PART 1: THE PERFECT WORKFLOWS

How the software behaves, step-by-step.

🅰️ APP 1: The Driver App (Demand Side)

Goal: Find a spot, book it, park, and leave without talking to anyone.

1. The Search (Discovery)

- **User Action:** Opens app. Grants "Location Permission."
- **System Logic:** App sends GPS coordinates to Supabase. PostGIS calculates distance.
- **UI:** Shows a Map with pins.
    - *Green Pin:* Available now.
    - *Red Pin:* Full.
- **User Action:** Taps a pin. Sees "Blue Haven Society - ₹50/hr - 4 Slots Left."

2. The Booking (Commitment)

- **User Action:** Selects "2 Hours" -> Clicks "Book Now."
- **System Logic:**
    - Checks availability one last time (prevent double booking).
    - Calculates Total: ₹100.
- **User Action:** Pays via UPI (Razorpay/Stripe).
- **System Logic:** On success, creates a booking in DB with status: upcoming. Generates a unique **QR Code**.

3. The Arrival (Check-In)

- **User Action:** Arrives at gate. Shows QR Code to Guard.
- **Guard Action:** Scans code.
- **System Logic:**
    - DB Status updates: upcoming → active.
    - **Start Timer** begins ticking.

4. The Warning (The "Nudge")

- **System Logic:** 1 hour 40 mins passed (20% time remaining).
- **Notification:** User gets SMS/Push: *"Your parking expires in 20 mins. Extend now to avoid penalty?"*
- **User Action:** Can click "Extend" (+1 hour) and pay difference. OR ignore and rush back.

5. The Exit (Check-Out)

- **Scenario A: On Time**
    - Guard Scans. Screen says "APPROVED." Gate opens.
    - DB Status: completed.
- **Scenario B: Late (The Penalty)**
    - Guard Scans. Screen says **"PAYMENT PENDING: ₹60"**.
    - User App Pop-up: "You overstayed by 30 mins. Pay ₹60 to exit."
    - User Pays.
    - **Realtime Sync:** Guard's screen turns Green instantly. Gate opens.

🅱️ **APP 2: The Partner App (Supply Side)**

Goal: Monetize empty space and manage security.

1. The Listing (Setup)

- **Partner Action:** Clicks "Add Spot."
- **System Logic:** Captures GPS Location.
- **Partner Action:** Uploads photo of gate. Sets Price (₹50/hr). Sets Capacity (10 cars).
- **System Logic:** Spot is now "Live" on the Driver Map.

2. The Dashboard (Guard Mode)

- **UI:** Big "SCAN QR" button at bottom. Live counter: "6/10 Spots Filled."
- **Partner Action:** Guard stands at gate.

3. The Entry Scan

- **Action:** Guard taps "Scan" -> Camera opens -> Scans Driver's phone.
- **Feedback:**
    - ✅ **Green Screen:** "Valid. Let them in." (Shows Car Number Plate).
    - ❌ **Red Screen:** "Invalid Ticket" or "Wrong Location."

4. The Exit Scan

- **Action:** Guard taps "Scan" -> Scans Driver's phone.
- **Feedback:**
    - ✅ **Green:** "Paid & Clear. Open Gate."

- ⚠️ **Orange:** "Overstay Penalty Due. Do NOT Open Gate."
  - *Guard says:* "Sir, please check your app and pay."
  - *Wait:* Once user pays, screen turns Green automatically.

PART 2: THE MAKING PROCESS (Development Roadmap)

How we build it today, and how we scale it tomorrow.

This is your **Strategic Engineering Plan**. We are building in **Phases**.

PHASE 1: The "Web MVP" (NOW)

Goal: Build a mobile-friendly website (parkeasy.com) that *feels* like an app.
Why: Fastest to build, cheapest (no App Store fees), easy to fix bugs.

- **Step 1: The Brain (Supabase Backend)**
  - We create the Database tables (users, bookings, spots).
  - We enable **PostGIS** (The Map Engine).
  - We write the "Business Logic" (SQL functions) to calculate distances and penalties.
  - *Value:* This "Brain" is permanent. It will power the website now AND the mobile apps later.
- **Step 2: The Face (Next.js Frontend)**
  - We build one project with two folders: /driver and /partner.
  - We use **Tailwind CSS** to make buttons big and touch-friendly (Mobile-First Design).
  - We use react-qr-reader to access the camera through the Chrome/Safari browser.
- **Step 3: The Simulation (The Pitch)**
  - You save the website to your phone's Home Screen. It creates an icon.
  - When you tap it, it opens full screen (no address bar).
  - *Result:* To an investor, it looks 99% like a real app.

PHASE 2: The "Native" Future (LATER)

Goal: Launch actual apps on Play Store and App Store.
When: After you get funding or have 500+ users.

- **The Good News:** You do **NOT** need to rebuild the backend.
    - The User Database? **Same.**
    - The Booking History? **Same.**
    - The Logic? **Same.**
- **The Process:**

1. **Hire Mobile Developers:** You hire a Flutter or React Native developer.
2. **Hand Over the Keys:** You give them your Supabase API Keys and the project_context.md.
3. **The Instruction:**
    - "Hey Developer, don't build a backend. Just connect the Mobile UI to my existing Supabase project."
4. **The Result:**
    - A Driver logs in on the Website → Books a spot.
    - They download the App → The booking is already there.
    - Everything stays in sync perfectly.