Project Report

Student Details:

- **Name:** Pranav Tiwari
  Student ID: 22f3001519@ds.study.iitm.ac.in
- **Course:** Modern Application Development II

Project Details:

- **Project Title: I**nfluencer Engagement & Sponsorship Coordination Platform - V2

- **Objective:** To develop a platform that connects sponsors with influencers, allowing sponsors to promote their products/services while influencers gain monetary benefits. The platform is built using SQLite, Flask, VueJS, Bootstrap, Redis, and Celery.

Problem Statement:

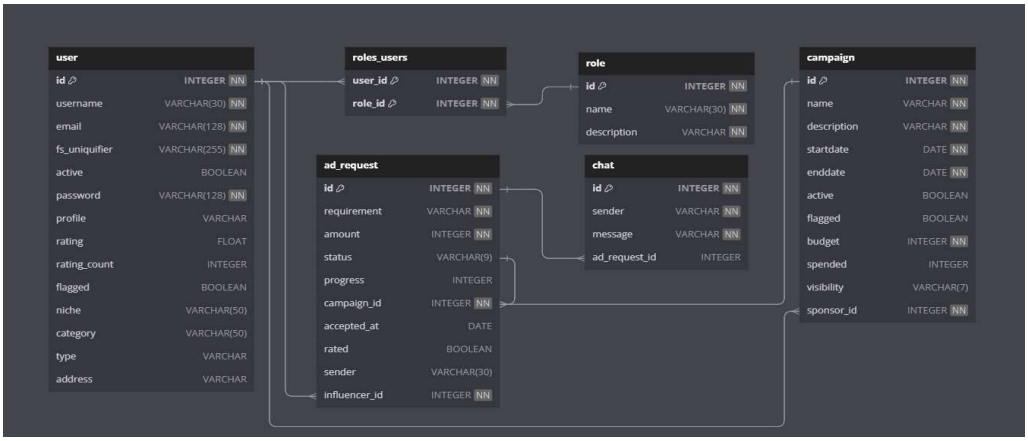
The project aims to build a scalable and secure platform that connects influencers and sponsors. Sponsors can create campaigns and send advertisement requests to influencers, while influencers can accept, reject, or negotiate these requests. The system also provides an admin panel to oversee activities, monitor users, and approve new sponsors. Core features include user authentication, campaign and ad request management, batch jobs, and caching for improved performance.

Frameworks and Libraries Used:

- - **Database:** SQLite
- - **Backend Framework:** Flask, Jinja2 for templating
- - **Framework:** VueJS, Bootstrap for responsive design
- - **Caching:** Redis
- - **Task Queue and Batch Jobs:** Redis and Celery
- - **Authentication:** Flask-Security

ER Diagram:

The following is the ER diagram of the modal used in the Project (consisting of all the tables and their relationships):



Drive Link of the Presentation Video:

https://drive.google.com/file/d/172o83OJFtM5kYuMxmxoCd2pNqlPlvAwq/view?usp=sharing

API Endpoints:

[("/api/influencerdetail",POST) , ("/api/sponsordetail",POST) , ("/api/admindetail",POST) , ("/api/flag",POST) , ("/api/pendingsponsorlist",GET) , ("/api/get_sponsors",POST) , ("/api/get_influencers",POST) , ("/api/get_campaigns",POST) , ("/api/campaign",POST) , ("/api/campaign",PUT) , ("/api/ad_request",POST) , ("/api/ad_request",PUT) , ("/api/acceptrequest",POST) , ("/api/rejectrequest",POST) , ("/api/deleterequest",POST) , ("/api/getadrequests",POST) , ("/api/getchat",POST) , ("/api/sendmessage",POST) , ("/api/stats",GET) , ("/api/updateprogress",POST) , ("/api/rate",POSt)]

Approach:

1. User Authentication & Role-Based Access: Implemented using Flask-JWT for secure access and role management.

2. Admin Dashboard: Displays statistics such as active users, flagged users, active campaigns, and pending ad requests.

3. Campaign & Ad Request Management: Sponsors can create, update, and delete campaigns. They can also send, edit, and track ad requests.

4. **Influencer Management:** Influencers can view, accept, reject, or negotiate ad requests and update their profile page.

5. Backend Jobs:

- Daily reminders for pending ad requests using Mails.
- Monthly activity reports generated using HTML and sent via email.
- User-triggered batch job for exporting campaign details as CSV.

6.Caching & Performance: Redis is used to cache frequently accessed data and improve API response time.