Influencer Sponsorship Co-ordination Platform - V2 – Project Report

Author Name: Kisalay Pan
Roll No: 22f2001094
Email ID: 22f2001094@ds.study.iitm.ac.in

---

1. Introduction

- **Project Title:** Brandifiers - Influencer Sponsorship Co-ordination Platform - V2
- **Project Summary:** This platform enables brands to efficiently coordinate and manage influencer sponsorship campaigns. It facilitates multiple user roles, such as **Admin**, **Influencers**, and **Advertisers**, offering an integrated approach to sponsorships, tracking, and performance analytics.
- **Objectives:**
    - To streamline the process of influencer sponsorship by automating the management of campaigns, statistics, and ad requests.
    - To implement a secure and role-based access control system for managing different types of users (Admin, Influencer, Advertiser).

---

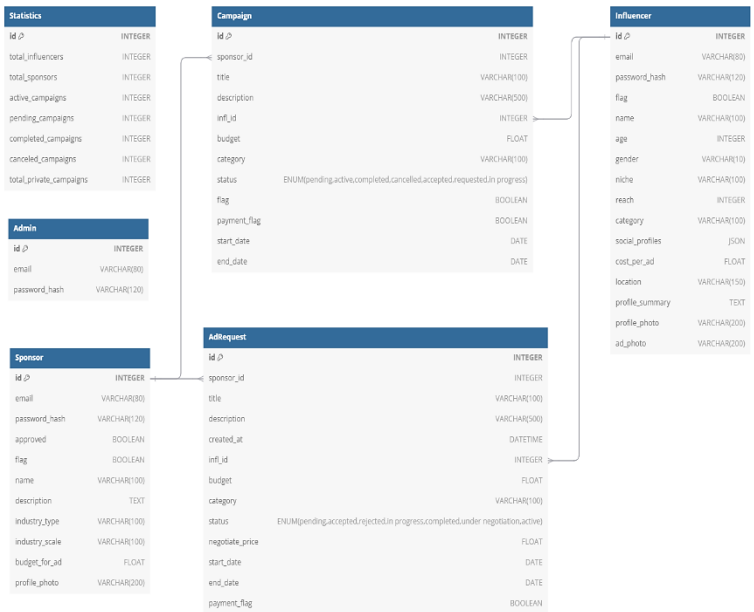
2. Technology Stack

- **Backend:** Flask, Flask-SQLAlchemy, Flask-Migrate for database management, Flask-JWT-Extended for authentication, and Celery for handling asynchronous tasks.
- **Frontend:** Vue.js for the user interface, Bootstrap for styling.
- **Database:** SQLite for storing user data, influencer campaigns, ad requests, and statistics.
- **Additional Tools:** Redis for caching, Celery for asynchronous task management.
- **Platform:** Deployed on a Linux virtual machine with PowerShell for Windows.

---

3. Database Models

- **Influencer Model:** Stores information about influencers, their social media platforms, audience metrics, and associated campaigns.
- **Admin Model:** Manages the system, including handling influencer, campaign, and advertisement approvals.
- **AdRequest Model:** Represents ad requests made by advertisers, and tracks the status of those requests (pending, accepted, completed).
- **Campaign Model:** Stores information about different campaigns, including the influencer associated with the campaign, advertisement content, and status.
- **Statistics Model:** Stores and tracks performance metrics for each campaign, including views, engagement rates, and ROI.



4. Features and Functionalities

- **User Authentication and Role-Based Access:**
  - Implemented with Flask-JWT-Extended, which uses JSON Web Tokens (JWT) for secure authentication and role-specific access.
  - Three user roles: **Admin**, **Influencer**, and **Advertiser**, each with specific access privileges.
- **Campaign Management:**
  - Admins can create, manage, and assign campaigns to influencers.
  - Influencers can view their assigned campaigns, ad requests, and their performance metrics.
- **Ad Request System:**
  - Advertisers can submit ad requests, view available influencers, and track their ad performance.
- **Statistics and Analytics:**
  - The system provides detailed statistics on the performance of campaigns, including data like views, likes, engagement rates, and ROI.
- **Asynchronous Task Handling (Celery):**
  - Celery, integrated with Redis, handles long-running tasks such as sending notifications, processing data, or generating reports asynchronously.

---

5. Development Process

- **Initial Setup:**
  - Created a virtual environment and installed the required packages using pip.
  - Set up Flask, Celery, Redis, and configured database models using Flask-SQLAlchemy.
- **Feature Implementation:**
  - Developed API routes for each user role (Admin, Influencer, Advertiser).
  - Integrated Celery to handle asynchronous tasks such as sending notifications and updating campaign statistics.
  - Created dynamic views for campaign and ad request management using Vue.js.
- **Testing:**
  - Ensured role-specific access controls were working correctly.
  - Verified that the ad request system was functioning and campaign performance data was accurate.

---

6. Challenges and Solutions

- **Asynchronous Task Management:**
  - Initially, there were issues with task completion time and queue management. These were addressed by properly configuring Redis and Celery settings.
- **Role-Based Access Control:**
  - Handling role-specific access to various system parts caused conflicts with routing. Solved by refining JWT token parsing and role-checking mechanisms.
- **Database Optimization:**
  - As the data grew, performance issues occurred during large queries. Optimized queries and introduced indexing for faster lookups, especially for the statistics model.

---

7. Conclusion

- **Outcomes:**
  - The **Brandifiers** platform enables a seamless, role-specific approach to influencer sponsorship management. Advertisers and influencers can now coordinate more effectively, while Admins can efficiently monitor and manage all activities.
- **Future Work:**
  - Potential future features include:
    - **Payment Gateway Integration** for advertisers to directly pay influencers.
    - **Enhanced Reporting Tools** for deeper insights into campaign performance.
    - **AI-Powered Recommendations** for advertisers to find the best influencers for their campaigns based on past data.

---

8. Presentation

- **Drive:** Click to view Video