GOAL:
Update the existing Cleanups feature so each cleanup operation supports "Paid Shifts" for roles beyond basic trash pickup. Users can browse and apply to paid roles like Safety Lead, Organizer, Logistics Lead, Data Verifier, etc. Pay is at least local minimum wage (use a single configurable MIN_WAGE_USD constant for the MVP).

IMPORTANT PRODUCT RULES:
- Do NOT pay for the same "volunteer cleanup" task. Paid shifts are for leadership/logistics/safety/data roles.
- Keep volunteer participation intact (existing cleanup flow stays; this is an add-on).
- Paid shifts should require certifications (or proof) and optionally require approval.
- Build end-to-end: database + backend APIs + frontend UI + seed data.
- Remove/avoid any references to "Education Hub" if still present.

SCOPE (MVP for hackathon):
1) Cleanups tab changes:
- In the Cleanups list view, each cleanup card shows:
  - urgency score (existing)
  - funding progress (existing if present)
  - NEW: "Paid Shifts Available" badge with count (if >0)
  - Button: "View Operation"
- Clicking a cleanup opens a detail page (or modal) with tabs:
  - Overview (existing info)
  - Route/Weather (existing if present)
  - NEW: "Jobs & Shifts"
  - NEW: "Applicants" (visible to organizer/admin only; can be hidden behind env flag)

2) Jobs & Shifts per Cleanup:
- Each cleanup can define multiple shift postings:
  Example roles (minimum set):
  - SAFETY_LEAD
  - ORGANIZER
  - LOGISTICS_LEAD
  - DATA_VERIFIER
  - DRONE_TECH (optional)
- Each posting includes:
  - title, role_type
  - description
  - hourly_wage_usd (must be >= MIN_WAGE_USD)
  - estimated_hours
  - location (inherit from cleanup or override)
  - shift_start, shift_end (or date + time window)
  - slots_total, slots_filled

  - minimum_certifications (array of strings)
  - required_equipment (array of strings, optional)
  - status: OPEN / FILLED / CLOSED
  - eligibility_note (short)
- A user can click “Apply” and submit:
  - name, email
  - certifications they claim (checkbox list + freeform)
  - short statement (why fit)
  - optional links (resume/portfolio)
- Application status lifecycle:
  - SUBMITTED → APPROVED / REJECTED / WAITLISTED
- If approved, it increments slots_filled (until FILLED).

3) Funding logic (lightweight):
- Add an “Unlocked by funding” mechanism:
  - Each shift can be either:
    a) ALWAYS_OPEN (for demo), or
    b) UNLOCKED_WHEN_FUNDED with a required_usd threshold.
- If your project already tracks donations per cleanup, integrate:
  - if cleanup_funds_raised >= required_usd, shift is OPEN
  - else show as LOCKED with “Unlock by reaching $X”
- If no existing donation totals, create a simple
`cleanup_funds_raised_usd` numeric field for MVP and seed it.

4) Certifications:
- Create a simple Certifications system:
  - A table of certifications (e.g., “First Aid/CPR”, “OSHA-10”,
“Hazardous Waste Handling”, “Drone Spotter”, “Incident Reporting”).
  - Shift postings reference required certs (many-to-many or store as text
array; prefer relational if easy with Drizzle).
  - On application, store claimed certs + evidence link optional.
- No need for full verification flows; just store + display.

5) Admin/Organizer review:
- Add a basic review screen for each cleanup:
  - list applicants by shift
  - approve/reject/waitlist
- Gate access with `ADMIN_MODE=true` env var (or an existing auth/admin
mechanism if present).
- On approve, update application status + slots_filled.

DATABASE (Postgres + Drizzle):
Assume you already have a `cleanups` (or `operations`) table. Reuse it.
Add new tables (names flexible; keep consistent style):

A) certifications
- id (uuid)

```
- name (unique)
- description (nullable)

B) cleanup_shifts
- id (uuid)
- cleanup_id (fk)
- role_type (ENUM)
- title
- description
- hourly_wage_usd (numeric)
- estimated_hours (numeric)
- shift_start (timestamp)
- shift_end (timestamp)
- slots_total (int)
- slots_filled (int default 0)
- status (ENUM: OPEN, FILLED, CLOSED, LOCKED)
- unlock_type (ENUM: ALWAYS_OPEN, UNLOCKED_WHEN_FUNDED)
- unlock_required_usd (numeric nullable)
- location_override (text nullable)
- eligibility_note (text nullable)
- created_at

C) cleanup_shift_required_certs (if relational)
- shift_id (fk)
- cert_id (fk)

D) shift_applications
- id (uuid)
- shift_id (fk)
- cleanup_id (fk) (denormalize for queries)
- applicant_name
- applicant_email
- statement (text)
- evidence_url (text nullable)
- claimed_certs (text[] or join table)
- status (ENUM: SUBMITTED, APPROVED, REJECTED, WAITLISTED)
- created_at
- reviewed_at (nullable)

Also add (if needed) to cleanups:
- funds_raised_usd (numeric default 0) OR derive from donations table.

API (Node/Express):
Implement endpoints:
- GET /api/cleanups (existing) should include `paid_shift_count`
- GET /api/cleanups/:id (existing) should include shifts + funding totals
- GET /api/cleanups/:id/shifts
```

- POST /api/cleanups/:id/shifts (admin only) [optional]
- POST /api/shifts/:shiftId/apply
- GET /api/shifts/:shiftId/applications (admin only)
- POST /api/applications/:id/review { status: APPROVED|REJECTED|WAITLISTED } (admin only)

Validation rules:
- hourly_wage_usd >= MIN_WAGE_USD
- cannot approve if slots_filled >= slots_total
- approving increments slots_filled; if equals slots_total, shift status becomes FILLED

Real-time updates:
- If you already use SSE, emit an event `cleanup_shift_update` on apply/review so the UI updates live; otherwise poll every 30–60s (acceptable).

FRONTEND (React/TS):
- Update Cleanups list to show paid shift badge + link to details.
- Add cleanup detail route:
  /cleanups/:id
  with tabbed UI and a “Jobs & Shifts” tab.
- Jobs & Shifts tab:
  - show list of shifts (cards)
  - show role, wage, hours, time, slots, required certs, status
  - if LOCKED, show funding needed
  - Apply flow: modal form with fields described
- Applicants tab (admin only):
  - grouped by shift
  - approve/reject/waitlist buttons

SEED DATA:
- Seed at least 6 cleanups with varied funding totals.
- For each cleanup, seed 2–5 shifts across different roles.
- Seed 8–12 certifications.
- Seed some applications (including pending) for at least 2 cleanups.
- Ensure at least one cleanup has LOCKED shifts that unlock when funding is high enough.

CLEANUP / CONSISTENCY:
- Keep styling consistent with existing app.
- Do not break existing cleanup functionality.
- Add a short CHANGELOG.md describing assumptions and new routes/endpoints.

Proceed to implement now without asking questions. Make reasonable assumptions, follow existing project patterns, and keep code readable.