**“Local Activity Finder”**

**Architecture Document**

**Version 1.0**

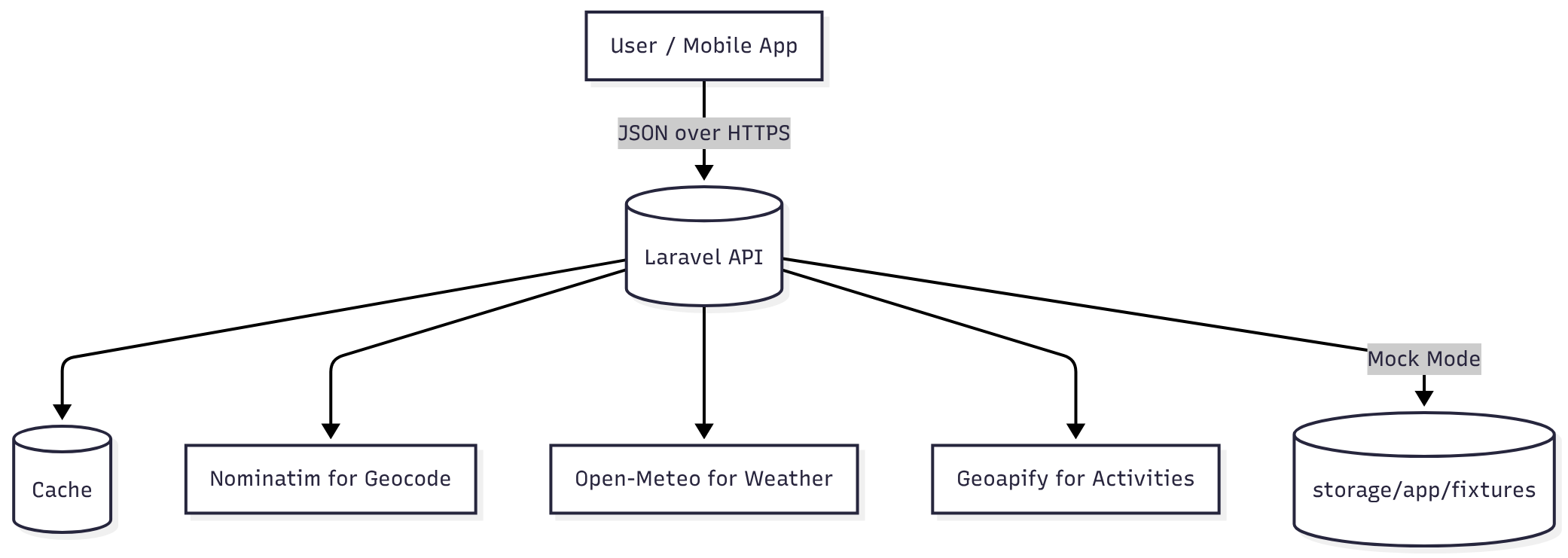
**10/18/2025**

### **Version 1.0** - *10/18/2025* – M. Rado – Initial document creation and base architecture added.

## Overview

A Laravel backend aggregates **Geocoding (Nominatim)**, **Weather (Open-Meteo)**, and **Activities (Geoapify)** into a small REST API. A React client (in ui/) consumes this API. The backend normalizes upstream responses, applies a weather-aware **indoor/outdoor recommendation** rule, and supports a fixture-backed **Mock Mode** for reliable local/CI development.

## High-level context



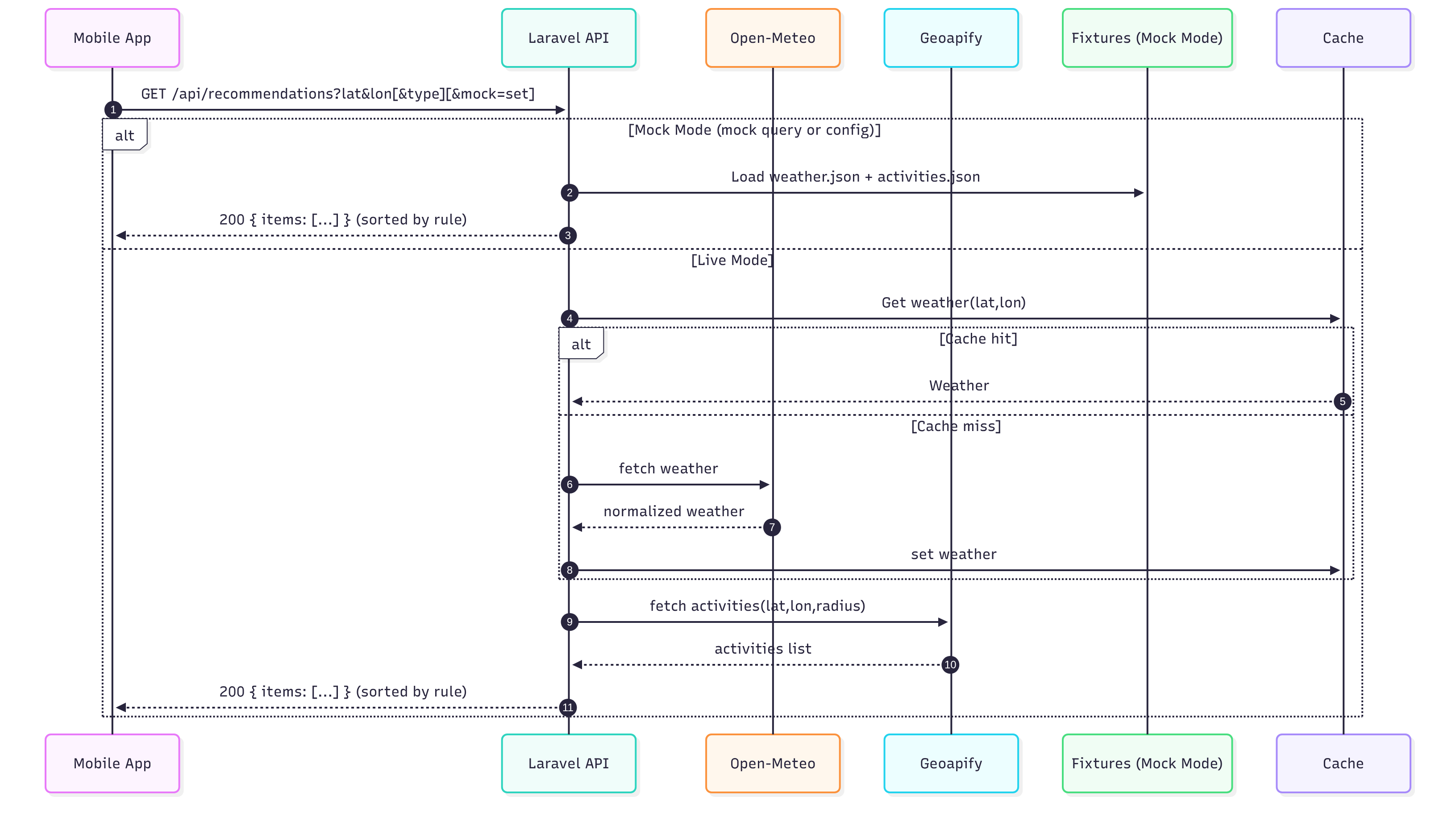
**Key points**

* Mock Mode bypasses upstream calls and serves JSON fixtures from disk.
* Cache smooths rate limits and improves latency for geocode/weather/POIs.
* All responses are normalized into predictable shapes for the UI/app.

## Components

* **Backend (Laravel)**
  + Routing: /api/geocode/search, /api/geocode/reverse, /api/weather, /api/activities, /api/recommendations
  + Validation: Form Requests (query + lat/lon/radius/type)
  + HTTP client: upstream calls (+ retry/backoff)
  + Caching: per-endpoint keys with short TTLs
  + Mock Mode: serves storage/app/fixtures/<set>/{weather,activities}.json
  + Tests: Pest; fixtures seeded for deterministic runs
* **Frontend (React,** ui/**)**
  + Vite dev/build
  + ESLint + Prettier (check-only in CI)
  + React Query for client-side caching
* **CI**
  + Two jobs: backend (PHP) and frontend (Node)
  + Steps: **install → lint → test**
  + Seeded fixtures for backend tests

## Request flow (Recommendations)



**Rule (indoor vs outdoor)**

* Activities are scored based on weather, increasing the weight of indoor items as the weather gets less favorable. Distance is also used to help show activities that are convenient to the user.
  + Precipitation probability above 40% adds .5
  + Precipitation above .2mm adds .5
  + Wind above 35kph adds .3
  + Else ⇒ outdoor first

## Normalization (payload shapes)

* **Weather**

{

"tempC": 21.4, "windKph": 8.7,

"precipMm": 0.0, "precipProb": 10,

"hourly": [{ "time": "2025-01-01T10:00Z", "tempC": 22.0 }]

}

* **Activities / Recommendations**

{

"items": [

{

"id": "poi-123",

"name": "City Park",

"lat": 40.7, "lon": -74.0,

"distanceM": 530,

"category": "park",

"indoor": false

}

]

}

* **Geocode search**

{ "items": [{ "name": "Times Square, NYC", "lat": 40.758, "lon": -73.9855 }] }

## Configuration

* **Env flags**
  + MOCK\_MODE=true|false (global default; can be overridden per-request with ?mock=<set>)
  + MOCK\_FIXTURE=default (or raining)
  + FILESYSTEM\_DISK=local
  + Cache/queue/session set to in-memory drivers for tests/CI
* **Per-request override**
  + Append ?mock=<set> to any API call to force fixtures for that call.
  + When mock mode is active, responses include X-Source: mock

## Fixtures

* Path used by the app/tests (relative to storage/app):  
  fixtures/<set>/{weather,activities}.json
* Example sets:
  + default: clear weather + mixed activities
  + raining: precipProb high + at least one indoor activity

## Error handling, retry & caching

* **HTTP retries** with exponential backoff for upstreams (bounded attempts).
* **Caching**
  + Weather: key by (lat,lon) rounded; short TTL (1 min).
  + Geocode: key by query string + limit; short TTL (10 min cfg in env).
  + Activities: key by (lat,lon,radius,type); short TTL (2 min).
* **Validation**
  + 422 on invalid inputs (lat/lon ranges, type enum, radius bounds).

### Error, timeout & rate-limit behaviors (add under “Error handling, retry & caching”)

#### Timeouts & retry policy

| Upstream | Connect timeout | Read timeout | Max attempts | Backoff sequence (± jitter) | Notes |
| --- | --- | --- | --- | --- | --- |
| Open-Meteo | 2s | 5s | 3 | 250ms → 500ms → 1s | No key required. |
| Geoapify | 2s | 5s | 3 | 250ms → 500ms → 1s | Keyed; monitor 4xx vs 5xx. |
| Nominatim | 2s | 5s | 3 | 250ms → 500ms → 1s | Identify with User-Agent; be respectful. Add local DB for production |

Implementation: exponential backoff with small random jitter; abort after max attempts and return normalized error to client.

#### Rate-limit handling (429)

* If Retry-After header present → honor it; otherwise use next backoff delay.
* Server returns **429** to clients only after exhausting retries; body includes a short, user-friendly message and a machine code (e.g., "code": "RATE\_LIMITED").
* Mobile guidance: surface a toast (“Temporarily rate-limited — try again”), and allow pull-to-refresh.

#### Error taxonomy (client-visible)

| HTTP | code | When | Client guidance |
| --- | --- | --- | --- |
| 422 | VALIDATION\_ERROR | Bad or missing params | Highlight fields; show inline error. |
| 429 | RATE\_LIMITED | Provider throttled after retries | Toast + retry affordance. |
| 502/503/504 | UPSTREAM\_UNAVAILABLE | Provider down/timeout after retries | Toast + retry; optionally fall back to cached/stale. |
| 500 | INTERNAL\_ERROR | Unexpected server error | Toast; report via telemetry. |

### Caching strategy & TTLs (replace/augment your caching bullets with this table)

| Endpoint | Cache key schema | Rounding | TTL | Invalidation |
| --- | --- | --- | --- | --- |
| /api/weather | wx:{lat\_2dp}:{lon\_2dp} | 2–3 decimals (≈1–3 km) | **1 min** | On TTL; manual bust on config change. |
| /api/geocode/search | geo:q:{q}:limit:{n} | n/a | **10 min** | On TTL. |
| /api/geocode/reverse | geo:rev:{lat\_4dp}:{lon\_4dp} | 4 dp (≈11 m) | **10 min** | On TTL. |
| /api/activities | poi:{lat\_3dp}:{lon\_3dp}:r:{radius}:t:{type} | 3 dp (≈100–150 m) | **2 min** | On TTL; manual bust on category map change. |
| /api/recommendations | Derived (weather + activities) | same as inputs | **1 min** | On TTL; re-compute on weather cache miss. |

Client hint: treat responses as fresh for 2–10 minutes depending on endpoint; always allow manual refresh.

## Testing & CI

* **Pest** feature tests cover:
  + Validation, caching, retry/backoff, recommendation rule, normalization
* **CI** (GitHub )
  + Backend: composer install → Pint (check) → seed .env.testing + fixtures → php artisan test
  + Frontend: npm ci → ESLint + Prettier **--check** → npm test --if-present

## Future notes / extensions

* Auth (JWT/Bearer w/ oauth integration) and per-app rate limiting
* Stronger weather/POI enrichment (opening hours, price tier, popularity)
* Vector tiles or offline tiles for maps
* API versioning (/v1)