Platform	Jupyter	Enhancement →	FastAPI — API Serving IDE (VS Code) docker — Containerization
Data Presentation (UI)		Enhancement →	Chart.js — Interactive Charts React — Web UI TAURI — Desktop
Database		Enhancement →	PostgreSQL — Relational Database SQLAlchemy — Python & Postgres Interface
Algorithms & Modeling		Enhancement →	scikit learn — Linear regression, preprocessing, metrics PROPHET — Time Series Forecasting PyTorch — Reinforcement Learning
Data Source & Transformation		Enhancement →	polars — Data Transformation United States Census Bureau — US Census API Pandas — Data Transformation
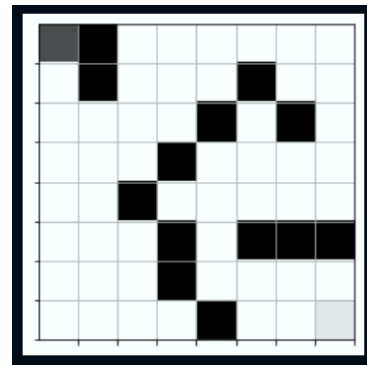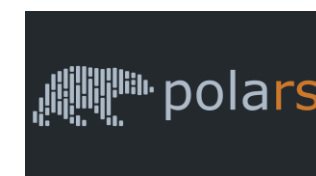
System Architecture

Data & ETL

A1. External Data Sources

US Census (ACS/BPS)

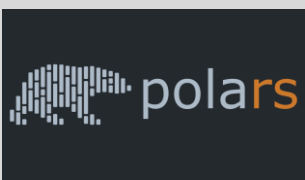Bureau of Labor & Statistics (JOLTS/Unemployment)

Federal Reserve & Economic Data (rates)

A2. ETL & Orchestration

Requests + Rate Limiting, Retries, Secrets via .env
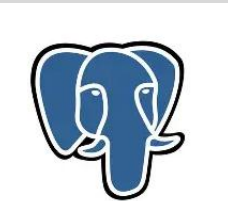
Batch ID, idempotent upserts, logging

A3. Data Processing

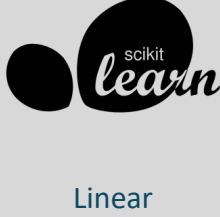Clean, Validate (nulls, ranges), normalize, feature engineering

Storage & Models

B1. PostgreSQL (Schemas: raw, stg, core, ml)

core.population_observations, core.indicator_values

Ml.feature_matrix (materialized view)

Indexes incl. uq_fm_geo_year

B2. Model Training (Dockerized)

Linear regression, preprocessing, metrics

Time Series Forecasting

Reinforcement Learning

Evaluation & Selection (MAE/MAPE)

B3. Model Artifacts / Registry

📁 **Repository Structure**

/PopulationPredictionPlatform
|— /data # Raw & processed Census data
|— /models # Trained model artifacts
|— /notebooks # Prototyping and experimentation
|— /api # FastAPI service

models/ + metadata.json (versioned)

API & UI

C1. FastAPI Service (SQL Alchemy)

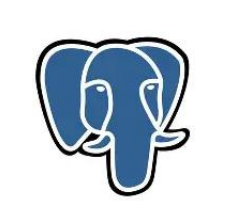/predicts, /metrics, /health; Auth (JWT/basic)

C2. React (Web) / Tauri (Desktop)

Scenario inputs, Chart.js forecasts, export PNG/CSV

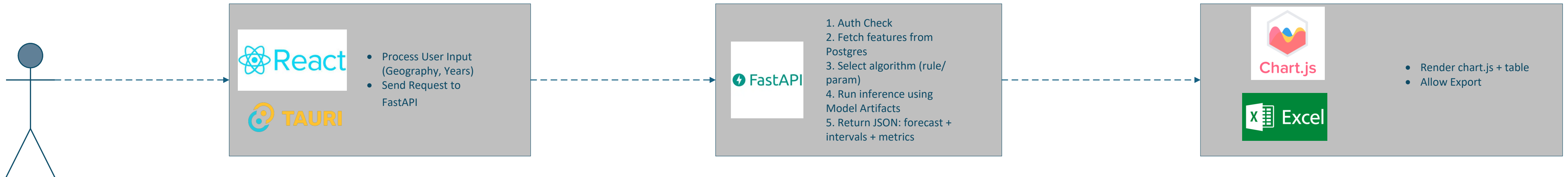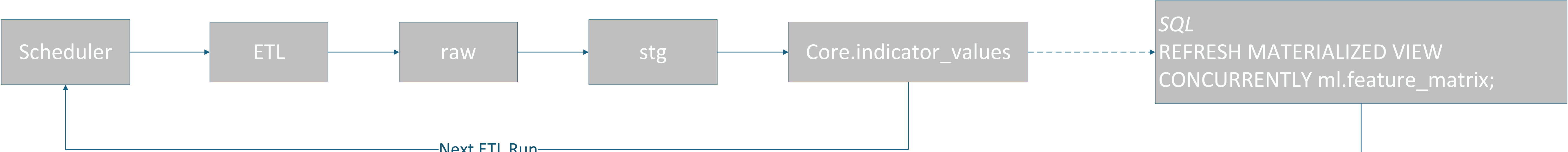## Security & Observability

PostgreSQL

Row Based Access Control

Docker

Secret management (.env)

Container Hardening

Structured Logs

Health Checks & Metrics

Request -> Predict -> Visualize

React / TAURI
- Process User Input (Geography, Years)
- Send Request to FastAPI

FastAPI
1. Auth Check
2. Fetch features from Postgres
3. Select algorithm (rule/param)
4. Run inference using Model Artifacts
5. Return JSON: forecast + intervals + metrics

Chart.js / Excel
- Render chart.js + table
- Allow Export

ETL Training Loop

ETL

Scheduler → ETL → raw → stg → Core.indicator_values ⟶ *SQL* REFRESH MATERIALIZED VIEW CONCURRENTLY ml.feature_matrix;

Next ETL Run

Training

ml.feature_matrix → Evaluates → Writes Model