You are a full multidisciplinary dev team (UI/UX designer, front-end and back-end developer, AI/NLU engineer, QA, and DevOps). Implement the missing PS-2 WhatsApp Chatbot intake features and admin dashboard functionality so the repository becomes fully PS-2 compliant. Use the existing CyberSathi app UI structure; maintain styling and Tailwind conventions. Reference the PS-2 problem statement for exact fields and workflow. :contentReference[oaicite:1]{index=1}

SCOPE (deliver end-to-end, production-ready prototype):

1. Add "Chatbot / WhatsApp Complaints" pipeline:
 - Endpoint(s) to accept complaints from WhatsApp webhook (simulated). Save all PS-2 fields:
   - name, guardian_name, dob (ISO date), phone, email, gender, village, post_office, police_station, district, pin_code, account_number (if provided), scam_main_category, scam_sub_category, acknowledgement/reference_number (generated), attachments metadata.
 - Generate unique reference number format: `CS-YYYY-XXX` where XXX is zero-padded sequence.
 - Validate phone (E.164), email (regex), pin_code (6 digits), DOB (>= 12 yrs).
 - Store attachments (images, PDFs) in object storage. Use local storage in dev, but code must support S3-compatible endpoint via env var.

2. Add Admin UI pages/components:
 - "Chatbot Complaints" table with server-side pagination, filters (by status, district, scam_category, date range), search (ref no / phone / name).
 - "Complaint Detail" modal/page showing all PS-2 fields and evidence previews (thumbnail + download).
 - "Status Check" tool: enter acknowledgement number or phone ▯ show complaint summary + history.
 - "Account Unfreeze Requests" list with filters and detail view.
 - "Other Queries" list (free-text queries from chatbot) and ability to convert to complaint.
 - Agent follow-up panel: assign agent, record call/text follow-ups, set follow-up due date.

3. Fraud type model:
 - Implement two-tier taxonomy:
   - Financial Fraud (exactly the 23 subtypes from PS-2).
   - Social Media Fraud (the listed platform subtypes).
 - UI: when selecting Financial Fraud, show the 23 checkable subtypes; for Social Media, show platform-specific subtypes.

4. Evidence & Document Upload:
 - Backend API to accept uploads (multipart/form-data).
 - Preview thumbnails and full-view on admin page.
 - Tag uploads by category (identity, bank_statement, transaction_screenshot, request_letter, url_proof).
 - Extract metadata from images: filename, size, mimetype, upload_timestamp.

5. Analytics & Dashboard enhancements:
 - Add metrics specific to PS-2:
   - Number of WhatsApp-submitted complaints (total, per day).
   - Distribution across 23 financial subtypes + social media types.
   - Number of uploaded evidence items and top upload types.

  - Pending follow-ups, account-unfreeze queued.
  - Status check request count.
- Charts: bar chart for top 10 fraud subtypes, time series of chatbot complaints, pie for social vs financial.
- Export CSV / Excel endpoints and UI that include all PS-2 fields and attachment metadata.

6. NLU basics:
 - Implement a small intent classifier for the chatbot intake to route users: {new_complaint, status_check, account_unfreeze, other_query}.
 - Intents examples: “I have been scammed”, “check status”, “account freezed”, “other query”.
 - For new_complaint, guide the user step-by-step to collect PS-2 fields (exact sequence described in PS-2).
 - Provide simple validation and friendly error messages.

7. Back-end & DB:
 - Use Flask (if backend is Flask) or existing backend. Provide SQLAlchemy models and migrations:
   - tables: complaints, complaint_attachments, fraud_types, agents, followups, status_checks, export_jobs.
 - Provide DB schema and migration script (Alembic).
 - Example complaint model fields included in schema.

8. Security & privacy:
 - Store PII encrypted at rest (column-level encryption). Provide implementation using environment key (e.g., AES encryption with key from env).
 - Authentication & Roles:
   - Roles: admin, agent, viewer.
   - Role-based access control for routes and UI actions (assign agent, view uploads).
 - Rate-limit the webhook endpoint.
 - Sanitize file uploads, limit size (e.g., 10MB), validate mime types.

9. Tests:
 - Unit tests for: reference number generation, validation rules, API endpoints.
 - Integration test: simulate WhatsApp webhook (POST) with typical payload and attachments and assert DB entry + file stored.
 - Basic UI tests (Playwright or Cypress skeleton) to assert the main flows.

10. DevOps & Deployment:
 - Dockerfile and docker-compose for local dev (backend, db, object-storage minio).
 - CI pipeline (GitHub Actions / Replit CI): run tests, run lint (flake8/black for python; eslint for frontend).
 - Provide a simple helm/manifest or Replit deploy instructions. Make the app run on Replit (or container registry) with env vars: DATABASE_URL, AWS_S3_ENDPOINT (optional), AES_KEY, JWT_SECRET.

ACCEPTANCE CRITERIA (must be met):
- Admin can see chatbot complaints table with the full PS-2 fields and open complaint details

including uploaded evidence.
- The webhook endpoint accepts a simulated WhatsApp payload and creates a complaint and returns reference number.
- Status check UI accepts ack number/phone and shows complaint summary.
- Fraud taxonomy (full 23 + social types) visible and filterable in UI.
- Export endpoint returns CSV with all PS-2 fields and attachment links.
- Basic NLU intent classification routes user correctly (simulated).
- DB migrations provided and tests pass in CI.

FILES & CHANGES:
- Backend: `app/models.py` (SQLAlchemy), `app/api/chatbot.py` (webhook + status check + uploads), `app/schemas.py` (pydantic / marshmallow), `migrations/`.
- Frontend: `src/pages/ChatbotComplaints.jsx`, `src/components/ComplaintDetail.jsx`, `src/pages/StatusCheck.jsx`, `src/pages/AccountUnfreeze.jsx`, update `src/routes.js`, update dashboard components for new analytics widgets.
- Add `docker-compose.yml` (postgres, minio), `Dockerfile`, `.github/workflows/ci.yml`.

EXAMPLE API CONTRACTS:
- POST `/api/webhook/whatsapp` (simulate)
- body: multipart/form-data or JSON with fields {intent_hint, name, guardian_name, dob, phone, email, gender, village, post_office, police_station, district, pin_code, scam_category, scam_subcategory, account_number, message_text}
- attachments: files[]
- response: {reference_number: "CS-2025-001", status: "created"}
- GET `/api/complaints?query=&page=&per_page=&status=&district=&fraud_category=`
- GET `/api/complaints/{ref}`

NLU:
- Small rule-based classifier + optional fastText/sentence-transformer fallback.
- Provide training file of 40 example utterances mapped to intents.

DELIVERABLES:
- Pull request that modifies repository with code, tests, migrations, and README with run & deploy instructions.
- Postman collection or curl examples demonstrating webhook and admin flows.
- Short security section in README describing encryption and access control.

IMPLEMENTATION NOTES:
- Keep UI consistent with existing tailwind theme and rounded cards.
- For file storage in Replit, default to local `/data/uploads` but read `S3_ENDPOINT` env var and use boto3 when present.
- Make all user-facing texts support i18n (strings in one file) – English default.

If you understand, create the branch `feature/ps2-whatsapp-chatbot` and open a PR template in the repo with the implementation checklist and testing instructions.