VIVO

Version 0.1 August 2026 virtualintegrationviaopen.com

Virtual Integration Via Open — a social platform built on declared intent.

ABSTRACT

VIVO is a social platform that replaces inferred, algorithmically ranked feeds with structure the user declares directly. It rests on three primitives: a social graph the user authors and orders by hand, presentation state stored as portable first-class data, and an edge-native architecture with no behavioral tracking layer. This paper describes the data model, the theming system, the request path, and the current implementation status — separating what ships today from what is specified but not yet wired.

1 The feed replaced the page

Between roughly 2003 and 2008, a social profile was a document its owner controlled. Presentation was user-authored — people wrote CSS into their own pages. Relationships were declared: you chose your Top 8 and everyone could see the ordering. The page was a place, and its structure was an act of authorship.

What replaced it is a ranking function: a row scored by a model the user cannot read, over a graph inferred from engagement rather than stated by intent. Three capabilities were lost, and VIVO is organized around recovering them:

- **Authored presentation.** Users could shape how their space looked. Customization is now reduced to an avatar and a display name.
- **Declared relationships.** Ordering was explicit, public, and chosen. It is now inferred from dwell time and click behavior, and kept private because it is a business asset.
- **Portable identity.** A handle, a graph, and a body of posts were once yours to move. They are now a tenancy that ends when the platform decides it does.

2 Design principles

Each principle maps to a concrete mechanism in the schema or the client. Sections 3 through 6 give the implementation for each.

Privacy

No behavioral profile is constructed, because nothing in the product monetizes one. There is no ad auction and therefore no reason to model the user.

Freedom

Declared beats inferred. Ordering that affects what a user sees is a value the user wrote, not a model output. A stored integer can be read, edited, and exported.

Customization

Presentation is data, not a purchase. A theme is a JSON document bound to the user row — inspectable, portable, and forkable.

3 Architecture

VIVO is edge-native and has no origin server. The application shell is a static Progressive Web App served from Cloudflare Pages; reads and writes are handled by Pages Functions executing at the edge; state lives in Cloudflare D1, a SQLite database replicated to the edge. There is no long-running backend process to attack, scale, or subpoena.

```
// request path
Client (PWA shell, service worker)
  → Cloudflare Pages           static shell, cached, offline-capable
  → Pages Functions /api/*     validation, auth, rate limiting
  → D1 (SQLite at edge)        durable state
```

The client is a hash-routed single-page application with five views, transitioned with the View Transitions API. The service worker caches the shell, so the app opens and navigates without a network round trip. Because the data layer is a small set of JSON-returning functions, the same client can be pointed at a self-hosted D1 or any backend implementing the same endpoints.

4 The declared social graph

The central data structure is a friendship edge carrying an explicit, user-assigned rank.

```
CREATE TABLE friendships (
  user_id   INTEGER NOT NULL REFERENCES users(id),
  friend_id INTEGER NOT NULL REFERENCES users(id),
  rank      INTEGER,        -- 1..8 = Top 8; NULL = friend, unranked
  label     TEXT,           -- "forever", "band", ...
  PRIMARY KEY (user_id, friend_id)
);
CREATE INDEX idx_friend_rank ON friendships(user_id, rank);
```

The consequences of storing rank rather than computing it are the argument of this paper:

- **Ordering is a value, not a model output.** There is no ranking service, no feature store, and no training data. Retrieving a user's Top 8 is an indexed read over a single table.
- **It is public by design.** The ordering is visible to everyone, which is what made it socially meaningful. A private inferred ranking serves the platform; a public declared one serves the user.
- **Scarcity carries the signal.** Eight slots mean adding someone costs something. Ranking is expressive precisely because it is constrained.
- **Engagement never feeds it.** No dwell time, click, or scroll event writes to this table. The only writer is the person whose graph it is.

Unranked friendships are ordinary edges with `rank IS NULL`, so the Top 8 is a view over the graph rather than a separate structure. Reordering updates at most eight rows.

5 The vibe engine

Customization is a small, closed set of design tokens the client resolves at runtime. The entire visual identity of the application is driven by seven CSS custom properties:

```
const VIBE_VARS = ["--bg", "--panel", "--panel2", "--border",
                   "--text", "--muted", "--grad"];
```

A theme — a "vibe" — is a JSON document binding values to those tokens, persisted per user in one row:

```
CREATE TABLE vibes (
  user_id    INTEGER PRIMARY KEY REFERENCES users(id),
  vibe_json  TEXT NOT NULL,   -- {type:"preset"|"custom", name, vars}
  updated_at TEXT DEFAULT (datetime('now'))
);
```

Because the token set is fixed and small, a theme is safe to accept from untrusted input: values apply to an allowlist of custom properties on the document root and cannot introduce selectors, scripts, or layout changes. This is what makes user-authored presentation tractable where arbitrary CSS injection was not.

Three authoring paths exist today: named presets, a random palette generator, and compilation from a natural-language prompt. All three produce the same artifact — a token map — so a prompt-generated theme is indistinguishable from a hand-authored one, and portable by construction: serializable, diffable, and applicable to any client honoring the same token contract.

6 Data model

The schema is deliberately small and relational. Every table below exists in the deployed D1 schema.

TABLE	PURPOSE
`users`	Identity, handle (unique), mood, profile song, presence, counters
`friendships`	Ranked social graph — the Top 8 and unranked edges
`vibes`	Per-user presentation state as a token document
`comments`	Profile wall, addressed to a profile rather than a feed
`tracks, playlists, playlist_tracks, liked_tracks`	Music graph; profile songs bind a track to a user
`conversations, messages`	Direct messaging, one conversation per peer pair
`reels`	Short-form and live entries
`waitlist, waitlist_top8`	Pre-launch signups and reserved handles

7 Privacy posture

VIVO does not run an advertising auction. That single decision removes the economic reason to build a behavioral profile, and most privacy properties follow from it rather than from policy language:

- **No engagement telemetry writes to user state.** Nothing a user views, scrolls past, or lingers on is recorded against their row. The tables in §6 hold only content and declared structure.
- **No cross-site identity.** No pixel, no third-party ad SDK, no identity resolution against an external graph.
- **Aggregate analytics only.** Screen-level page views and funnel counters; no per-user event stream.
- **Handles are reserved atomically.** Registration performs a transactional insert against a `UNIQUE` constraint and returns `409` on collision, so a handle cannot be double-allocated under concurrency.

Portability follows: a user's complete record is a bounded set of rows across the tables in §6, exportable as JSON without reconstructing derived state — because there is none to reconstruct.

8 Implementation status

Stated precisely, because a whitepaper that blurs shipped work into planned work is not useful to a developer evaluating it.

COMPONENT	STATUS	NOTES
Application shell, 5 views, routing	SHIPPED	PWA, service worker, offline shell, View Transitions
Vibe engine (presets, random, prompt)	SHIPPED	Client-side; persists locally
Waitlist + handle reservation API	SHIPPED	Live against D1; atomic, with rollback
D1 schema (all tables in §6)	DEPLOYED	Created and seeded; app reads not yet cut over
Application reads from D1	IN PROGRESS	Currently served from static JSON shaped identically to the D1 queries
Accounts and authenticated profiles	PLANNED	Prerequisite for server-side vibe persistence
Public theming API	PLANNED	Read/write vibes as portable documents
Data export	PLANNED	Full per-user JSON export

Static JSON seeds match the D1 query results exactly, so cutting the read path over is a transport change rather than a client rewrite. Target debut is **Q4 2026**.

9 Open questions

Stated openly, since they materially affect anyone building against VIVO:

- **Federation.** Whether the identity and graph layer is exposed for third-party implementation, and under which protocol. The schema does not preclude it.
- **Moderation at scale.** A declared graph reduces amplification but does not itself solve abuse. Policy and tooling are unspecified.
- **Theme distribution.** Whether vibes are shared peer-to-peer, through a public index, or both.
- **Licensing.** Which components are released as open source, and under what terms.

About the name

VIVO is an acronym for **V**irtual **I**ntegration **V**ia **O**pen. Both marks were filed with the United States Patent and Trademark Office in May 2014 by David Otero — VIVO under Serial No. 86279544 and VIRTUAL INTEGRATION VIA OPEN under Serial No. 86280483, each in Class 42 (Computer & Software Services). Both applications lapsed in 2016; current use is unregistered.