Internal Memo: UID Security and Token Verification (Draft)

Status: Internal use only until verification logic is complete and tested

This document captures the rationale, risks, and implementation plan for securing `firebase_uid` references in D1 and protecting the `/api/sync-user` and related endpoints from misuse or impersonation.

Vulnerability Overview

Scenario: The project is open source. A bad actor (or careless contributor) could remove the Firebase ID token validation check from a Worker route (e.g. `/api/sync-user`) and push insecure code to production. This would allow an attacker to send requests with someone else's UID and modify their data.

Threat type: Insider bypass / supply chain manipulation

Assets at risk:
- Integrity of user posts
- Trust in platform security
- Reputation of project

Mitigation Strategy

1. Centralize Token Verification

Create a reusable helper function:

```
export async function requireVerifiedUID(request, env) {
  const idToken = getAuthHeader(request); // Authorization: Bearer <token>
```

```
  if (!idToken) return new Response("Missing token", { status: 401 });

  const decoded = await verifyFirebaseToken(idToken);
  if (!decoded?.uid) return new Response("Invalid token", { status: 403 });

  return decoded.uid;
}
```

Usage:

```
const verifiedUID = await requireVerifiedUID(request, env);
if (verifiedUID instanceof Response) return verifiedUID;

if (verifiedUID !== body.uid) return new Response("UID mismatch", { status: 403 });
```

2. Restrict Production Deploys

- Use GitHub Actions to only allow `wrangler deploy` from `main` branch
- Optional: lock prod deploy behind Cloudflare Access or secrets rotation

3. Log All Sensitive Writes

Create an `audit_log` table in D1 to store:
- Timestamp
- Verified UID
- Endpoint called
- Operation type (insert/update/delete)

4. Add CI Verification (optional, future)

- Lint for missing `requireVerifiedUID()` in any route file
- Alert if `uid` values appear in route logic without verification

---

Why We Store `firebase_uid` in D1

The `uid` is not secret. It's used like a username:

- It's globally unique
- It maps to the user's Firestore and Firebase record

BUT -- we must never trust a UID in a request unless it's verified via token.

Summary Rules

Rule | Why

-----|-----

UID must be paired with verified token | Prevent impersonation

Only use decoded UID for writes | Enforces user ownership

Never accept raw UID from body alone | Avoid forgery

Deploy to prod only from audited source | Block insecure code

---

Next Steps

- [ ] Implement `auth/requireVerifiedUID.js`
- [ ] Integrate into `/api/sync-user` and other sensitive routes
- [ ] Test against invalid/missing tokens
- [ ] Set up audit logging (optional)
- [ ] Consider commit hook or CI gate for security validation