Morocco Property Value Estimator – Complete Workflow Overview

1. User Interface (Frontend) — price-estimator.blade.php

Users interact with a clean form on a Laravel Blade view, where they input key property details:

- **Property Type**: Apartment or Villa
- **Size**: Area in square meters
- **Bedrooms & Bathrooms**: Count of each
- **Property Age**: In years
- **Floor Level**: (0 for ground)
- **Special Features**: Checkboxes – Parking, Garden, Pool
- **Location**: City & Neighborhood dropdowns

Trigger: Button "See What Your Home Is Worth" posts the form.

2. Form Submission — POST /estimate

- Route: price.estimate in routes/web.php
- Mapped to estimate() method in **PriceEstimatorController.php**

3. Laravel Controller (Backend Logic):

In PriceEstimatorController.php:

1. **Validation** of all input fields (required, numeric, valid values)
2. **Data Formatting** into a structured associative array
3. **Conversion** of array to JSON format
4. **Temporary File Creation** to store JSON
5. **Shell Command** assembled to run predict.py with that file
6. **Execute Python** via shell_exec() to begin ML processing

4. Python Prediction Engine — predict.py:

Executes the actual machine learning steps:

1. **Load JSON Input** from temp file
2. **Load ML Models** trained for:
    - Apartment model
    - Villa model
3. **Convert Input** to Pandas DataFrame
4. **Process Features**:
    - Convert booleans: parking, pool, garden
    - Handle city/neighborhood using preprocessed encoders or mappings
5. **Select Model** based on property type

5. BKAM IPAI Integration — ipai_data.py:

Adds realism with economic adjustment:

- Contains **IPAI index data** from **Q4 2024** (e.g., Casablanca: +15.2%, Tangier: +17.7%)
- Fetches adjustment factor for the city
- Adjusts base predicted price accordingly

6. Price Prediction Logic:

1. Base prediction made using selected model
2. IPAI adjustment applied (e.g., `price *= 1.152` for Casablanca)
3. **Prediction range** calculated (e.g., ±5%)
4. Response formatted in **JSON** structure:

```
{
  "base_price": 123456,
  "adjusted_price": 142234,
  "range": [135000, 150000],
  "details": { ... }
}
```

7. Laravel Result Handling & Display:

- Laravel controller **reads Python output**
- Parses and checks for errors
- If successful:
  - Redirects back to form with results
  - Blade template displays:
    - Predicted price
    - Price range
    - Summary of inputs
- If failed:
  - Shows an error message to the user

8. Data Flow Diagram: