

Rolling Thunder — demand-verified ordering for Banglador's Ultimate

Mobile app concept

Draft — not yet published

The idea in one line

You can't just order the Ultimate. You signal the craving, the crowd piles on, the AI verifies the demand is real — and *then* the order window opens, DoorDash-style.

How it works

1. **Crave.** You open the app, tap "I NEED THE ULTIMATE." That's a demand signal, pinned to your zone — not an order. One signal per user per window; no spam-tapping your way to lunch.
2. **Aggregate.** Signals pool by geozone and time window. The app shows a live thunder meter: how close your zone is to the threshold.
3. **AI verifies.** When the meter fills, the AI checks: is the demand legitimate (not bots, not one guy with five phones)? Can the kitchen handle the batch? Is courier coverage there? If yes — **approved.**
4. **Order window opens.** Push goes out: "ROLLING THUNDER IN YOUR ZONE — 30 MINUTES." Now it's a normal food order: cart, checkout, payment, live tracking.
5. **Fulfillment.** Orders batch to Banglador's kitchen display as one wave, couriers get batched dispatch, food moves like a DoorDash order.

Why gate it at all

- **Hype as a feature.** Scarcity you can watch fill up is more fun than a menu that's always there. The thunder meter *is* the marketing.
- **Kitchen efficiency.** The kitchen never fires up for two orders. Every wave is a verified batch — predictable prep, less waste.
- **The AI is the bouncer, not the waiter.** It doesn't take your order; it decides whether the party happens. That's the novelty — demand-gated commerce instead of always-on commerce.

The rules

1. **One signal per human per window.** Device attestation + reputation score. Gamers get shadow-throttled.
2. **Thresholds are public.** Everyone sees the meter and the number. No black-box vibes.
3. **Kitchen capacity is real.** The AI can't approve what Banglador's can't cook. Capacity API is the hard ceiling.
4. **Windows expire.** 30 minutes, then the thunder rolls elsewhere. FOMO is load-bearing.
5. **Refunds are instant if the wave fails.** Kitchen goes down mid-wave? Everyone gets their money back before they can complain.

Scoring success (if this were the hackathon)

- Signal-to-order conversion: do meters actually turn into checkouts?

- Fraud rate: how many waves were gamed?
 - Kitchen throughput: orders per wave-hour vs. normal service.
 - Retention: do users come back for the *next* thunder, or was it a one-tap novelty?
-

Reference solution architecture: see [rolling-thunder-architecture.html](#).