MAD-1 Project – Influencer Engagement and Sponsorship Coordination Platform

Aniruddha Mukherjee[1,2]

[1]School of Computer Engineering, Kalinga Institute of Industrial Technology, Street, Bhubaneswar, 751024, Odisha, India

[2]BS Data Science and Programming, Indian Institute of Technology, Madras, 600036, Tamil Nadu, India

[*]Corresponding author: `mukh.aniruddha@gmail.com`, `23f1003186@ds.study.iitm.ac.in`

Abstract

Project Report for MAD-1 Project

This is the project I made for MAD-1. It is an Influencer Engagement and Sponsorship Coordination Platform. It's a platform to connect Sponsors and Influencers so that sponsors can get their product/service advertised and influencers can get monetary benefit.

Keywords: flask, application-development, jinja2, sqlite, postman, apis.

1. Introduction

This is an app which enables sponsors (big companies) to reach out to influencers (celebrities) to advertisements. This is done by the sponsors to increase engagement with their products and this is called influencer marketing. With social media on the rise and the internet occupying a ubiquitious position in our lives, influencer marketing is a major source of income for companies and a big driver of revenue. This application, aims to solve this problem.

It has 3 roles

- Admin.
- Sponsor.
- Influencer.

1.1 TLDR;

The Admin can monitor everything going on in the application. The sponsor, is the company that wants to create "campaigns", which is nothing but a container for multiple ad-requests to be grouped under. A particular campaign contains any-number of ad-requests to any number of influencers. It is obvious that the ad-requests under that campaign should related to the overall goal of that campaign. The influencer is the celebrity that has a large following on social media or otherwise. These influencers can monetize their follower-base by accepting or rejecting ad-requests made to them from sponsors. The influencers usually have to either use the sponsor's product or service in ad-shoots (for TV ads) or instragram posts etc.

So the broad overall summary of the application is, that it enables sponsors to create campagins (containers for ad-requests). Sponsors then can create ad-requests within campaigns with two modes, one mode where the sponsor has already identified the influencer they want to request (so they already know who the ad is for) and the other mode whereby the the sponsor leave the ad-request open. If the campaign is public, then this ad-request will be visibile to all influencers. Influencers can show their interest by "requesting" to be a part of the ad-request. Multiple influencers can request to the same 'open' ad-request made by the sponsor. The sponsor then can decide whether to Accept or Reject a particular influencer's request. Note that a particular ad-request can have only one influencer assigned to it. Influencers can accept / reject directy ad-requests made to them by sponsors.

2. Login

The admin is pre-pushed into the database when the application runs for the first time. After that influencers and sponsors can "register" themseleves with the registration form, which is pushed to the database in the backend. The password is hashed and stored & checked later on in the database using the `generate_password_hash` and `check_password_hash` respectibely, both of which are available in `werkzeug.security`.

3. Admin Functions

Admins can view all sponsors currently registered. Admins can view all influencers currently registered and admins can also view all the ongoing campaigns (public or private). The admin can search for these entities and also filter on the basis of the role (admin/sponsor/influencer) The admins can also see the progress of a particular campaign. This is calculated on the basis of the staring and ending date of the campaign.

$$\text{campaign\_progress\%} = \frac{(today - starting\_date)_{\text{in days}}}{(ending\_date - starting\_date)_{\text{in days}}} \tag{1}$$

Admins can also view all the ad-requests made (accepted / rejected / pending / pending_influ_req).

The Admin also has the ability to flag a particular user (influencer / sponsor) following which the user is locked out of their account on the login page.

Lastly the Admin has a statistics page using which they can view the global statistics of the application.

4. Sponsor Functions

A sponsor can login and see their current campaigns. They can view the campaign and see the number of ad-requests currently ongoing. The sponsor also has a section called "Influencer Requests". Here, requests made by influencers to "Open" ad-requests put out by them are displayed with 3 options to View/Accept/Reject.

The sponsor can search for influencer profiles by filtering on the basis of their niche, their name or their minimum reach.

The sponsor can create campaigns with a certain budget, however the budget of the campaign must be lesser than the total budget of the influencer. Faliure to meet the budget requirement leads to an error message being flashed on the screen using flask's inbuilt flash feature.

Within the campaign, the sponsor can create multiple ad-requests. The payment amount for an ad-request cannot exceed the campaign budget, and if an ad-request has already been created, the new ad-requests payment amount can't exceed the "remaining campaign budget" which is obtained by

$$\text{Remaining\_Campaign\_Budget} = \text{Total\_Campaign\_Budget} - \sum_{i=1}^{n} \text{Payment\_Amount}_i \qquad (2)$$

This equation calculates the remaining campaign budget by subtracting the total payment amounts for all ad requests (from 1 to n) from the initial campaign budget. Each Payment_Amount$_i$ represents the cost associated with the i-th ad request in that campaign.

The sponsor can create two types of ad-requests.

One, where the sponsor knows which influencer they want to request beforehand, so when the ad-request form is being filled out, the specific influencer is selected and a request appears in that influencer's dashboard only.

Another, where the sponsor is unsure about which influencer to request. So under a public campaign, they set the ad-request status to "Open". All influencers can view this 'Open" ad-request in their "Find Campaigns" view. Then multiple influencers can request to be a part of that ad-request. These requests pool into the sponsor's dashboard. The sponosor then can review these applications and pick the most suitable influencer, rejecting all the other influencers.

The sponsor has access to various statistics on their dashboard, such as amount spent from their budget and a chart depicting the Influencer's name on the x-axis and the number of ad-requests ongoing with the sponsor on the y-axis i.e. Ad-count by influencer. Lastly the sponsor can also view basic details such as total campaigns, total ads (accepted/rejected/pending).

5. Influencers

Influencers are greeted with their Active Ads on their dashboard and a section called “New Requests” where new ad-requests made by sponsors to them directly show up. Influencers can also “Find” or browse all public and non-flagged campaigns. If a particular campaign has a “Open Request” a green badge with the text “Open Request” appears in the “Find” view for the influencers. The influencer then can click into the campaign and view the details of said ad-request and click “Request” which sends a request to the sponsor’s dashboard under the “Influencer Requests” section.

Influencers can also view basic statistics such as their earning

$$\text{Influencer Earning} = \sum_{\text{active accepted ad-requests}} \text{payment amount} \tag{3}$$

6. API

Endpoint	**Method**	**Description**
`/api/campaigns`	GET	List all campaigns
`/api/campaigns/<campaign_id>`	GET	Get campaign by ID
`/api/ad_requests/<int:ad_request_id>`	DELETE	Delete Ad-request
`/api/sponsors`	GET	List all sponsors
`/api/sponsors/<sponsor_id>`	GET	Get sponsor by ID
`/api/influencers`	GET	List influencers
`/api/influencers/<influencer_id>`	GET	Get influencer
`/api/ad_requests`	GET	List ad requests
`/api/ad_requests/<ad_request_id>`	GET	Get ad request

Table 1: API Endpoints

I have decied to integrate the api routes directly into my controllers and not use a package such as flask_restful.

I have made mainly GET endpoints however I have included one DELETE endpoint as well. I have tested these endpoints with postman.

7. Tech Used

- Code Editor: VS Code
- Backend Framework: Flask
- Frontend Framework: Bootstrap (HTML, CSS)
- Frontend Dynamic HTML content: Jinja2

- Object-Relational Mapping Tool: SQLAlchemy
- Database: SQLite
- Charts: ChartJS
- API Tester: Postman

8. ER-Diagram

Below is the ER-Diagram for my App:

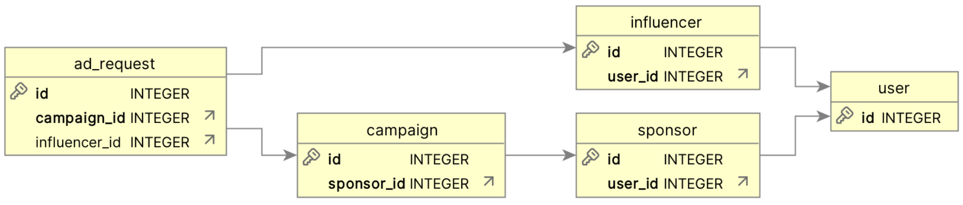


Figure 1: ER Diagram with only the Primary Keys and Foreign Keys

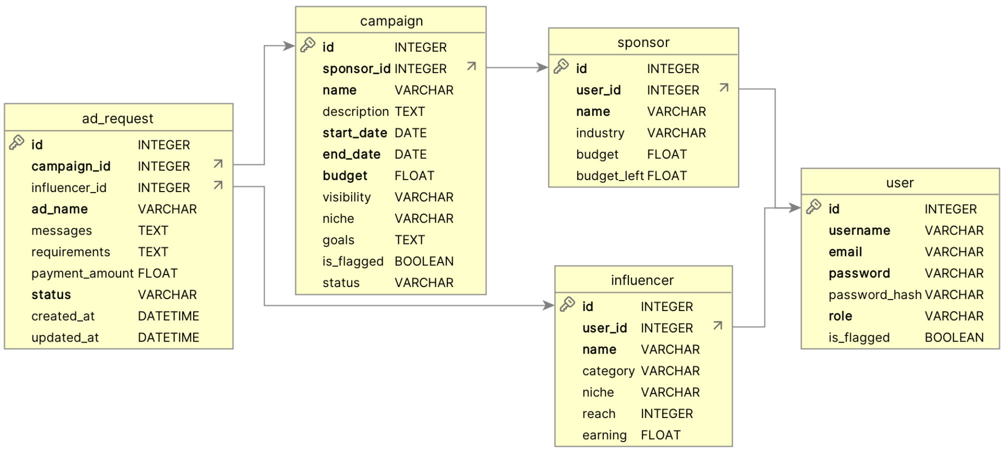
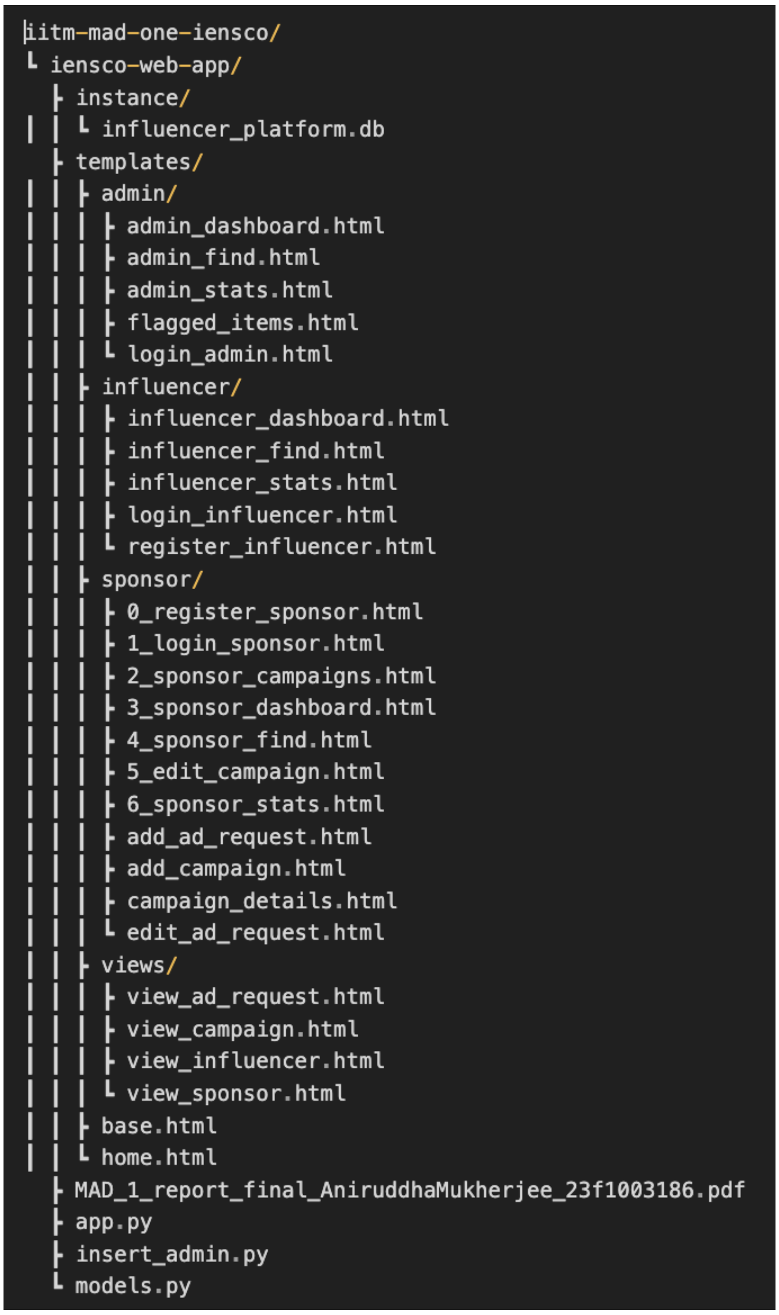


Figure 2: ER Diagram with all attributes.

9. Video Demo

YouTube (unlisted): https://youtu.be/LwQO7TaMMa0
IITM-Drive Link: https://drive.google.com/file/d/1JBOhJ4Naj1g0VdN0xDuI-NSZjgoOIdzu/view?usp=sharing

10. Directory Tree



```
iitm-mad-one-iensco/
└ iensco-web-app/
  ├ instance/
│ │ └ influencer_platform.db
  ├ templates/
│ │ ├ admin/
│ │ │ ├ admin_dashboard.html
│ │ │ ├ admin_find.html
│ │ │ ├ admin_stats.html
│ │ │ ├ flagged_items.html
│ │ │ └ login_admin.html
│ │ ├ influencer/
│ │ │ ├ influencer_dashboard.html
│ │ │ ├ influencer_find.html
│ │ │ ├ influencer_stats.html
│ │ │ ├ login_influencer.html
│ │ │ └ register_influencer.html
│ │ ├ sponsor/
│ │ │ ├ 0_register_sponsor.html
│ │ │ ├ 1_login_sponsor.html
│ │ │ ├ 2_sponsor_campaigns.html
│ │ │ ├ 3_sponsor_dashboard.html
│ │ │ ├ 4_sponsor_find.html
│ │ │ ├ 5_edit_campaign.html
│ │ │ ├ 6_sponsor_stats.html
│ │ │ ├ add_ad_request.html
│ │ │ ├ add_campaign.html
│ │ │ ├ campaign_details.html
│ │ │ └ edit_ad_request.html
│ │ ├ views/
│ │ │ ├ view_ad_request.html
│ │ │ ├ view_campaign.html
│ │ │ ├ view_influencer.html
│ │ │ └ view_sponsor.html
│ │ ├ base.html
│ │ └ home.html
  ├ MAD_1_report_final_AniruddhaMukherjee_23f1003186.pdf
  ├ app.py
  ├ insert_admin.py
  └ models.py
```

Figure 3: Filetree

11. References

1. For ER Diagram: https://www.dbvis.com/download/
2. For fundamentals of login, flash: https://github.com/annimukherjee/Notes-App-Flask
3. For Bootstrap Documentation (full frontend made with this!):

   https://getbootstrap.com/docs/4.1/getting-started/introduction/

4. For API testing: https://www.postman.com/
5. Flask Docs: https://flask.palletsprojects.com/en/3.0.x/

12. Conclusions

This concludes my MAD-1 Project Report.

Really thankful to IIT-M for assigning this project. It pushed me outside of my comfort zone which was uncomfortable, but ultimately was good for me!

Conflicts of Interest

The authors have no conflict of interests related to this publication.

Author Contributions

Aniruddha Mukherjee: Conceptualization, Methodology, Software.

Funding

No funding.

Data Availability

The entire code base will be available publicly post the May term. I will upload everything to https://github.com/annimukherjee/

Acknowledgments

Thank you to the internet, random stack-overflow blogs, flask docs and IITM course instructors!