--ini--.py file  
  
*from flask import Flask*

*from flask\_sqlalchemy import SQLAlchemy*

*from dotenv import load\_dotenv*

*import os*

*# Load environment variables*

*load\_dotenv()*

*db = SQLAlchemy()*

*def create\_app():*

*app = Flask(\_\_name\_\_)*

*# Set the database URI*

*app.config['SQLALCHEMY\_DATABASE\_URI'] = os.getenv("DATABASE\_URI")*

*app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False*

*db.init\_app(app)*

*# Register blueprints*

*from .routes import main*

*app.register\_blueprint(main)*

*return app*data\_acquisation.py file  
  
*import requests*

*from sqlalchemy import create\_engine*

*from sqlalchemy.orm import sessionmaker*

*from sqlalchemy.sql import text*

*from dotenv import load\_dotenv*

*import os*

*# Load environment variables*

*load\_dotenv()*

*API\_KEY = os.getenv("ALPHA\_VANTAGE\_API\_KEY")*

*DATABASE\_URI = os.getenv("DATABASE\_URI")*

*engine = create\_engine(DATABASE\_URI)*

*Session = sessionmaker(bind=engine)*

*def fetch\_data(symbol):*

*"""Fetch stock data from Alpha Vantage."""*

*print(f"Fetching data for symbol: {symbol}...")*

*url = f"https://www.alphavantage.co/query"*

*params = {*

*"function": "TIME\_SERIES\_DAILY",*

*"symbol": symbol,*

*"apikey": API\_KEY*

*}*

*response = requests.get(url, params=params)*

*print(f"HTTP Status Code: {response.status\_code}")*

*if response.status\_code == 200:*

*print("Raw API Response:")*

*print(response.json()) # Print the entire JSON response*

*return response.json()*

*else:*

*print(f"Failed to fetch data: {response.status\_code}")*

*return {}*

*def save\_to\_database(data, engine):*

*"""Save data to the database."""*

*print("Saving data to the database...")*

*session = Session()*

*time\_series = data.get("Time Series (Daily)", {})*

*if not time\_series:*

*print("No time series data found in the API response!")*

*return*

*for date, values in time\_series.items():*

*print(f"Processing date: {date}...")*

*try:*

*session.execute(*

*text("""*

*INSERT INTO daily\_stock\_prices (date, open, high, low, close, volume)*

*VALUES (:date, :open, :high, :low, :close, :volume)*

*ON DUPLICATE KEY UPDATE*

*open=:open, high=:high, low=:low, close=:close, volume=:volume*

*"""),*

*{*

*"date": date,*

*"open": float(values["1. open"]),*

*"high": float(values["2. high"]),*

*"low": float(values["3. low"]),*

*"close": float(values["4. close"]),*

*"volume": int(values["5. volume"])*

*}*

*)*

*session.commit()*

*print(f"Inserted data for date: {date}")*

*except Exception as e:*

*print(f"Failed to insert data for date {date}: {e}")*

*session.rollback()*

*session.close()*

*if \_\_name\_\_ == "\_\_main\_\_":*

*symbol = "IBM" # Example stock symbol*

*print("Starting data acquisition script...")*

*data = fetch\_data(symbol)*

*if data:*

*save\_to\_database(data, engine)*

*print("Data acquisition script completed!")*

database.py file  
  
*from flask\_sqlalchemy import SQLAlchemy*

*db = SQLAlchemy()*

model.py file  
  
*from . import db*

*class DailyStockPrice(db.Model):*

*id = db.Column(db.Integer, primary\_key=True)*

*date = db.Column(db.Date, nullable=False)*

*open = db.Column(db.Float, nullable=False)*

*high = db.Column(db.Float, nullable=False)*

*low = db.Column(db.Float, nullable=False)*

*close = db.Column(db.Float, nullable=False)*

*volume = db.Column(db.BigInteger, nullable=False)*

routes.py file  
  
*from flask import Blueprint, render\_template*

*from app import db*

*from .models import DailyStockPrice # Import your model*

*main = Blueprint('main', \_\_name\_\_)*

*@main.route("/")*

*def dashboard():*

*try:*

*# Fetch data from the database*

*raw\_data = DailyStockPrice.query.order\_by(DailyStockPrice.date).all()*

*# Handle no data case*

*if not raw\_data:*

*print("No data found in the database!")*

*return render\_template("dashboard.html", data=[])*

*# Process data for the template*

*data = [*

*{*

*'date': row.date.isoformat(),*

*'open': row.open,*

*'high': row.high,*

*'low': row.low,*

*'close': row.close*

*}*

*for row in raw\_data*

*]*

*print(f"Formatted data: {data}")*

*return render\_template("dashboard.html", data=data)*

*except Exception as e:*

*print(f"Error in dashboard route: {e}")*

*return "An error occurred while processing the dashboard.", 500*

dashboard.html file  
  
*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>Dashboard</title>*

*<script src="https://www.gstatic.com/charts/loader.js"></script>*

*<script>*

*// Load the Google Charts library*

*google.charts.load('current', {'packages':['corechart']});*

*// Set a callback to run when the Google Visualization API is loaded*

*google.charts.setOnLoadCallback(drawChart);*

*function drawChart() {*

*// Data passed from Flask*

*const chartData = {{ data|tojson }};*

*// Handle empty data gracefully*

*if (!chartData.length) {*

*document.getElementById('curve\_chart').innerHTML = '<p>No data available.</p>';*

*return;*

*}*

*// Prepare the data for Google Charts*

*const dataArray = [*

*['Date', 'Open', 'High', 'Low', 'Close'], // Table headers*

*...chartData.map(row => [*

*new Date(row.date), // Convert the date string to a JavaScript Date object*

*row.open,*

*row.high,*

*row.low,*

*row.close*

*])*

*];*

*// Create a Google Visualization DataTable*

*const data = google.visualization.arrayToDataTable(dataArray);*

*// Set chart options*

*const options = {*

*title: 'Stock Prices',*

*curveType: 'function',*

*legend: { position: 'bottom' },*

*width: 900,*

*height: 500*

*};*

*// Draw the chart in the specified element*

*const chart = new google.visualization.LineChart(document.getElementById('curve\_chart'));*

*chart.draw(data, options);*

*}*

*</script>*

*</head>*

*<body>*

*<h1>Stock Prices Dashboard</h1>*

*<div id="curve\_chart" style="width: 900px; height: 500px;"></div>*

*</body>*

*</html>*