OGAP – Architecture Decision Records

ADR-001: Supabase Backend-as-a-Service

1. Context
   - The application requires authentication, persistent storage of parking spots, credit transactions, and real-time change propagation.
   - Delivery timeline is constrained by an academic semester, limiting capacity to operate and secure a bespoke backend.
   - The team prefers managed hosting to reduce operational overhead while retaining SQL flexibility.
2. Decision
   - Adopt Supabase (Postgres, Auth, Realtime, Edge Functions) as the primary backend and data platform.
3. Alternatives Considered
   - Custom backend (Node/Go with managed Postgres): greater control but higher build/ops load and longer lead time.
   - Firebase (Firestore + Auth): strong BaaS, but weaker SQL semantics and more friction for relational queries already modeled in SQL migrations.
4. Trade-offs
   - Gains: rapid delivery, hosted Auth/DB/Realtime, SQL-first schema, type-safe client generation.
   - Downsides: vendor coupling to Supabase APIs and billing; limited control over low-level tuning compared to self-managed Postgres.
5. Consequences
   - Short-term: faster feature delivery with minimal DevOps; migrations and edge functions align with Supabase tooling.
   - Long-term: scaling stays manageable if Supabase tiers fit growth; migration off Supabase later would require effort to reimplement Auth/Realtime and replace platform-specific functions.

ADR-002: Realtime Subscriptions for Parking State

1. Context
   - Parking spot availability and handshake states must be reflected to users with minimal latency.
   - The browser is already connected to Supabase; adding a separate realtime channel would increase complexity.
2. Decision
   - Use Supabase Realtime (Postgres changes channels) to subscribe to `parking_spots` and `handshake_deals` updates in the client.
3. Alternatives Considered

    - Polling REST endpoints on an interval: simpler to reason about, but higher latency and wasted requests.
    - Custom WebSocket service: flexible, but duplicates capabilities already provided by Supabase and increases ops burden.

4. Trade-offs
    - Gains: low-latency UI updates, efficient network usage, reduced client logic for synchronization.
    - Downsides: reliance on Supabase channel availability; client-side complexity to manage subscriptions and cleanup.

5. Consequences
    - Short-term: responsive map and handshake flow with minimal additional backend code.
    - Long-term: channel scaling and multiplexing must be monitored; fallback strategies (e.g., degraded polling) may be required if channel limits or connection drops become frequent.

ADR-003: Credit-Based Handshake Incentive

1. Context
    - The product needs a fair mechanism to motivate users to announce and hand over parking spots.
    - Pure altruistic sharing risks low participation; monetary payments introduce compliance and payment integration overhead.

2. Decision
    - Implement an in-app credit economy tied to the handshake flow, with balances, transactions, and packages stored in the backend.

3. Alternatives Considered
    - Non-incentivized sharing: minimal complexity but likely insufficient engagement.
    - Direct payments (cash or third-party payment rails): clearer real-world value but higher legal, UX, and integration complexity for the academic scope.

4. Trade-offs
    - Gains: lightweight incentive, easy to simulate within academic constraints, supports gamification without payment compliance.
    - Downsides: credits are a proxy value and may not map to real-world willingness to pay; requires careful anti-abuse rules and balance synchronization.

5. Consequences
    - Short-term: higher engagement in offers/requests; straightforward accounting via Supabase tables.
    - Long-term: future monetization would need conversion between credits and currency or replacement of the credit model; fraud prevention and rate limits may be necessary as usage grows.

ADR-004: Web-First React with Capacitor Wrappers

1. Context

- Primary target is fast iteration and broad reach via the web.
- Mobile presence is desired for demo and device-level access (location), but native feature depth is limited by timeline and team size.

2. Decision

    - Build the core experience as a responsive React/Vite web app and wrap it with Capacitor for iOS/Android shells.

3. Alternatives Considered

    - Pure native (Swift/Kotlin): best platform integration and performance, but duplicative effort and slower delivery.
    - Web-only (no Capacitor): simplest delivery but weaker mobile distribution story and reduced access to native bridges if later needed.

4. Trade-offs

    - Gains: single codebase, fast iteration, ability to ship to app stores via Capacitor shells.
    - Downsides: some native UX/performance gaps versus fully native apps; added build tooling for Capacitor sync and platform-specific packaging.

5. Consequences

    - Short-term: rapid feature delivery and workable mobile shells for demos.
    - Long-term: if deeper native capabilities are required (offline maps, background services), further native plugins or partial rewrites may be needed.

ADR-005: Heuristic Parking Availability Model

1. Context

    - The UI displays a probability that a spot remains free to guide user decisions.
    - Data for training a machine learning model is unavailable within the project timeframe; explainability is preferred for academic evaluation.

2. Decision

    - Use a handcrafted heuristic that decays availability probability over time, with different ranges for peak vs off-peak hours.

3. Alternatives Considered

    - Machine learning model (e.g., historical survival analysis): potentially more accurate but requires datasets, feature engineering, and infra not available now.
    - Static binary availability (no probability): simplest but offers poor guidance and less user trust.

4. Trade-offs

    - Gains: deterministic, explainable behavior; minimal dependencies and no training pipeline.
    - Downsides: limited adaptability to real-world patterns; may misestimate availability under atypical conditions.

5. Consequences

    - Short-term: immediate, predictable signal to users; easy to tune manually.

- Long-term: model may need replacement or augmentation with data-driven approaches as telemetry accumulates; maintainability depends on periodically revisiting heuristics.