Animap – Wildlife Sightings SaaS (Resume Project)

A mobile-first platform to record and share wildlife sightings on an interactive map. Users log animals with GPS, species details, photos or short videos, and publish for the community. Others discover nearby sightings, filter by species or date, and engage via likes and comments. Moderation tools allow admins to handle reports and flagged content.

- Stack: Expo React Native (mobile), Spring Boot (backend), Postgres + PostGIS (geo), AWS S3 (media), AWS EKS (Kubernetes), Google Maps SDK.
- Auth: OIDC (Auth0) using PKCE on mobile; Spring validates JWT as a resource server.
- Deployment: Manual deploy on AWS (no CI/CD required for resume scope).

---

1. Product Overview

1.1 Personas

- Explorer (User): create sightings, upload media, browse map, filter, like/comment.
- Moderator/Admin: review reports, hide/unhide or delete content, resolve cases, ban users.

1.2 Core User Journeys

- Create Sighting: capture GPS → select species → add notes → add photos/video → publish.
- Discover: map with clustered markers → filter by species/date → open details → like/comment → report if needed.
- Moderate: review reports queue → inspect sighting/comment → act (hide/delete/ban) → resolve.

1.3 MVP Scope (Must-Have)

- OIDC PKCE login with Auth0; roles USER/MODERATOR/ADMIN.
- Sightings CRUD with publish/unpublish and visibility (public/unlisted/private).
- Media uploads via S3 presigned URLs (images JPEG/PNG; videos MP4 ≤ 60s, ≤ 100 MB).
- Map discovery: Google Maps clustering, nearby search (radius), filters (species/date).
- Likes and comments.
- Reporting/flagging + admin moderation dashboard (web SPA).
- Geo queries with PostGIS (nearby and viewport/bbox).

1.4 Nice-to-Have (Time Permitting)

- Offline drafts and background upload retries.
- Sensitive species location fuzzing (rounding/jitter).
- Thumbnails Lambda for images/videos; CloudFront CDN for media.
- Push notifications (likes/comments/moderation events).

---

2. Architecture

2.1 Mobile App (Expo React Native, TypeScript)

- Libraries: `react-native-maps` (Google), `expo-location`, `expo-image-picker`/`expo-camera`, `@tanstack/react-query`, `zod`, `react-navigation`, `react-native-app-auth` (PKCE), `expo-file-system`.
- Flows: Auth (browser-based), Map (clusters + callouts), Create/Edit Sighting (wizard), Sighting Detail, Profile, Settings.
- Uploads: direct S3 PUT via presigned URLs from backend; client-side image compression; progress UI.

2.2 Backend API (Spring Boot)

- Spring starters: Web, Security (OAuth2 Resource Server), Data JPA, Validation, Flyway, Actuator.
- Integrations: Postgres + PostGIS, AWS SDK (S3 presign), OpenAPI/Swagger.
- Security: JWT validation via JWKs; method-level `@PreAuthorize`; ownership checks.

2.3 Data

- AWS RDS Postgres 15 with PostGIS extension.
- S3 bucket `animap-media` for uploads; object keys per sighting.
- Optional CloudFront CDN for media delivery (stretch).

2.4 Deployment (Manual, Resume-Friendly)

- Dockerized backend → push to ECR.
- EKS cluster with ALB Ingress + ACM TLS.
- K8s manifests: Deployment, Service, Ingress, Secrets (DB, OIDC, S3), ConfigMap.
- Admin SPA built and uploaded to S3 (static site) or behind CloudFront.

---

3. Data Model (ER Summary)

- User(id, auth_sub, display_name, avatar_url, role[USER|MODERATOR|ADMIN], created_at)
- Species(id, common_name, scientific_name, category, iucn_status)
- Sighting(id, user_id, species_id, title, description, location geography(Point,4326), location_precision_m, taken_at, published, visibility[public|unlisted|private], like_count, comment_count, created_at, updated_at)
- Media(id, sighting_id, type[image|video], s3_key, url, width, height, duration_s, thumb_url, order_index)
- Comment(id, sighting_id, user_id, text, parent_comment_id, created_at)
- Like(sighting_id, user_id, created_at) UNIQUE(sighting_id,user_id)
- Report(id, entity_type[sighting|comment|user], entity_id, reporter_user_id, reason, notes, status[open|action_taken|dismissed], resolved_by, resolved_at, resolution_notes)

Indexes: GIST on Sighting.location; btree on foreign keys and (species_id, taken_at).

---

4. Key API Endpoints

- Auth: OIDC (no custom sign-in endpoints). Backend validates JWT.
- Users: `GET /me`, `PATCH /me`
- Species: `GET /species?query=...` (+ admin CRUD/import optional)
- Sightings:
  - `POST /sightings` (metadata create, returns id)
  - `POST /sightings/{id}/media/presign` (get presigned PUT for S3)
  - `POST /sightings/{id}/publish`
  - `GET /sightings/{id}`
  - `GET /sightings/nearby?lat&lng&radius_km&species_id&from&to&page`
  - `GET /sightings/explore?bbox&species_id&from&to` (viewport)
  - `PATCH /sightings/{id}` (owner), `DELETE /sightings/{id}` (owner/admin)
- Engagement: `POST/DELETE /sightings/{id}/likes`, `GET/POST /sightings/{id}/comments`
- Reports: `POST /reports`
- Admin: `GET /admin/reports?status=open`, `POST /admin/reports/{id}/resolve`, `POST /admin/sightings/{id}/{hide|unhide}`, `POST /admin/users/{id}/ban`

---

5. Geo Queries (PostGIS)

- Nearby (meters):

```
SELECT id, title,
       ST_Distance(location, ST_MakePoint(:lon, :lat)::geography) AS distance_m
FROM sighting
WHERE published = TRUE
  AND visibility = 'public'
  AND ST_DWithin(location, ST_MakePoint(:lon, :lat)::geography, :radius_m)
  AND (:species_id IS NULL OR species_id = :species_id)
  AND (:from IS NULL OR taken_at >= :from)
  AND (:to IS NULL OR taken_at <= :to)
ORDER BY distance_m ASC, taken_at DESC
LIMIT :limit OFFSET :offset;
```

- Viewport (bbox):

```
SELECT id, title
FROM sighting
WHERE published = TRUE
  AND visibility = 'public'
  AND ST_Intersects(
    location,
    ST_MakeEnvelope(:minLon,:minLat,:maxLon,:maxLat,4326)::geography
  );
```

---

6. Security & Privacy

- OIDC PKCE via Auth0 for mobile and admin SPA.
- Spring Security Resource Server with JWKs; role-based access control.
- Ownership checks for edit/delete; `@PreAuthorize` on admin endpoints.
- Input validation (Bean Validation); MIME whitelist; file size caps.
- Rate limiting (user + IP) for write endpoints.
- Transport security with HTTPS (ALB + ACM). Secrets stored in K8s Secrets.

---

7. Auth0 Setup (Assumed Provider)

- Domain (Issuer): `https://YOUR_TENANT.auth0.com/`
- API (Identifier/Audience): `animap-api`
- Mobile App (Native):
  - Allowed Callback: `com.animap.app://oauthredirect`
  - Allowed Logout: `com.animap.app://oauthredirect`
  - Allowed Origins: `com.animap.app://*`
  - Grant Types: Authorization Code (PKCE)
- Admin SPA (Single Page App):
  - Callback URL: `https://admin.animap.example.com/callback`
  - Logout URL: `https://admin.animap.example.com/`
  - Allowed Web Origins: `https://admin.animap.example.com`
- Scopes: `openid profile email offline_access` (if refresh tokens are desired)
- Backend config envs:
  - `OIDC_ISSUER_URI=https://YOUR_TENANT.auth0.com/`
  - `OIDC_AUDIENCE=animap-api`

---

8. Media Pipeline

- Presigned PUT from backend with constraints:
  - Images: JPEG/PNG ≤ 10 MB
  - Videos: MP4 ≤ 60s, ≤ 100 MB
- Client uploads directly to S3, then confirms on backend (stores metadata).
- Optional: S3 Event → Lambda to generate thumbnails; store `thumb_url`.
- Delivery: serve S3 URLs directly; add CloudFront later if needed.

---

9. Israel Seed Data Plan

- Regions: Tel Aviv coast, Haifa/Carmel, Galilee/Sea of Galilee, Golan Heights, Jerusalem Hills, Negev/Beer Sheva, Ramon Crater, Arava/Eilat, Hula Valley wetlands.
- Species examples: Palestine sunbird, hoopoe, rock hyrax, Nubian ibex, griffon vulture, loggerhead sea turtle.
- Volume: ~2,000 synthetic sightings over the last 12 months with realistic spatial clustering.
- Include: some flagged reports, a few unlisted/private items for visibility demos.

---

10. Manual Deployment Checklist (AWS)

- Backend: build Docker image → push to ECR → deploy to EKS (Deployment/Service/Ingress).
- RDS Postgres 15: enable PostGIS; apply Flyway migrations.
- S3: create `animap-media` bucket; configure CORS for presigned uploads.
- Ingress: AWS ALB + ACM cert; HTTPS; set CORS on backend for mobile/admin origins.
- Admin SPA: build → upload to S3 (static site) or behind CloudFront.

- Mobile: set `EXPO_PUBLIC_API_BASE_URL`, `EXPO_PUBLIC_OIDC_*`, and Google Maps API key.

One-Month Execution Timeline (4 Weeks)

Week 1 – Foundations & Auth

- Repos + project scaffolding (mobile, backend, admin). Local PostGIS via Docker. Flyway schema.
- Expo app shell: navigation, Google Maps keys, map screen placeholder.
- Auth: Auth0 setup; mobile PKCE with `react-native-app-auth`; backend JWT validation; `/me` endpoint.
- Species model + search API; seed core species for Israel.
- Sighting create flow: metadata + presign endpoint; S3 bucket; image upload E2E; detail view scaffold.

Milestone: Login and create a sighting with a photo locally.

Week 2 – Geo, Discover, Engagement

- PostGIS nearby query (`ST_DWithin`) + pagination; bbox query for viewport.
- Map clustering (`supercluster`); filters (species/date) + radius control; list view.
- Sighting detail polish; likes (optimistic) and counts; comments thread (single-level).
- Video upload path (MP4 ≤ 60s) via presign; playback; thumbnail placeholder.
- Publish/unpublish and visibility (public/unlisted/private); owner edit/delete.

Milestone: Map/list browsing with filters; like/comment; video upload working.

Week 3 – Moderation & Hardening

- Reports model + `POST /reports` from mobile detail screen.
- Admin endpoints: review queue, resolve actions, hide/unhide, delete, (optional ban).
- Admin SPA (Vite + MUI): login, reports queue, sighting moderation panel.
- Security hardening: validation, MIME caps, basic rate limiting. JSON logs & simple metrics.
- Israel seed generator: ~2,000 sightings; load into DB; verify performance.

Milestone: Admin can moderate flagged content; seeded Israel data visible.

Week 4 – Deploy, Polish, Showcase

- Dockerize backend; push to ECR; create EKS cluster; apply K8s manifests; set up ALB + ACM.
- Provision RDS Postgres + PostGIS, S3 bucket; configure backend envs and CORS.
- Admin SPA deploy to S3; wire to API; test end-to-end on AWS.
- Mobile polish: upload progress/retry, empty/error states, image compression, accessibility.
- Documentation: README, screenshots, architecture diagram, 60–90s demo video.

Milestone: Public demo live on AWS with polished UX and clear docs.

What to Trim if Behind (Priority Order)

1. Video uploads → ship images only. 2) Comment threading → single-level only. 3) User ban → keep hide/unhide + delete. 4) Unlisted/private → public only. 5) Rate limiting → post-MVP.