<https://gemini.google.com/u/1/app/7a9daee28ed15dc2>

<https://gemini.google.com/u/1/app/8f9ebd64aa0fd57f>

Okay, I will update the "6. User Interaction Module" overview document in full, incorporating all the refinements and decisions we've made. This updated overview will provide a clear and current picture of the module.

\* \* \* \* \*

User Interaction Module Overview

--------------------------------

Version: 2.0 (Incorporating V2.1 Table Specifications and API Conceptualization)

Date: May 18, 2025

1\. Executive Summary

This database module enables core user engagement by allowing pilgrims to cast votes on waypoints and share textual tips. It supports community sentiment analysis, content moderation for tips, and robust, multilingual category management for enhanced user experience. Key functionalities include capturing user votes which directly update waypoint popularity, managing user-submitted tips through a defined moderation workflow, and categorizing tips using a centralized, translatable system. These features directly fuel platform capabilities like sorting content by popularity and providing users with recent, practical, and categorized insights.

2\. Group-Level Snapshot

| Group | Key Tables | Primary Purpose | Top Inter-Group Links |

| 6\. User Interaction Module | `user\_waypoint\_votes`, `user\_waypoint\_short\_tips`, `tip\_categories\_master` | Capturing user votes on waypoints, managing user-submitted tips with moderation, and defining standardized, translatable categories for tips. | `profiles` (for user identity/authorship), `waypoints` (for content being interacted with), `languages\_master` (for tip language), `translations` (for category i18n). |

3\. Narrative Walkthrough

This module, "6. User Interaction Module," focuses on capturing direct user feedback and contributions related to waypoints.

- `tip\_categories\_master` (v2.1):

- Role: Defines a standardized, admin-managed list of categories for user tips. This supports multilingual display of category names and consistent tip organization.

- 🔑 This table is referenced by `user\_waypoint\_short\_tips` via `tip\_category\_code`.

- Critical:

- `category\_code`: Primary key (e.g., 'practical\_advice').

- `default\_name`, `default\_description`: Store the category name and description in the primary reference language (e.g., English). These fields are translatable via `public.translations`.

- `icon\_identifier`: Optional string for a UI icon.

- `is\_active`: Boolean to enable/disable categories.

- `created\_by\_profile\_id`, `updated\_by\_profile\_id`: Standard audit columns linking to `profiles`.

- `created\_at`, `updated\_at`: Timestamps for category definition, with `updated\_at` managed by the `public.handle\_updated\_at` trigger.

- An `AFTER DELETE` trigger (`trigger\_cleanup\_translations\_on\_tip\_category\_delete`) ensures orphaned entries in `public.translations` are removed.

- `user\_waypoint\_votes` (v2.1.1):

- Role: Enables authenticated users (pilgrims) to cast "thumbs up" or "thumbs down" votes on waypoints.

- 🔑 This table acts as a many-to-many junction table between `profiles` and `waypoints` through its composite primary key `(profile\_id, waypoint\_id)`. Each user can vote once per waypoint.

- Critical:

- `vote\_type`: An ENUM (`'up'`, `'down'`) indicating the vote.

- `created\_at`, `updated\_at`: Timestamps for vote creation and modification, with `updated\_at` managed by the `public.handle\_updated\_at` trigger.

- `deleted\_at`: For soft deletion (vote retraction).

- `update\_waypoint\_vote\_counts` trigger: This is a crucial `AFTER INSERT/UPDATE/DELETE` trigger that denormalizes `up\_vote\_count` and `down\_vote\_count` onto the `public.waypoints` table for performance. This requires `up\_vote\_count` and `down\_vote\_count` columns to exist on `public.waypoints`.

- `user\_waypoint\_short\_tips` (v2.1.1):

- Role: Stores brief, user-submitted textual tips about waypoints, subject to a moderation workflow.

- 🔑 This table has many-to-one relationships with `profiles` (tip author), `waypoints` (tip subject), `languages\_master` (tip language), and `tip\_categories\_master` (tip category).

- Critical:

- `tip\_text`: The content, with a 500-character limit enforced by a `CHECK` constraint.

- `language\_code`: Stores the language of the tip, FK to `languages\_master`.

- `tip\_category\_code`: Foreign key to `tip\_categories\_master.category\_code` (`ON DELETE SET NULL`).

- `moderation\_status`: An ENUM (`'pending\_approval'`, `'approved\_visible'`, etc.) defaulting to `'pending\_approval'`.

- `is\_publicly\_visible`: A `STORED` generated column (PostgreSQL 12+) derived from `moderation\_status = 'approved\_visible'::public.content\_moderation\_status\_enum AND deleted\_at IS NULL`.

- `moderated\_by\_profile\_id`, `moderation\_timestamp`, `moderation\_notes\_internal`: For moderation audit trail.

- `profile\_id` serves as the creator ID.

- `created\_at`, `updated\_at`, `deleted\_at`: Standard audit and soft-delete columns, with `updated\_at` managed by the `public.handle\_updated\_at` trigger.

4\. Cross-Cutting Concerns

- Users & Roles:

- `profile\_id` from the `profiles` table is consistently used to link votes and tips to their authors.

- Ownership is enforced via RLS, allowing users to manage their own contributions (e.g., insert, update under conditions, view their own votes/tips).

- Moderation roles are critical for `user\_waypoint\_short\_tips`, with fields like `moderated\_by\_profile\_id` and specific RLS policies for moderators. The RLS policies rely on a function like `public.check\_if\_user\_is\_moderator(UUID)` for identifying users with moderation privileges.

- Translations / i18n:

- `tip\_categories\_master` now stores `default\_name` and `default\_description` in the primary reference language. These are translatable via `public.translations`. An `AFTER DELETE` trigger on `tip\_categories\_master` handles orphaned translation cleanup.

- The `public.view\_tip\_categories\_localized` view is available to provide category names and descriptions in the user's current language.

- `user\_waypoint\_short\_tips.tip\_text` is stored in the language it was submitted in, indicated by `language\_code` (FK to `languages\_master`). This allows for UI-level filtering or future UI-driven translation features.

- `user\_waypoint\_votes.vote\_type` ('up', 'down') are internal system values; UI representation is handled by the frontend.

- ENUM & Taxonomy Registry:

- `public.vote\_type\_enum`: Used in `user\_waypoint\_votes.vote\_type`. Defined as `('up', 'down')`.

- `public.content\_moderation\_status\_enum`: Used in `user\_waypoint\_short\_tips.moderation\_status`. Defined as `('pending\_approval', 'approved\_visible', 'rejected\_hidden', 'flagged\_for\_review\_by\_admin', 'archived\_by\_admin')`.

- Tip categories are managed by the `public.tip\_categories\_master` lookup table.

- Media & Files:

- `tip\_categories\_master.icon\_identifier` can store a string to link categories to UI icons. No direct file storage is managed by these tables.

- Audit / Soft-Delete / Versioning:

- All three core tables include `created\_at` (default `now()`) and `updated\_at` (default `now()`, auto-updated by `public.handle\_updated\_at` trigger).

- `tip\_categories\_master` now includes `created\_by\_profile\_id` and `updated\_by\_profile\_id`.

- For `user\_waypoint\_votes`, `profile\_id` serves as the creator/updater identifier due to RLS.

- For `user\_waypoint\_short\_tips`, `profile\_id` is the author, and `moderated\_by\_profile\_id` tracks moderation updates.

- Soft Deletion: `user\_waypoint\_votes` and `user\_waypoint\_short\_tips` implement soft deletion using a nullable `deleted\_at` timestamp. `tip\_categories\_master` uses `is\_active`.

- Versioning: Detailed history of changes (e.g., previous `tip\_text` values) is considered V2+ and not implemented.

5\. Security & Access Control 🔐

- RLS Overview:

- Row Level Security is enabled for `user\_waypoint\_votes`, `user\_waypoint\_short\_tips`, and `tip\_categories\_master`.

- `tip\_categories\_master`: RLS allows admins to manage categories and public read access to active categories.

- `user\_waypoint\_votes`: Policies allow authenticated users to insert, update (change vote type or retract via `deleted\_at`), and view their own votes.

- `user\_waypoint\_short\_tips`: Policies allow authenticated users to insert their own tips (defaulting to 'pending\_approval'), view all publicly visible tips (via `is\_publicly\_visible`), view their own tips regardless of status, and update their own tips under specific conditions (e.g., retraction, or editing if still pending approval). Moderator policies (relying on `public.check\_if\_user\_is\_moderator(UUID)`) grant broader access for viewing and updating tips for moderation.

- Dedicated SECURITY DEFINER Functions:

- `public.cleanup\_tip\_categories\_master\_translations()`: For cleaning orphaned translations.

- `public.update\_waypoint\_vote\_counts()`: Modifies `public.waypoints` and may require `SECURITY DEFINER` if RLS on `waypoints` restricts direct updates by users.

- The conceptual `public.check\_if\_user\_is\_moderator(UUID)` function, used for RLS on `user\_waypoint\_short\_tips`, would likely need to be `SECURITY DEFINER` if it queries privileged role information.

- Notes on `anon` vs `authenticated` access:

- `user\_waypoint\_votes`: Anonymous users (`anon` role) generally do not have rights. Authenticated users interact with their own votes.

- `user\_waypoint\_short\_tips`: Anonymous users generally do not have write access. Read access for `anon` role is typically limited to tips where `is\_publicly\_visible = true`, often managed at the API gateway or by specific RLS if direct DB access is granted to `anon`.

- `tip\_categories\_master`: Anonymous users can read active categories via the RLS policy.

6\. Prerequisite Objects & Build Order ⚙️

(Order assumes auth.users, public.profiles, public.waypoints, public.languages\_master, public.translations tables, and extensions.moddatetime or public.handle\_updated\_at function exist).

1. Global Helper Functions (SQL/PLPGSQL)

- `public.handle\_updated\_at()`: Updates `updated\_at` timestamp.

- `public.check\_if\_user\_is\_moderator(UUID)`: (Needs definition) Checks if a user has moderator privileges.

2. Types & ENUMs

- `public.vote\_type\_enum AS ENUM ('up', 'down');`

- `public.content\_moderation\_status\_enum AS ENUM ('pending\_approval', ...);`

3. `public.waypoints` Table Modifications

- Ensure `up\_vote\_count INTEGER NOT NULL DEFAULT 0` and `down\_vote\_count INTEGER NOT NULL DEFAULT 0` columns exist.

4. Core Table: `public.tip\_categories\_master` (v2.1)

- Create table with `default\_name`, `default\_description`, audit columns.

- DDL for `public.cleanup\_tip\_categories\_master\_translations()` trigger function.

5. Core Table: `public.user\_waypoint\_votes` (v2.1.1)

- Create table.

- DDL for `public.update\_waypoint\_vote\_counts()` trigger function.

6. Core Table: `public.user\_waypoint\_short\_tips` (v2.1.1)

- Create table with generated column `is\_publicly\_visible`.

7. Views

- `public.view\_tip\_categories\_localized`: Provides localized category names.

8. Indexes & Constraints

- Apply all defined PKs, FKs, CHECK constraints, and specific indexes for each table.

9. Triggers (Application of DML Triggers)

- Apply `handle\_updated\_at` triggers to all three tables.

- Apply `after\_user\_waypoint\_votes\_change` trigger to `user\_waypoint\_votes`.

- Apply `trigger\_cleanup\_translations\_on\_tip\_category\_delete` to `tip\_categories\_master`.

10. RLS Policies

- Enable RLS and apply all defined policies for each table.

11. Seed Data

- Populate `tip\_categories\_master` with initial categories and their default names/descriptions. Ensure corresponding translations are planned/added to `public.translations`.

7\. Performance & Optimization Extras

- Key Indexes & Why:

- `user\_waypoint\_votes`: Composite PK `(profile\_id, waypoint\_id)` for uniqueness and lookups; `idx\_user\_waypoint\_votes\_active` for querying active votes.

- `user\_waypoint\_short\_tips`: `idx\_user\_waypoint\_short\_tips\_visibility` for efficient display of public tips; `idx\_user\_waypoint\_short\_tips\_moderation` for moderation queues; indexes on `profile\_id`, `language\_code`, `tip\_category\_code`.

- `tip\_categories\_master`: PK on `category\_code`; `ix\_tip\_categories\_master\_active\_order` for UI lists.

- Denormalization:

- `up\_vote\_count` and `down\_vote\_count` on `public.waypoints`, updated by trigger from `user\_waypoint\_votes`, are critical for UI performance, avoiding costly aggregations.

- Generated Column:

- `user\_waypoint\_short\_tips.is\_publicly\_visible` (`STORED`) simplifies queries for visible tips and improves read performance.

- Views:

- `public.view\_tip\_categories\_localized` optimizes fetching and localizing tip category data for UI elements.

8\. Visuals (Conceptual ERD - Key Tables & Links)

Code snippet

```

erDiagram

profiles {

uuid id PK

# ... other fields

}

waypoints {

integer id PK # CRITICAL: Confirm type

string name

integer up\_vote\_count "DENORMALIZED"

integer down\_vote\_count "DENORMALIZED"

# ... other fields

}

languages\_master {

text language\_code PK

# ... other fields

}

tip\_categories\_master {

text category\_code PK

text default\_name "(Translatable)"

text default\_description "(Translatable, Nullable)"

text icon\_identifier "(Nullable)"

boolean is\_active

smallint sort\_order

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

timestamptz created\_at

timestamptz updated\_at

}

translations {

bigint id PK # Assuming from main translations spec

text table\_identifier

text column\_identifier

text row\_foreign\_key

text language\_code FK

text translated\_text

# ... other fields

}

user\_waypoint\_votes {

uuid profile\_id PK, FK

integer waypoint\_id PK, FK # CRITICAL: Confirm type

vote\_type\_enum vote\_type

text vote\_source\_platform "(Nullable)"

timestamptz created\_at

timestamptz updated\_at

timestamptz deleted\_at "(Nullable)"

}

user\_waypoint\_short\_tips {

bigint id PK

uuid profile\_id FK

integer waypoint\_id FK # CRITICAL: Confirm type

text tip\_text "CHECK len<=500"

text language\_code FK

text tip\_category\_code FK "(Nullable)"

text tip\_source\_platform "(Nullable)"

content\_moderation\_status\_enum moderation\_status

boolean is\_publicly\_visible "GENERATED"

boolean is\_pinned\_by\_admin

uuid moderated\_by\_profile\_id FK "(Nullable)"

timestamptz moderation\_timestamp "(Nullable)"

text moderation\_notes\_internal "(Nullable)"

timestamptz created\_at

timestamptz updated\_at

timestamptz deleted\_at "(Nullable)"

}

profiles ||--o{ user\_waypoint\_votes : "casts\_vote"

waypoints ||--o{ user\_waypoint\_votes : "receives\_vote\_for"

profiles ||--o{ user\_waypoint\_short\_tips : "submits\_tip"

waypoints ||--o{ user\_waypoint\_short\_tips : "has\_tip\_about"

languages\_master ||--|| user\_waypoint\_short\_tips : "tip\_language\_is"

tip\_categories\_master ||--o{ user\_waypoint\_short\_tips : "categorized\_by"

profiles ||--o{ user\_waypoint\_short\_tips : "tip\_moderated\_by"

profiles ||--o| tip\_categories\_master : "category\_created\_by"

profiles ||--o| tip\_categories\_master : "category\_updated\_by"

tip\_categories\_master ..> translations : "default\_name translated\_in"

tip\_categories\_master ..> translations : "default\_description translated\_in"

```

9\. Data & Workflow Flowchart

- A. User Waypoint Vote Lifecycle:

1. User Action (Frontend): Pilgrim clicks "up-vote" or "down-vote" on a waypoint.

2. Application Backend: Receives request (`profile\_id` from auth, `waypoint\_id`, `vote\_type`). Performs an UPSERT on `user\_waypoint\_votes`.

- No existing vote: `INSERT` new record (`deleted\_at = NULL`).

- Existing vote (same `profile\_id`, `waypoint\_id`):

- Different `vote\_type` & `deleted\_at IS NULL`: `UPDATE vote\_type`.

- Same `vote\_type` & `deleted\_at IS NULL` (re-click same): `UPDATE deleted\_at = NOW()` (Retract).

- `deleted\_at IS NOT NULL` (re-vote after retraction): `UPDATE vote\_type`, `deleted\_at = NULL`.

3. Database (`user\_waypoint\_votes`): `BEFORE UPDATE` trigger updates `updated\_at`. `AFTER INSERT/UPDATE/DELETE` trigger `after\_user\_waypoint\_votes\_change` fires.

4. Database (`waypoints` via trigger): `update\_waypoint\_vote\_counts` function adjusts `up\_vote\_count` and `down\_vote\_count`.

5. End-User Consumption: Waypoints display aggregated counts. UI reflects user's own vote status.

- B. User Waypoint Short Tip Lifecycle:

1. User Action (Frontend): Pilgrim submits a tip (`tip\_text`, `tip\_category\_code` (optional), `language\_code`).

2. Application Backend: Validates input. `INSERT` into `user\_waypoint\_short\_tips`. `moderation\_status` defaults to `'pending\_approval'`.

3. Database (`user\_waypoint\_short\_tips`): `is\_publicly\_visible` is computed.

4. Moderation Workflow (Admin Interface): Moderators view tips (e.g., `moderation\_status = 'pending\_approval'`). Can change `moderation\_status`, set `is\_pinned\_by\_admin`, add `moderation\_notes\_internal`. `moderated\_by\_profile\_id` and `moderation\_timestamp` are recorded. `updated\_at` is updated.

5. User Action (Update/Retract Tip): Pilgrim views their own tips. Can retract (`deleted\_at = NOW()`) or edit if `moderation\_status = 'pending\_approval'`. Edits to approved tips must revert status to `pending\_approval` (critical business rule).

6. End-User Consumption: Pilgrims view tips where `is\_publicly\_visible = true`. Tips can be filtered by `language\_code` or localized `tip\_category\_code`. Pinned tips are prominent.

10\. Critical Gaps & Risks

- 🔴 `waypoints` Table Modifications & Vote Count Trigger (P1): `up\_vote\_count` and `down\_vote\_count` columns \*must\* be added to `public.waypoints`. The `public.update\_waypoint\_vote\_counts()` trigger must be robustly implemented and tested.

- 🔴 User Tip Edit Policy & RLS Definition (P1): Business rules for users editing their own tips (especially after approval) \*must\* be finalized. RLS policies and any associated application logic/triggers must correctly manage `moderation\_status` changes upon such edits.

- 🔴 Confirm `waypoint\_id` Data Type (P1): Verify that `integer` (or chosen type) for `waypoint\_id` in `user\_waypoint\_votes` and `user\_waypoint\_short\_tips` matches `public.waypoints.id` type. Adjust all related DDL if necessary.

- 🔴 Moderator Identification Function (P1): The `public.check\_if\_user\_is\_moderator(UUID)` function for `user\_waypoint\_short\_tips` RLS \*must\* be implemented.

- 🟠 Translation Data for Categories (P2): Ensure initial `tip\_categories\_master` seed data has corresponding entries in `public.translations` for all supported languages.

- 🟠 Application Logic for Vote Lifecycle (P2): The UPSERT pattern for vote handling needs robust implementation in the application backend.

11\. Scalability & Future-Proof Notes

- Partitioning: Not anticipated for V1 on `user\_waypoint\_votes` or `user\_waypoint\_short\_tips`. Denormalization and indexing are primary strategies.

- Text Search (V2+ for Tips): For `user\_waypoint\_short\_tips.tip\_text`, plan for PostgreSQL Full-Text Search (FTS) by adding a `tsvector` column and GIN index.

- Vote Abuse/Spam (V2+ for Votes): Application-level rate limiting or other detection mechanisms may be needed.

- Tip "Helpfulness" Voting / Reporting (V2+ for Tips): Current schema allows for future extension.

- Audit Columns & Soft Deletion: Provide good foundational audit and data preservation.

12\. Next Steps (Module Finalization)

1. P1: Implement `waypoints` Table Modifications for vote counts.

2. P1: Implement and Test `update\_waypoint\_vote\_counts` Trigger.

3. P1: Confirm and Align `waypoint\_id` Data Type across tables.

4. P1: Finalize and Implement User Tip Edit Policy (RLS and any app/DB triggers).

5. P1: Define and Implement `public.check\_if\_user\_is\_moderator(UUID)`.

6. P1: Create `tip\_categories\_master` table, `view\_tip\_categories\_localized`, associated triggers, RLS, and seed data (including planning for `translations` entries).

7. P2: Fully Implement and Test All RLS Policies for all module tables.

8. P2: Develop and Test Application Backend Logic for vote lifecycle (UPSERT) and tip submission/moderation workflows.

9. P2: If applicable, create and run backfill script for denormalized vote counts on `waypoints`.

10. P2: Document the `app.current\_lang` GUC setup or chosen language context mechanism for `view\_tip\_categories\_localized`.

\* \* \* \* \*