Influencer Engagement and Sponsorship Platform - V2

A platform to connect influencers to sponsors

A Project report for the MAD II Project

Submitted by:-

Name: Tathagat Suman

Roll number: 22F3002936

22f3002936@ds.study.iitm.ac.in

IITM Online BS Degree Program,

Indian Institute of Technology, Madras, Chennai

Tamil Nadu, India, 600036

Question Statement:

The platform connects sponsors and influencers to facilitate advertising and monetary benefits. Sponsors can create campaigns, search for relevant influencers, and manage ad requests. Influencers can browse public campaigns, accept or reject ad requests, and negotiate terms with sponsors.

Approach:

To solve the problem, I developed a platform that includes three types of users: Admin, Sponsors, and Influencers. Each user type has different functionalities:

- Admin: Manages users, campaigns, and ad requests, with the ability to flag inappropriate content.
- Sponsors: Create campaigns, search for influencers, send ad requests, and track campaign performance.
- Influencers: Search for relevant campaigns, accept or reject ad requests, and manage their profiles.

Database Design: Designed the database schema to manage users, campaigns, requests, and influencer information. Ensured data integrity with relationships and constraints.

Backend Implementation: Used Flask for server-side logic. Implemented user authentication and authorization with JWT, and handled database interactions using SQLAlchemy.

Frontend Implementation: Created responsive SPA Frontend using vue.js framework of javascript. Developed forms for user registration, campaign management, and request handling.

APIs and Routes: Implemented RESTful routes for CRUD operations on users, campaigns, and requests, ensuring appropriate permissions and access control.

Frameworks and Libraries Used:

Flask: For creating the web application and handling routing, rendering templates, and request processing.

Flask-SQLAlchemy: For ORM-based database interactions.

Flask-Migrate: For handling database migrations.

Flask-JWT-Extended: For user authentication and session management using JWT.

SQLite: As the database for storing user, campaign, request and blacklisted token data.

Flask-Caching: For caching frequently accessed data using Redis.

Celery: For scheduling background tasks, using Redis as the message broker.

Flask-Cors: For handling cross-origin requests.

Vue.js: For client-side rendering and creating a responsive frontend.

ER Diagram:

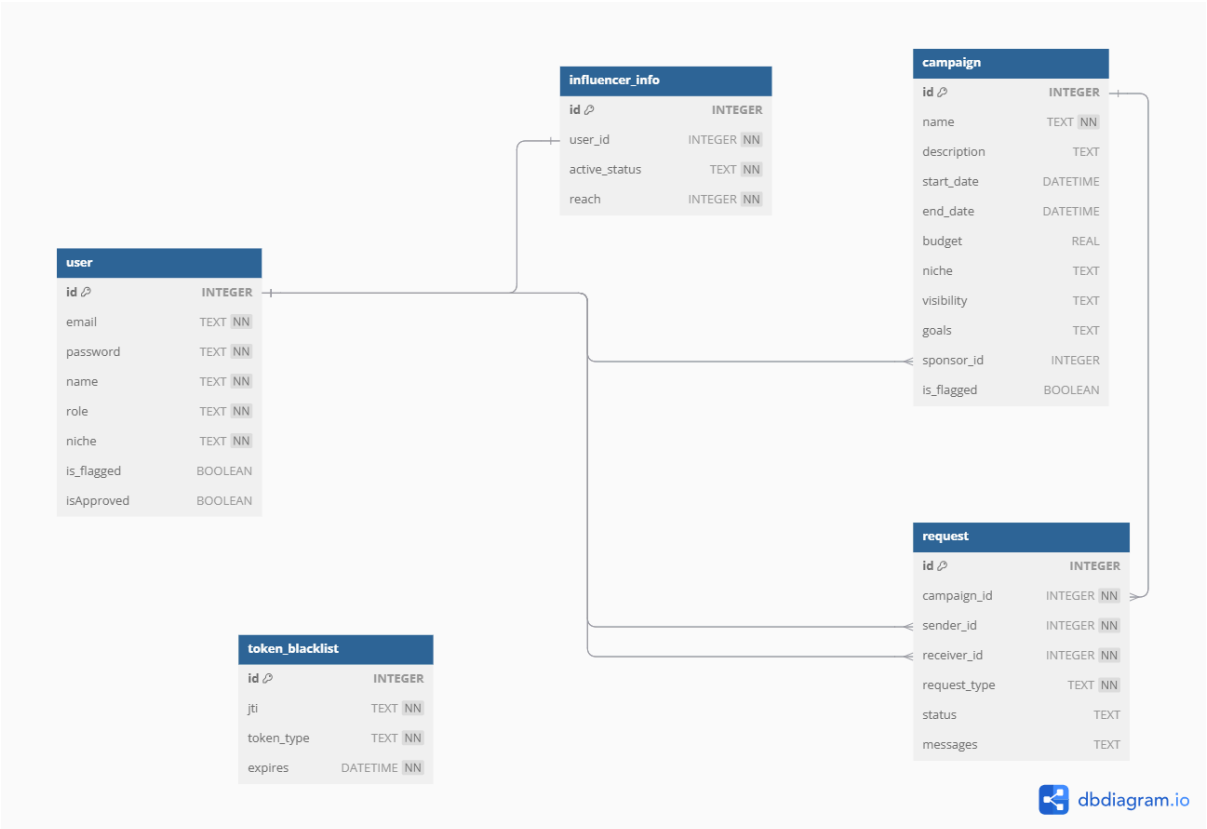

fig. ER-Diagram of the Database Schema that I will be using.

Entities:

User: Represents users of the system with roles of admin, sponsor, or influencer. Manages influencer-specific information and campaigns.

InfluencerInfo: Contains additional details for users with the role of influencer.

Campaign: Represents campaigns created by sponsors, including attributes like budget, description, and visibility.

Request: Represents requests sent between influencers and sponsors related to campaigns.

TokenBlacklist: Stores invalidated tokens to prevent unauthorized access.

Relationships:

One-to-One: User to InfluencerInfo

One-to-Many: User to Campaign, User to Request (as sender/receiver)

Many-to-One: Request to Campaign

API Resource Endpoints:

GET / and /path:path: Index.html or static file serving.

GET /login: Login page.

POST /login: Handle login form submission.

POST /refresh: Handle new access token generation with refresh token.

GET /register: Registration page.

POST /register: Handle user registration.

POST /logout: Handle access token invalidation.

POST /logout_refresh: Handle refresh token invalidation.

GET /dashboard: Dashboard for users based on their role.

PUT /admin/approve_sponsor/int:user_id: Approve or Reject a user(sponsor) (admin only).

PUT /admin/flag/user/int:user_id: Flag a user (admin only).

PUT /admin/unflag/user/int:user_id: Unflag a user (admin only).

DELETE /delete_user/int:user_id: Delete a user (admin only).

POST /create_campaign: Create a new campaign (sponsor only).

PUT /edit_campaign/int:campaign_id: Edit an existing campaign (sponsor only).

GET /sponsor/export_csv_report: Exports csv-report for prev. months activity (sponsor only).

PUT /admin/flag_campaign/campaign/int:campaign_id: Flag a campaign (admin only).

PUT /admin/unflag_campaign/campaign/int:campaign_id: Unflag a campaign (admin only).

DELETE /delete_campaign/int:campaign_id: Delete a campaign.

POST /send_request/int:campaign_id/int:receiver_id: Send a request to an influencer.

POST /request_influencer: Send a request from a sponsor to an influencer.

PUT /update_request/int:request_id: Edit or Update a request data.

DELETE /delete_request/int:request_id: Delete a request.

PUT /manage_request/int:request_id/<action>: Accept or reject a request.

PUT /profile: Update profile information.

Drive Link of the Presentation Video:

Presentation Video Link: click here to view