Influencer Engagement & Sponsorship Coordination Platform (V2)

G HAMSINI

22F3000767

Overview and Frameworks

This platform connects sponsors with influencers, enabling sponsors to advertise their products/services while influencers gain monetary benefits.

Core Frameworks

SQLite: For data storage (mandatory).
Flask: For API development.
Bootstrap: For styling and responsive design.
Redis & Celery: For caching and asynchronous tasks (batch jobs).

Roles

Admin:
- Full monitoring of campaigns, ad requests, and users.
- Can flag inappropriate content.

Sponsor:
- Creates campaigns, manages ad requests, and tracks performance.
- Searches for suitable influencers.

Influencer:
- Accepts/rejects ad requests, negotiates terms, and participates in public campaigns.

Terminologies

Ad Request: A contract between a campaign and an influencer defining advertisement requirements and payment.
Campaign: A goal-oriented initiative by sponsors to manage multiple ad requests.

Functionalities and Advanced Features

Core Functionalities

Role-Based Access Control (RBAC):
- Secure login/register system for Admin, Sponsor, and Influencer roles.

Admin Dashboard:
- View statistics on active users, campaigns, and flagged content.
- Approve new sponsor signups.

Campaign Management (Sponsors):

- Create, update, and delete campaigns.
- Manage visibility (public/private) and set goals.

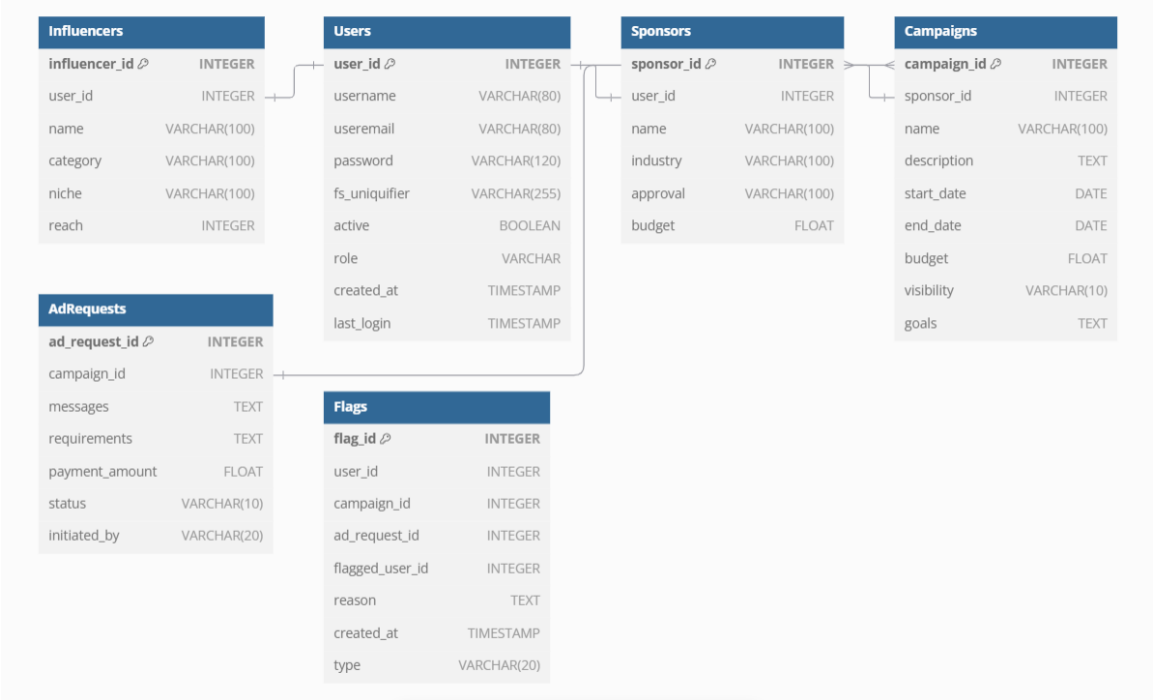
Ad Request Management (Sponsors & Influencers):

- Create, update, and delete ad requests.
- Accept/reject requests and negotiate terms.

Search Functionality:

- Sponsors can search influencers by niche, reach, and followers.
- Influencers can search for public campaigns by relevance.

Database:



Key Relationships

1. User ↔ Role: Many-to-many through the roles_users table.

2. Sponsor ↔ User: One-to-one relationship.

3. Influencer ↔ User: One-to-one relationship.

4. Campaign ↔ Sponsor: Many-to-one relationship.

5. AdRequest ↔ Campaign: Many-to-one relationship.

6. AdRequest ↔ Influencer: Many-to-one relationship.

7. Flag ↔ Campaign / AdRequest / User: Tracks flagged items.

Asynchronous Tasks (Using Celery & Redis)

monthly_report

Generates a monthly performance report for sponsors, detailing campaigns, ad requests, and budgets used.

Converts the report to a PDF and sends it as an email attachment to each sponsor.

send_daily_reminder

Sends daily reminders to influencers about pending ad requests via email.

Includes details about campaigns and ad requests that require their attention.

generate_campaign_csv

Exports campaign details for a sponsor into a CSV file, including fields like name, description, budget, and goals.

Saves the file with a timestamped name and provides a download link for the sponsor.

VUE JS

1. Dynamic Rendering:

Vue.js is used to dynamically render data from the Flask backend. For example, when viewing ad requests or campaigns, Vue.js is responsible for fetching the data asynchronously via API calls (e.g., using axios), and displaying it in tables, modals, or other UI components.

2. Component-Based Architecture:

Vue.js utilizes a component-based structure where different parts of the page (e.g., tables, modals, forms) are broken down into reusable components. This enhances maintainability and makes it easy to update parts of the UI without affecting the entire application.

3. Modals and Form Handling:

Vue.js is used to handle user interactions with modals for adding, editing, or deleting ad requests. Form inputs are dynamically bound to Vue's reactive data, allowing for real-time validation and updates.

4. Asynchronous API Calls:

With Vue.js, asynchronous calls are made to backend endpoints (via 'axios') to fetch or manipulate data, such as creating or updating ad requests, campaigns, or sending reminders. The responses are then used to update the frontend state without reloading the page.

5. State Management:

Vue.js uses reactive data properties to track the state of the application, such as the list of ad requests or campaigns. This makes it easy to automatically update the UI when data changes.

6. Routing:

Vue.js uses Vue Router to handle navigation between different pages, such as navigating to the 'sponsor_adrequests' page or the campaign details page. This allows for a smooth, single-page application (SPA) experience.

7. Event Handling:

Vue.js is used to manage user events, such as button clicks to open modals, submit forms, or trigger updates. The use of Vue's event listeners (`@click`, `@submit`) helps in managing interactions between UI components.

Link : https://drive.google.com/file/d/1LWHv__dTZ9uPQ5IbSzzFnCWed-qnM5dO/view?usp=drive_link