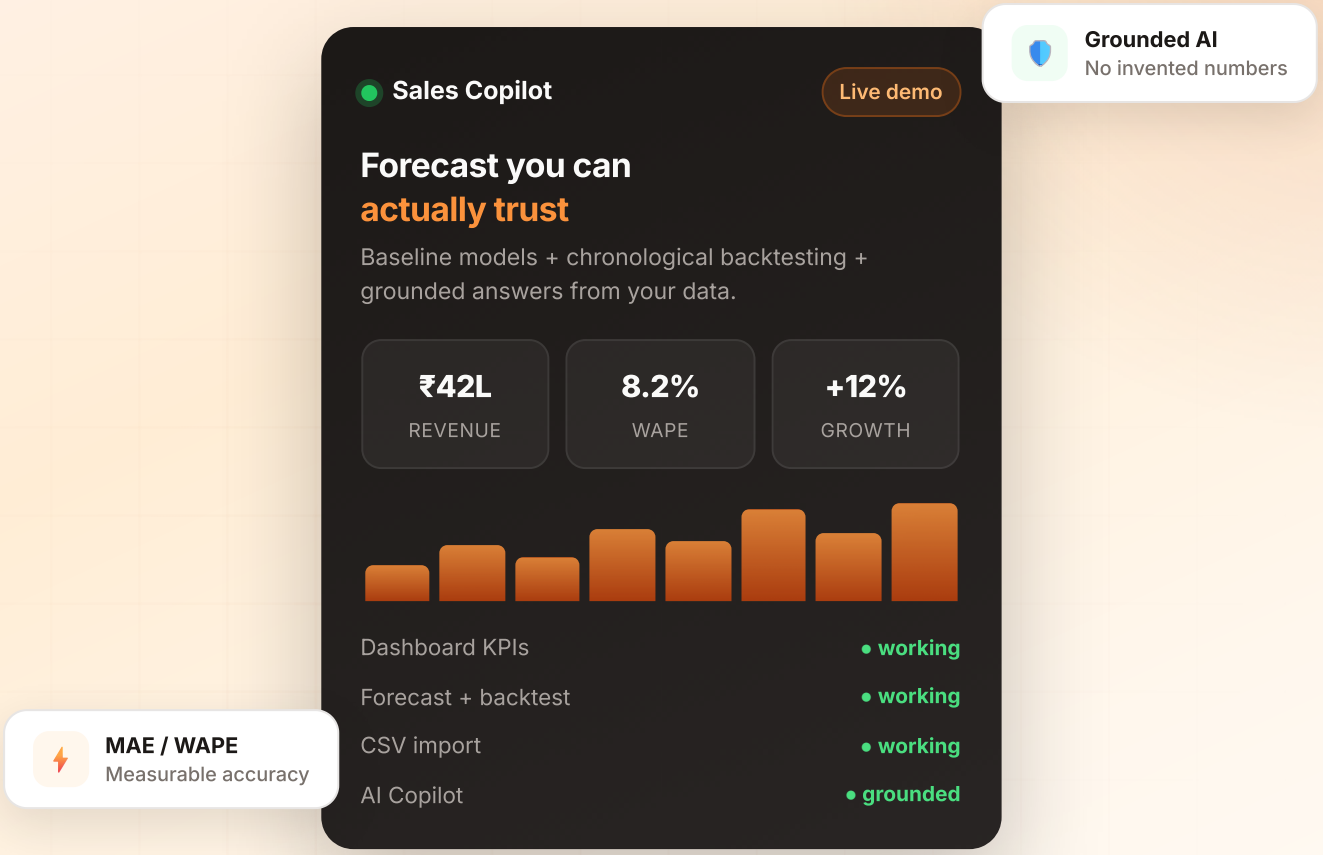


SOFTWARE DEVELOPMENT GUIDE

AI Sales Forecasting Copilot

Complete step-by-step guide to build a production-style Sales Forecasting + AI Copilot application. Ideal for final-year projects, interviews, and real engineering practice.

Nov 17, 2026 v1.0 React · Spring Boot · PostgreSQL

React + TypeScript | Java 17/21 + Spring Boot | PostgreSQL / H2 | Baseline Forecasting | Grounded AI Copilot

Boost Your Skills
Expert-led courses · Data Science · Web Dev · AI & ML
Explore Courses →

01 Executive Summary

AI Sales Forecasting Copilot helps sales, finance, and operations teams understand historical sales, estimate future demand, surface risks and opportunities, and ask natural-language questions — all grounded in real data.

It supports decisions. It does **not** guarantee revenue or replace managerial judgment.

> **One-line for a senior:**
> "We ingest sales history, validate and aggregate it, generate forecasts with accuracy metrics, visualize actuals vs predictions, flag meaningful trends, and let users ask grounded questions based on authorized business data."

What the working project includes

Feature	Status
Dashboard KPIs + charts	**Working**
Forecast Studio (MA + trend, MAE/WAPE)	**Working**
Chronological backtesting	**Working**
Sales Explorer (filter)	**Working**
CSV Upload + validation	**Working**
Scenario Planner (volume uplift)	**Working**
AI Copilot (grounded)	**Working**
Real LLM when API key set	**Working**
Rule-based fallback (no key)	**Working**
Sample data auto-seed + H2 demo	**Working**
Docker Compose for Postgres	**Ready**

02 Business Problem, Users & Value

Teams often rely on spreadsheets and intuition. Data is fragmented, reporting is repetitive, forecasts are inconsistent, and declining demand is discovered too late.

Users

- **Sales manager** — reviews trends and plans targets
- **Finance / FP&A** — compares forecast with budget, explains variance
- **Leadership** — KPIs, risks, opportunities
- **Analyst / Admin** — uploads data, maps columns, quality checks, access

Potential value (not guaranteed)

Less manual work
Cut repetitive spreadsheet and reporting effort.

Earlier visibility
Spot seasonality, demand shifts, declining products sooner.

Consistent planning
Same numbers and assumptions across teams and regions.

Traceable forecasts
Model version, training window, MAE/WAPE metrics.

Natural language
Non-technical users get grounded answers without SQL.

> **Important:** Measure pilots on time saved, forecast error, adoption, data quality, and decisions supported. Never promise guaranteed profit uplift.

03 Scope: MVP vs Later

MVP — build first	Release 2	Advanced
Login + roles	Multi-tenant	SSO / policies
CSV upload + validation	Scheduled imports / ERP	Streaming data
KPI dashboard + filters	Drill-down	Advanced segmentation
Baseline forecast + backtest	Model comparison	Probabilistic forecasts
Chart/table + metrics	Scenario overrides	Optimization
Rule insights + grounded Q&A	Alerts / saved reports	Agent workflows
Audit + CSV export	Email / Slack	Advanced governance

04 Architecture & Technology

- **Frontend:** React + TypeScript + Vite
- **Backend:** Java 17/21 + Spring Boot 3 (REST, validation, security)
- **Database:** H2 (demo) or PostgreSQL + Flyway
- **Forecast:** Transparent Java baseline (Moving Average + trend) with MAE & WAPE
- **AI:** LLM only from backend; rule-based fallback without API key
- **Deploy:** Docker Compose or jar + npm

Logical flow

```
Browser (React + TypeScript)
        |  HTTPS / REST + JSON
        ▼
Spring Boot API — Auth / Validation / Audit
  ├── Import & Data Quality ── Database
  ├── KPI & Analytics ──────── Database
  ├── Forecast module ──────── Forecast tables
  ├── Copilot module ──────── LLM (server-side only)
  └── Export / Scenarios
```

End-to-end workflow

1. User opens app; frontend talks to backend APIs.
2. Backend seeds sample sales data (demo profile).
3. Dashboard shows KPIs, trend, top products/regions, insights.
4. User runs forecast → MAE/WAPE calculated.
5. User uploads CSV → validated and saved.
6. User creates scenario (e.g. +10% volume) vs baseline.
7. User asks Copilot → grounded answer from approved analytics.

05 Prerequisites

Free tools. Each takes a few minutes to install.

1 Java 17 or 21

Download from adoptium.net. Then:

```
java -version
```

2 Node.js 18+

Download LTS from nodejs.org. Then:

```
node -v
npm -v
```

3 Docker (optional)

Only for real PostgreSQL. From docker.com.

4 OpenAI key (optional)

For real LLM answers. Without it, rule-based grounded mode still works.

06 How to Run

> **Demo login:** `admin@demo.com` / `admin123`

A — Backend

```
cd ai-sales-forecasting-copilot/backend

./mvnw spring-boot:run -Dspring-boot.run.profiles=demo

# Windows:
mvnw.cmd spring-boot:run -Dspring-boot.run.profiles=demo
```

Wait for "Started SalescopilotApplication". Backend: **http://localhost:8080**

B — Frontend

```
cd ai-sales-forecasting-copilot/frontend
npm install
npm run dev
```

Open **http://localhost:5173**

C — Optional Postgres

```
cd ai-sales-forecasting-copilot
docker compose up -d
```

D — Optional real LLM

```
export OPENAI_API_KEY=sk-your-key-here
# restart backend
```

Inside the app

1. **Dashboard** — KPIs, trend, products, regions, insights
2. **Forecasting** — set horizon → Run → see MAE / WAPE
3. **Sales Explorer** — filter by product / region
4. **Upload Center** — upload `sample-data/sales_sample.csv`
5. **Scenarios** — +5% / +10% volume vs baseline

6. **AI Copilot** — “growth rate?”, “top products”, “which region leads?”
7. **Reports** — latest forecast metrics

07 Folder Structure

```
ai-sales-forecasting-copilot/
├── frontend/                      ← React + TypeScript
│   ├── src/App.tsx                ← All screens (MVP)
│   ├── src/App.css
│   ├── package.json
│   └── vite.config.ts
├── backend/                       ← Spring Boot
│   ├── src/main/java/.../
│   │   ├── config/                Security, DataLoader
│   │   ├── controller/            REST endpoints
│   │   ├── service/               Analytics + Forecast
│   │   ├── copilot/               CopilotService + LlmClient
│   │   ├── entity/                SalesRecord, User
│   │   └── repository/
│   ├── src/main/resources/
│   │   ├── application-demo.yml   ← H2 profile
│   │   └── db/migration/
│   └── pom.xml
├── sample-data/sales_sample.csv
├── docker-compose.yml
└── README.md
```

08 Database Design

Flyway creates the schema on first start.

```
V1__init_schema.sql (core tables)
```

```
CREATE TABLE organizations (
  id BIGINT PRIMARY KEY, name VARCHAR(200) NOT NULL,
  timezone VARCHAR(50) DEFAULT 'Asia/Kolkata',
  default_currency CHAR(3) DEFAULT 'INR'
);

CREATE TABLE users (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  name VARCHAR(200), email VARCHAR(200) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  role_id BIGINT, organization_id BIGINT,
  status VARCHAR(30) DEFAULT 'ACTIVE',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE sales_records (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  organization_id BIGINT NOT NULL,
  sale_date DATE NOT NULL,
  product_name VARCHAR(200), region VARCHAR(100),
  quantity DECIMAL(12,2),
  revenue DECIMAL(14,2) NOT NULL,
  currency CHAR(3) DEFAULT 'INR',
  order_id VARCHAR(100),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE forecast_runs (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  organization_id BIGINT NOT NULL,
  target VARCHAR(50) DEFAULT 'revenue',
  grain VARCHAR(20) DEFAULT 'monthly',
  horizon INT DEFAULT 3,
  model_name VARCHAR(50) DEFAULT 'moving_average',
  status VARCHAR(30) DEFAULT 'COMPLETED',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE forecast_values (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  forecast_run_id BIGINT NOT NULL,
  period DATE NOT NULL,
  predicted_value DECIMAL(14,2) NOT NULL,
  lower_bound DECIMAL(14,2), upper_bound DECIMAL(14,2)
);

CREATE TABLE forecast_metrics (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  forecast_run_id BIGINT NOT NULL,
  metric_name VARCHAR(50) NOT NULL,
  metric_value DECIMAL(12,4)
);
```

- Money as `DECIMAL`, never float
- Index `organization_id + sale_date`
- Every query filters by organization

09 REST API Contract

Method	Endpoint	Purpose
GET	/api/v1/health	Health check
POST	/api/v1/auth/login	Login
GET	/api/v1/dashboard/summary	KPI cards
GET	/api/v1/analytics/revenue-trend	Revenue series
GET	/api/v1/analytics/top-products	Top products
GET	/api/v1/analytics/top-regions	Top regions
GET	/api/v1/insights	Rule insights
POST	/api/v1/forecasts	Create forecast
GET	/api/v1/forecasts/latest	Latest forecast
GET	/api/v1/sales	Search / filter
POST	/api/v1/imports	Upload CSV
POST	/api/v1/scenarios	Create scenario
POST	/api/v1/copilot/messages	Grounded Q&A

10 Forecasting Math

Time-series problem. Target = revenue, grain = monthly.

MVP models

- **Moving Average** — average of last k periods (k=3)
- **Simple trend** — linear slope added to MA

Workflow

1. Aggregate revenue by month
2. Chronological train/test split (never random)
3. Fit MA + trend on training
4. Predict test period, compare to actuals
5. Compute **MAE** and **WAPE**
6. Refit on all history → forecast future horizon

- **MAE** = avg |actual − forecast|
- **WAPE** = sum(|actual − forecast|) / sum(|actual|) × 100

> Avoid MAPE when actuals can be zero. Prefer MAE and WAPE. Accuracy is not a promise.

11 Copilot — Grounded Answers

LLM only explains data. Never invents numbers or runs arbitrary SQL.

Flow

1. User asks a question
2. Backend applies org filter
3. Allow-listed functions only (summary, forecast, top products…)
4. Compact results sent to LLM or rule engine
5. Answer returned with figures, period, and caveat

Example questions

- What was revenue growth month over month?
- Which five products generated most revenue?
- Compare actual vs forecast for last three months
- Which region has the largest forecast decline?

> Guardrails: validate args, access control before retrieval, never expose secrets, show “insufficient data” instead of fabricating.

12 Frontend

MVP UI lives in `frontend/src/App.tsx`. Orange + white professional theme.

Screen	Purpose
Dashboard	KPIs, trend, products, regions, insights
Forecasting	Horizon, Run, MAE/WAPE, chart
Sales Explorer	Filterable table
Upload Center	CSV drop + validation
Scenarios	Volume uplift vs baseline
AI Copilot	Chat with grounded answers
Reports	Latest forecast metrics

```
cd frontend
npm install
npm run dev
```

13 Backend Code Highlights

SalescopilotApplication.java

```
@SpringBootApplication
public class SalescopilotApplication {
    public static void main(String[] args) {
        SpringApplication.run(SalescopilotApplication.class, args);
    }
}
```

SecurityConfig (MVP – permitAll for demo)

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf → csrf.disable())
        .cors(Customizer.withDefaults())
        .sessionManagement(s → s.sessionCreationPolicy(SessionCreationPolicy.STAT
        .authorizeHttpRequests(auth → auth
            .requestMatchers("/api/v1/auth/**", "/api/v1/health").permitAll()
            .requestMatchers("/api/v1/**").permitAll()
            .anyRequest().permitAll()
        );
    return http.build();
}
```

application-demo.yml (H2)

```
spring:
  datasource:
    url: jdbc:h2:mem:salescopilot;MODE=PostgreSQL;DB_CLOSE_DELAY=-1
    driver-class-name: org.h2.Driver
  jpa:
    hibernate:
      ddl-auto: none
  flyway:
    enabled: true
server:
  port: 8080
app:
  llm:
    api-key: ${OPENAI_API_KEY:}
    model: gpt-4o-mini
```

CSV format

```
sale_date,product,region,quantity,revenue,currency,order_id
2024-01-01,Product A,North India,120,240000,INR,ORD-1001
2024-02-01,Product A,North India,135,270000,INR,ORD-1002
```

14 Presentation Script (5–7 min)

Opening

“I’m building an AI Sales Forecasting Copilot to make sales planning data-driven and easier to explain.”

Problem

“Teams spend time on spreadsheets. Forecasts are inconsistent; risks are detected late.”

Solution

“Import + validate data, calculate KPIs, forecast with metrics, visualize actual vs predicted, answer natural-language questions using approved analytics.”

Architecture

“React/TypeScript UI, Spring Boot APIs, PostgreSQL/H2, transparent baseline model. LLM calls are server-side and grounded.”

Demo flow

1. Dashboard KPIs and trend
2. Run forecast → show MAE / WAPE
3. Upload sample CSV
4. +10% scenario vs baseline
5. Ask Copilot: “Which region may decline?” — show supporting numbers

Engineering decisions

- Baseline first for measurability
- Chronological backtesting (no leakage)
- Organization filter for security
- Saved forecast runs for reproducibility

Limitations

“Forecast quality depends on data. Copilot explains metrics — it does not guarantee outcomes or make decisions autonomously.”

15 Implementation Plan

Phase	Work	Days
0 Planning	Metric, grain, schema, roles, ERD	1–2
1 Setup	Repo, React, Spring Boot, Docker	1
2 Backend base	Profiles, CORS, Flyway, health	2–3
3 Auth	Users, hashing, Security	2–3
4 Import	CSV, validation, errors	3–5
5 Sales APIs	Search, filter, pagination	2–3
6 Dashboard	KPIs, charts	3–4
7 Forecast	Baseline, backtest, MAE/WAPE	4–6
8 Copilot	Tools, LLM, chat UI	3–5
9 Scenario	Uplift, export	2–4
10 Polish	Tests, responsive, docs	4–6
11 Deploy	Docker, demo	1–3

Solo timeline: 4–8 weeks. Do not start with complex LLM models.

16 Final Checklist

- [ ] README with setup, env, architecture, demo
- [ ] ER diagram + API docs
- [ ] Synthetic CSV + data dictionary
- [ ] Working login + role checks
- [ ] CSV import with validation
- [ ] Dashboard KPIs and charts
- [ ] Forecast with backtesting + MAE/WAPE
- [ ] Copilot grounded in authorized analytics
- [ ] Unit / integration tests
- [ ] Responsive UI
- [ ] Docker notes
- [ ] Presentation script + known limitations

17 Start Today — First Five Tasks

1. Create repo + README (or use the provided zip)
2. Prepare synthetic CSV + data dictionary
3. Draw dashboard wireframe + ERD
4. Start H2/Postgres + Spring Boot; health endpoint + first migration
5. **Do not add LLM until import, KPIs, and baseline forecast work end-to-end**

18 Glossary

Term	Meaning
Forecast	Estimate of future value from history / assumptions
Time series	Chronological observations at fixed intervals
Backtesting	Test on periods not used to fit the model
MAE	Mean Absolute Error
WAPE	Weighted Absolute Percentage Error
Data leakage	Using info unavailable at forecast time
RBAC	Role-Based Access Control
LLM	Large Language Model
RAG	Retrieval-Augmented Generation

19 Why This Project Matters

Transparent models

MA + trend are easy to explain. No black-box first.

Measurable accuracy

MAE and WAPE on held-out data. You can prove quality.

Grounded AI

Copilot never invents numbers — only approved analytics.

Works offline

H2 + rule mode runs without cloud. OpenAI is optional.

Production path

Same code works with Postgres, Docker, real auth.

Interview-ready

Clear architecture, metrics, limitations, and a 5–7 min script.



Ready to level up?

Explore Courses →

Data Science · Web Development · AI & ML on Uncodemy

AI Sales Forecasting Copilot — Complete Project Guide

React + TypeScript · Java 17/21 + Spring Boot · PostgreSQL / H2 · Baseline Forecasting · Grounded AI