Detailed Explanation of Project Architecture & Workflow

This project builds a **real-time dynamic pricing system for parking lots** using Pathway's streaming engine. The system ingests parking data and applies pricing models to compute optimal prices based on occupancy, traffic, demand, and vehicle type. It also visualizes these prices using Bokeh.

---

1. Data Ingestion Layer

- The project starts by loading a static historical dataset (`dataset.csv`) simulating real-time parking data.
- Key features include:
    - `Occupancy`, `Capacity` → parking usage
    - `QueueLength`, `TrafficConditionNearby` → congestion indicators
    - `IsSpecialDay`, `VehicleType` → contextual demand influencers
    - `LastUpdatedDate`, `LastUpdatedTime` → combined into `Timestamp` for streaming

This data is cleaned and written to `Parking_stream.csv` for further use.

---

2. Stream Simulation using Pathway

- Pathway's `replay_csv()` method is used to simulate **real-time streaming** of parking data.
- A schema (`ParkingSchema`) defines expected column types for streaming.
- The replay simulates data flow row by row using a fast `input_rate`, creating the illusion of a live system.

---

3. Model 1: Linear Occupancy-Based Pricing

- Implements a simple pricing formula:

$$\text{Price} = \text{BasePrice} + \alpha \cdot \left(\frac{\text{Occupancy}}{\text{Capacity}}\right)$$

- Price is clamped between 0.5× and 2× the base price.
- This model is **simple, interpretable, and responsive to crowding**.
- The output is written to `model1_output.jsonl`.

---

4. Model 2: Demand-Based Multi-Feature Pricing

- A more advanced pricing model that considers:

Feature	Description
`Occupancy / Capacity`	Real-time usage
`QueueLength`	Indicates urgency
`TrafficConditionNearby`	Coded as `low` = 0, `high` = 2
`IsSpecialDay`	Boosts demand
`VehicleType`	Weighted (e.g., car = 1, truck = 2)

- Demand is calculated with weighted coefficients and normalized.
- Final price is:

Price=BasePrice⋅(1+λ⋅NormalizedDemand)\text{Price} = \text{BasePrice} \cdot (1 + \lambda \cdot \text{NormalizedDemand})

- This allows **finer control and smarter pricing under varying conditions**.
- Output is saved as `model2_output.jsonl`.

---

5. Visualization Layer: Bokeh + Panel

- In **Model 1 & 2**, the final pricing stream is:
  - Either **written to a `.jsonl` file and plotted statically** using Bokeh + Pandas
  - Or **visualized live using `delta_window.plot(...)`** in Panel (starter notebook style)
- The chart plots **`Timestamp (t)` vs `Price`**, showing price fluctuations over time.

💡 On Colab, the team used static rendering via Bokeh, since Panel's dynamic server is unsupported.

---

6. Workflow Summary

1. Load & clean raw data
2. Simulate real-time data with `Pathway`
3. Apply **Model 1 (Linear Pricing)** logic
4. Apply **Model 2 (Demand-Based Pricing)** logic
5. Output results
6. Visualize using Bokeh

Architectural Summary

- **Ingestion** → cleans + streams data
- **Processing Layer (Pathway)** → handles model logic as `pw.Table` transformations
- **Output Layer** → writes to file or visualizes via Bokeh
- **Interaction (optional)** → Panel or dashboard for real-time plots

---

Why This Architecture Works

- **Real-time simulation** with control over input rate
- **Multiple pricing models** supported in modular stages
- **Visualization-ready output** for decision making
- **Extensible** to add more models like competition-aware pricing (Model 3)