You're thinking like a pro now, Joyce 👏 — **security** is *critical* when dealing with something as sensitive as abuse reports.

Let's break it down layer by layer: **Database** → **APIs** → **Admins** → **Files** → **User privacy**.

🔐 1. Database Security

☑ What you must do:

- **No Personal Data unless needed.**
  - Only store contact info *if victim consents*.
- **Use Django's built-in ORM.**
  - Avoid raw SQL to prevent SQL injection.
- **Encrypt sensitive fields (like phone numbers).**
  - Use Django packages like `django-cryptography`.

🔒 Good Practices:

Risk	Solution
Hackers accessing your DB	Use strong DB passwords & limit DB access to your app only
Abuse of stored data	Never expose the raw DB in your frontend
Data theft	Regularly back up and use encrypted storage (e.g., `AES` encryption for files)

🌐 2. API Security

☑ What you must do:

- **Authentication:**
  - Use Django REST Framework (DRF) with **token-based or JWT authentication** for admins.
- **Authorization:**
  - Only allow users with the correct **roles** to access or respond to reports.
- **Rate Limiting:**
  - Limit how often someone can hit your APIs (prevent bots).
- **Data Validation:**
  - Don't trust any incoming data — always validate abuse type, age, etc.
- **CORS Protection:**
  - Only allow requests from your frontend site (e.g., `https://sautiyetu.co.ke`).

👩🏾‍💻 3. Admin Security

✅ Must-Have Features:

- **Unique accounts for each admin.**
- **Role-based access control.**
- **Two-Factor Authentication (2FA)** for superadmins.
- **Logs of every admin action** stored securely.
- Disable login for admins who leave the team or are unverified.

Admin Level	Access
SuperAdmin	Can see logs, manage others
Counselor	Can view/respond to reports
Analyst	View data only, no access to identities

📁 4. File Security (Voice Recordings)

✅ How to protect them:

- **Upload to a secure folder** outside public access.
- **Use random names** for files — never `voice_from_joyce.mp3`.
- **Encrypt on save** and decrypt only when played by an authorized admin.
- **Do not expose file URLs directly** — use temporary access links that expire.

👤 5. User Privacy & Safety

Your users (especially victims) need to feel **completely safe**.

✅ Measures you must take:

- **Never require login to report abuse.**
- **Quick Exit button** that takes them to Google.
- **No cookies / no tracking pixels.**
- **Swahili option** ensures they understand every instruction.
- **All data optional unless consented.**
- **Contact info hidden unless victim agrees to be helped.**

🔐 6. Deployment/Hosting Security

- Use **HTTPS (SSL)** only — no HTTP.
- Keep Django's `DEBUG = False` in production.
- **Set strong `ALLOWED_HOSTS`** in your settings file.
- Regularly update Django and all packages.
- Use a **firewall** on your server or host on a platform with built-in protection.

🔁 Summary – Your Full Security Stack

Area	What to Secure	Tools / Practices
Database	No personal data by default, encrypted fields	ORM, AES encryption
API	Auth, role-checks, rate limiting	DRF, JWT, CORS
Admins	Separate roles, 2FA, logging	Django Users + Permissions
Voice Files	Encrypted uploads, access control	Random names, protected folders
User Privacy	Anonymity, safety tools, no cookies	Consent-first design
Deployment	SSL, debug off, firewalls	Cloud host, strong configs

Would you like me to:

- Help you set up one of these in your Django project?
- Or give you a security checklist as a PDF for your planning?

You're creating a safe space — and it starts by *making sure it's actually safe*. 🔐✨

Ask ChatGPT