finish Phase 2 and make a clean demo build:

- Rename remaining “Humans Square” strings to **World Mall**.
- Fix a regression where **guest posts are blocked** with a red “World ID verification required” banner even when within guest limits.
- Loosen **Guest Mode** to a short sentence: **60 chars**, **3 msgs/day**, **90s cooldown**, server-enforced.
- Implement **World ID Cloud** server verification (**v2** endpoint) and flip role to `verified` on success (no PII; store **hashed** nullifier).
- Keep existing **Permit2 foundations** but keep **NO real token transfers** (safe no-ops behind a feature flag).
- Add small diagnostics: `/api/policy`, `/api/me`, plus simple logs/metrics.
- Preserve Replit single-port deploy and health check.

Constraints & guardrails

- **Do not break** current production start behavior: `.replit` runs `NODE_ENV=production node dist/index.js` and serves on port 80. No dev servers in prod.
- **No PII** storage: only store **SHA-256 of the nullifier hash** from World ID.
- **Do not gate** guest-allowed posts behind verification. Only guard **verified-only features** (long posts, stars, reports, Work Mode).
- **Permit2**: verification endpoint/utilities OK; **no token transfers**; keep behind `ENABLE_PERMIT2` flag (default off). Keep project compiling cleanly.
- Keep DB **idempotent migrations** and **unique constraints** (see below).

What to change (high level plan)

1. **Rename polish — “Humans Square” → “World Mall”**
    - Search the repo (exclude `node_modules`) and replace visible strings in UI components, meta tags/OG, headings, and any constants.
    - Keep the “For full functionality, open in World App” notice for web; it’s fine. Web should still allow guests to say hi.
2. **Shared policy & env wiring**
    - Add a small `POLICY` config on the server (constants or `server/config.ts`) with **defaults**:
        - `guestCharLimit=60`, `guestDaily=3`, `guestCooldownSec=90`
        - `verifiedCharLimit=240`, `verifiedPerMin=5`, `verifiedPerHour=60`, `verifiedPerDay=200`

        - World ID envs: `WORLD_ID_APP_ID`, `WORLD_ID_ACTION`, optional `WORLD_ID_API_BASE` defaulting to `https://developer.worldcoin.org`
    - Expose `GET /api/policy` returning `{ guestCharLimit, guestDaily, guestCooldownSec, verifiedCharLimit, verified:{perMin,perHour,perDay} }`.
    - Expose `GET /api/me` returning `{ role: 'guest' | 'verified' | 'admin' }` based on session/cookies.
3. **Session + cookies**
    - Ensure we set a lightweight cookie (e.g., `wm_sid` for guests; `wm_uid` after verify). If absent, generate a UUID session id and set it (httpOnly, sameSite=lax, 1-year).
    - Do **not** require login to post as guest; use session for quotas/cooldowns.
4. **DB constraints & indexes (Drizzle)**
    - Ensure the following exist (create or verify):
        - `verifications`: `user_id`, `nullifier_hash_hashed` (**UNIQUE** on `nullifier_hash_hashed`), `created_at`.
        - `stars`: **UNIQUE** `(user_id, message_id)`.
        - `reports`: **UNIQUE** `(user_id, message_id)`.
        - `messages`: indexes for `(room, created_at)` and `(user_id, created_at)`.
        - `guest_sessions`: `id` (session id), `last_post_at`, `post_count_today`, `day_bucket`.
    - Add a **smoke query** on boot to fail fast if DB or env is missing.
5. **Message creation route — fix guest regression**
    - In the `POST /api/messages` route:
        - If `role === 'guest'`:
            - Enforce `text.length <= guestCharLimit`.
            - Enforce `guestDaily` and `guestCooldownSec` using `guest_sessions`.
            - **Accept** and insert message (room = `global`, sessionId = cookie).
            - **Return success**; **never** show “World ID verification required” for a valid guest post.
        - If `role !== 'guest'` (verified/admin): enforce `verifiedCharLimit` and existing rate limits; then accept.
    - Ensure **stars/reports/work-mode** routes **require verified**; message create must not.
6. **World ID Cloud verification (server, v2)**
    - `POST /api/verify/worldid` accepts `{ nullifier_hash, proof, merkle_root, verification_level, action, signal? }`.

    - Validate `action` matches `WORLD_ID_ACTION`. If client sends `signal`, hash it on server (keccak/field hash) to `signal_hash` to forward.
    - Call **Cloud verify v2**: `POST ${WORLD_ID_API_BASE}/api/v2/verify/${WORLD_ID_APP_ID}` with JSON body `{ nullifier_hash, merkle_root, proof, verification_level, action, (optional) signal_hash }`. Use `Content-Type: application/json` and a simple `User-Agent`.
    - On success:
        - Compute `sha256(nullifier_hash)` and store it in `verifications` (**unique**).
        - Upsert a `users` row (or update existing) with `role='verified'`.
        - Set cookie `wm_uid` to bind session to user.
        - Return `{ ok:true, role:'verified' }`.
    - On failure: `400` with a clear error, no role changes.
7. **Permit2 foundations (keep as no-ops)**
    - Keep existing `/api/permit2/verify` and EIP-712 utils. Hide behind `ENABLE_PERMIT2` env flag (default `0`).
    - Do not wire any token transfers. Make sure the route cleanly returns `{ ok:true }` when enabled and signature checks out; otherwise `{ ok:false, error }`.
    - In World App Dev Portal, do **not** add tokens (we're demoing verification, not payments).
8. **Client updates**
    - On load, call `/api/policy` and `/api/me`. If role is guest, set composer limit to **policy.guestCharLimit (60)** and show **"Guest Mode"** badge and **Verify** CTA.
    - Show a **live counter** and a "Guest limit" pill at 60 chars.
    - Map server error codes:
        - `403` length → "Guest limit is 60 characters. Verify to unlock full chat."
        - `429` cooldown/quota → show server message ("Please wait Ns…" or "3/day reached").
        - Never display "World ID verification required" for guest-allowed posts.
    - After a successful `/api/verify/worldid`, update UI state to `role='verified'` **without full reload** and bump limit to 240.
9. **Observability**
    - Log on message create: `[post] role=<role> accepted|blocked reason=<reason>`.
    - Log on verify success/fail: `[worldid.verify]`.
    - Optionally count guest accepts/blocks in simple counters for the demo.
10. **Feature flag fallback**

- If `DISABLE_WORLDID=1`, treat everyone as guests with guest limits; keep Verify button visually present but route can return a friendly "temporarily unavailable" message. This guarantees a working demo even if Cloud is flaky.

ENV checklist (set in Replit Secrets)

- `WORLD_ID_APP_ID` = (from World App Dev Portal)
- `WORLD_ID_ACTION` = e.g., `world-mall/verify`
- `WORLD_ID_API_BASE` = `https://developer.worldcoin.org` (default if unset)
- `GUEST_CHAR_LIMIT=60`, `GUEST_DAILY=3`, `GUEST_COOLDOWN_SEC=90` (optional overrides)
- `VERIFIED_CHAR_LIMIT=240`, `VERIFIED_PER_MIN=5`, `VERIFIED_PER_HOUR=60`, `VERIFIED_PER_DAY=200`
- `ENABLE_PERMIT2=0` (default)
- Optional: `DISABLE_WORLDID=0`

Acceptance tests (run these after you implement)

Policy sanity

- `GET /api/policy` → `{ guestCharLimit:60, guestDaily:3, guestCooldownSec:90, ... }`

Guest happy path

- On Web or Mini App as guest:
  - Composer shows **Guest Mode**, **0/60** counter.
  - Send: “hey, how’s it going everyone?” (≤60) → **succeeds** (no red verify banner).
  - Immediate second send → **429** with human message (cooldown). After **90s** → can send again, up to **3/day**.

Guest over-limit

- 61+ chars → **403** “Guest limit is 60 characters. Verify to unlock full chat.”

Verify flow

- Use World App to obtain proof; frontend POSTs proof to `/api/verify/worldid`.
- Server calls Cloud verify v2; on success:
  - Role becomes **verified** (UI updates without reload).
  - Composer limit = **240**, stars/report/work-mode visible.
  - Duplicate verification attempts with same nullifier do **not** create duplicates (unique enforced).

Regression fixed

- The red “World ID verification required” banner **never** appears for guest-allowed posts.

Permit2

- With `ENABLE_PERMIT2=1`, `/api/permit2/verify` accepts a valid EIP-712 signature and returns `{ ok:true }`. No transfers occur. With flag off, route is disabled.

Deploy

- App runs on a single port; `.replit` start continues to `node dist/index.js`.
- Health endpoint still works.

Work style / how to proceed

1. Scan the repo to confirm file layout (server routes, schema, client composer).
2. Make the smallest set of changes to satisfy the above.
3. Create or update Drizzle migrations safely (idempotent). Verify constraints exist.
4. Implement routes and client tweaks.
5. Add lightweight logs.
6. Build → run locally in Replit → execute the acceptance checks.
7. Deploy to Autoscale.
8. Post a short summary of what changed and the exact files touched.

Important: If anything is ambiguous, choose the simplest approach that satisfies the acceptance tests and keeps the demo resilient.