OGAP – System Workflows & Data Flows

1. High-Level System Overview

- **Frontend:** React/Vite single-page app with map-based UI; interacts with Supabase via client SDK; subscribes to realtime channels for parking spots, handshakes, credits, and presence.
- **Backend (Supabase Postgres):** Tables for `parking_spots`, `handshake_deals`, `user_credits`, `credit_transactions`, `parking_history`, and `profiles`. Business logic partly encoded in Edge Functions.
- **Realtime layer:** Supabase Realtime channels propagate Postgres changes to connected clients; presence channel tracks online users; clients subscribe to per-table channels and update local state immediately.
- **Edge Functions:**
  - `process-credits`: enforces credit rules for reporting spots and completing handshakes; mutates `user_credits`, `credit_transactions`, `handshake_deals`, `parking_history`, `parking_spots`.
  - `cleanup-parking-spots`: time-based cleanup of stale `parking_spots` and expired `handshake_deals`; resets associated spots.
  - `mapbox-proxy`: returns Mapbox token to the frontend and proxies requests (avoids exposing secret key).
- **External services:** Mapbox GL for map tiles and geospatial rendering; browser Geolocation for current position.

ASCII sketch:

```
[Browser SPA] --HTTP/WS--> [Supabase Realtime + Postgres]
      |                          |\
      |                          | Edge Functions (credits, cleanup, mapbox)
      v                          v
   Mapbox GL tiles        Postgres tables + channels
```

2. Core User Workflows

2.1 Reporting a parking spot when leaving

1. User indicates they are leaving and provides/uses current location.
2. App writes/updates `parking_spots` (available=true, lat/lng, available_since now).
3. App may create a `handshake_deal` (status=open, departure_time).
4. Edge Function `process-credits` action `new_spot_reported` awards credits to the giver.
5. Realtime broadcasts the new/updated spot and deal to all subscribers.

2.2 Discovering an available parking spot

1. Client subscribes to `parking_spots` channel on load.
2. Client fetches current spots and overlays Mapbox markers with probability heuristic.
3. As updates arrive (insert/update/delete), markers refresh and nearest-spot helper updates.
4. User taps a spot to view details and start navigation or request a handshake.

2.3 Creating and completing a handshake

1. **Offer:** Giver calls `createHandshakeOffer` → insert into `handshake_deals` (status=open).
2. **Request:** Receiver calls `requestDeal` → update to `pending_approval` with `receiver_id`.

3. **Accept/Decline:** Giver accepts (`status=accepted`) or declines (reset to open).
4. **Complete:** At departure, giver triggers `process-credits` action `complete_handshake` ; function validates status/roles, transfers credits (giver +20, receiver -10), sets deal `status=completed` , marks `parking_spots.available=false` , and records `parking_history` for giver.
5. Client subscribed to `handshake_deals` and `parking_spots` receives updates; receiver auto-starts a parking session locally when completion update arrives.

2.4 Earning and spending credits

1. **Earning:**
   - Reporting spot → `process-credits` (`new_spot_reported`) increments `user_credits` and logs `credit_transactions` .
   - Completing handshake as giver → `process-credits` credits +20 and logs transaction.
   - Welcome bonus on first interaction (if no `user_credits` row) → +20 and transaction.
2. **Spending:**
   - Completing handshake as receiver → `process-credits` debits 10 credits and logs transaction.
3. **Visibility:** Clients subscribe to `user_credits` changes per user; balance changes trigger UI updates and animations.

3. Data Flow per Workflow

Reporting a parking spot when leaving

- **Writes:** `parking_spots` (available=true, coords, available_since), optionally `handshake_deals` (status=open), `credit_transactions` , `user_credits` (via Edge Function).
- **Realtime:** `parking_spots` and `handshake_deals` channels publish inserts/updates.
- **Edge Functions:** `process-credits` (`new_spot_reported`) adjusts balance; `mapbox-proxy` may provide token for map rendering.
- **Propagation:** Other clients receive spot/deal inserts and render immediately; credit changes propagate only to the owner via filtered channel.

Discovering an available parking spot

- **Reads:** `parking_spots` (initial fetch), `handshake_deals` (open deals for map/UI).
- **Realtime:** Subscriptions keep markers and deal lists current.
- **Edge Functions:** None on read path; `mapbox-proxy` for token retrieval.
- **Propagation:** Any insert/update/delete on `parking_spots` or `handshake_deals` updates all connected clients' views.

Creating and completing a handshake

- **Writes:** `handshake_deals` (insert/open → pending_approval → accepted → completed or cancelled), `parking_spots` (may be hidden when deal opens; set unavailable on completion), `parking_history` (on completion), `user_credits` , `credit_transactions` .
- **Realtime:** `handshake_deals` channel pushes lifecycle updates; `parking_spots` channel reflects availability changes.
- **Edge Functions:** `process-credits` (`complete_handshake`) enforces role/status, applies credit transfers, updates tables atomically.
- **Propagation:** Giver/receiver UIs update via deal status changes; receiver auto-starts parking session when `status=completed` for their `receiver_id` .

Earning and spending credits

- **Writes:** `user_credits`, `credit_transactions`; triggered by Edge Function actions.
- **Realtime:** `user_credits` channel filtered per user_id notifies balance changes.
- **Edge Functions:** `process-credits` (`new_spot_reported`, `complete_handshake`, `check_balance`).
- **Propagation:** Only the affected user's client updates (channel filter by `user_id`).

4. Realtime Behavior

- **Channels used:**
  - `parking_spots`: broadcast inserts/updates/deletes to all clients.
  - `handshake_deals`: broadcast lifecycle changes; clients adjust active/open lists.
  - `user_credits`: filtered per `user_id` for balance updates.
  - `handshake-completion` (per-user filter in UI): listens for completed deals where `receiver_id` matches current user.
  - Presence channel `online-users`: tracks active connections for UI display.
- **Concurrent interactions:**
  - Multiple receivers can request the same open deal, but state transitions are guarded by status checks (e.g., `status=open` when setting `pending_approval`, `status=pending_approval` when accepting).
  - Supabase row-level updates ensure only one accepted receiver; others see updated status via realtime and must retry elsewhere.
- **Consistency:**
  - Clients optimistically update lists but rely on Postgres as source of truth; realtime updates reconcile discrepancies.
  - Credit updates are centralized in Edge Functions to avoid client-side race conditions.

ASCII timeline (handshake):

```
Giver INSERT open deal --> Realtime -> All clients
Receiver UPDATE pending_approval --> Realtime -> Giver UI prompts
Giver UPDATE accepted --> Realtime -> Receiver notified
Giver invoke process-credits (complete_handshake)
    -> Updates deals, credits, parking_spots, history
    -> Realtime broadcasts updated deal + spot
Receiver client starts parking session on completion
```

5. Cleanup & Expiration Flows

- **Stale parking spots:** `cleanup-parking-spots` runs time-based cleanup (daytime threshold 30m, nighttime 90m); deletes available spots older than threshold. Realtime delete events remove markers from clients.
- **Expired handshakes:** Same function cancels `handshake_deals` that remain `open` past `departure_time + threshold`; sets status `cancelled` and reopens associated `parking_spots` (available=true, available_since=now). Realtime updates ensure all clients see reopened availability.
- **Abandoned deals:** Clients can cancel manually (`status=cancelled`); channel update removes deal from lists and frees spot visibility.

- **Local sessions:** Frontend persists active parking session in localStorage; if a completed handshake is received for a receiver without a session, the client creates one to preserve state across reloads.