**✅ Step 1: Define Core Data Models**

We'll start by creating schemas for:

**🏭 Asset**

Represents a physical entity being monitored (e.g., HVAC unit, generator, pump).

sql

CopyEdit

CREATE TABLE assets (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name TEXT NOT NULL,

type TEXT NOT NULL,

location TEXT,

status TEXT DEFAULT 'active',

created\_at TIMESTAMPTZ DEFAULT NOW(),

updated\_at TIMESTAMPTZ DEFAULT NOW()

);

**📡 Telemetry**

Time-series data tied to an asset. Stored using TimescaleDB’s hypertables.

sql

CopyEdit

CREATE TABLE telemetry (

id BIGSERIAL PRIMARY KEY,

asset\_id UUID REFERENCES assets(id),

timestamp TIMESTAMPTZ NOT NULL DEFAULT NOW(),

temperature DOUBLE PRECISION,

vibration DOUBLE PRECISION,

humidity DOUBLE PRECISION,

pressure DOUBLE PRECISION,

voltage DOUBLE PRECISION

);

Convert telemetry into a hypertable (TimescaleDB):

sql

CopyEdit

SELECT create\_hypertable('telemetry', 'timestamp');

**🚨 Alerts**

Generated from ML model inference or rule-based logic.

sql

CopyEdit

CREATE TABLE alerts (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

asset\_id UUID REFERENCES assets(id),

severity TEXT CHECK (severity IN ('info', 'warning', 'critical')),

message TEXT NOT NULL,

triggered\_at TIMESTAMPTZ DEFAULT NOW(),

acknowledged BOOLEAN DEFAULT FALSE

);

**🧪 Development Step**

Would you like:

* A SQL seed script for setting up the DB locally?
* A SQLAlchemy ORM model for FastAPI?
* Or should we move directly to the **FastAPI backend setup (app.py + db/init)**?

You call the shot.

2/2

**You said:**

FastAPI

**ChatGPT said:**

💥 Let’s get that **FastAPI backend** up and humming. We’ll lay down the core structure to support:

* API initialization (app.py)
* Database connection
* ORM models using SQLAlchemy
* Modular routing for assets, telemetry, and alerts

**✅ Step 2: FastAPI Backend Setup**

**📁 File: backend/app.py**

python

CopyEdit

from fastapi import FastAPI

from backend.db.database import engine, Base

from backend.routers import assets, telemetry, alerts

app = FastAPI(title="Digital Twin API", version="1.0")

# Create DB tables

Base.metadata.create\_all(bind=engine)

# Include routers

app.include\_router(assets.router, prefix="/assets", tags=["Assets"])

app.include\_router(telemetry.router, prefix="/telemetry", tags=["Telemetry"])

app.include\_router(alerts.router, prefix="/alerts", tags=["Alerts"])

**📁 File: backend/db/database.py**

python

CopyEdit

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

DATABASE\_URL = "postgresql://postgres:password@localhost:5432/digital\_twin"

engine = create\_engine(DATABASE\_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative\_base()

**📁 File: backend/models/models.py**

python

CopyEdit

from sqlalchemy import Column, String, Float, TIMESTAMP, Boolean, ForeignKey

from sqlalchemy.dialects.postgresql import UUID

from sqlalchemy.sql import func

import uuid

from backend.db.database import Base

class Asset(Base):

\_\_tablename\_\_ = "assets"

id = Column(UUID(as\_uuid=True), primary\_key=True, default=uuid.uuid4)

name = Column(String, nullable=False)

type = Column(String, nullable=False)

location = Column(String)

status = Column(String, default="active")

created\_at = Column(TIMESTAMP(timezone=True), server\_default=func.now())

updated\_at = Column(TIMESTAMP(timezone=True), onupdate=func.now())

class Telemetry(Base):

\_\_tablename\_\_ = "telemetry"

id = Column(String, primary\_key=True)

asset\_id = Column(UUID(as\_uuid=True), ForeignKey("assets.id"))

timestamp = Column(TIMESTAMP(timezone=True), server\_default=func.now())

temperature = Column(Float)

vibration = Column(Float)

humidity = Column(Float)

pressure = Column(Float)

voltage = Column(Float)

class Alert(Base):

\_\_tablename\_\_ = "alerts"

id = Column(UUID(as\_uuid=True), primary\_key=True, default=uuid.uuid4)

asset\_id = Column(UUID(as\_uuid=True), ForeignKey("assets.id"))

severity = Column(String)

message = Column(String)

triggered\_at = Column(TIMESTAMP(timezone=True), server\_default=func.now())

acknowledged = Column(Boolean, default=False)

**📁 Folder scaffolding to match:**

pgsql

CopyEdit

backend/

├── app.py

├── db/

│ └── database.py

├── models/

│ └── models.py

├── routers/

│ ├── assets.py

│ ├── telemetry.py

│ └── alerts.py