PROJECT REPORT

MODERN APPLICATION DEVELOPMENT

ANMAYA PANDA
22F1001109
22f1001109@ds.study.iitm.ac.in

I am a final year undergraduate student pursuing BE in Mechatronics Robotics and Automation engineering at Thapar University and a BS in Data Science and Applications from Indian Institute of Technology, Madras. I am currently in the Diploma level

Project Description
It's a platform to connect Sponsors and Influencers so that sponsors can get their product/service advertised and influencers can get monetary benefit.

Frameworks to be used.

Flask for application code
Jinja2 templates + Bootstrap for HTML generation and styling
SQLite for data storage
VueJs for UI
Redis for Caching

In this project, I am making a Influencer Engagement and Sponsorship Platform using HTML, Bootstrap and Jinja2 templates for HTML generation and styling and SQLite for data storage. In the User dashboard, all the relevant statistics can be viewed. A user can login as Influencer and Sponsor. In sponsor dashboard, the user can ad/edit/delete campaigns and manage user

Entity and Relationship Diagram

Users Table:

- Stores general user information.
- Includes all types of users (`Admin`, `Sponsor`, `Influencer`).
- Each user has a unique `username` and a role.

Sponsors Table:

- Contains details specific to sponsors.
- Linked to the `Users` table through a foreign key `id`.

Influencers Table:

- Contains details specific to influencers.
- Linked to the `Users` table through a foreign key `id`.

Campaigns Table:

- Stores information about campaigns created by sponsors.
- Each campaign is associated with a sponsor (`sponsor_id`).

AdRequests Table:

- Represents advertisement requests linking campaigns to influencers.
- Contains details like `requirements`, `payment_amount`, and `status`.
- Linked to both `Campaigns` and `Influencers`.

FlaggedUsers Table:

- Tracks users who have been flagged for various reasons.
- Linked to the `Users` table.

FlaggedCampaigns Table:

- Tracks campaigns that have been flagged.
- Linked to the `Campaigns` table.

Messages Table:

- Stores communication between sponsors and influencers regarding ad requests.
- Linked to both `AdRequests` and `Users` (as `sender_id`).

Users
-----
id (Primary Key, Integer)
username (String, Unique, Not Null)
password (String, Not Null)
role (String, Not Null) # 'Admin', 'Sponsor', 'Influencer'

Sponsors
----
id (Primary Key, Integer, Foreign Key -> Users.id)
company_name (String)
industry (String)
budget (Float)

Influencers

```
id (Primary Key, Integer, Foreign Key -> Users.id)
category (String)
niche (String)
reach (Integer)
```

Campaigns

```
---------
id (Primary Key, Integer)
name (String, Not Null)
description (Text, Not Null)
start_date (Date, Not Null)
end_date (Date, Not Null)
budget (Float, Not Null)
visibility (String, Not Null)  # 'public', 'private'
goals (Text, Not Null)
sponsor_id (Integer, Foreign Key -> Sponsors.id)
```

AdRequests

```
----------
id (Primary Key, Integer)
campaign_id (Integer, Foreign Key -> Campaigns.id, Not Null)
influencer_id (Integer, Foreign Key -> Influencers.id, Not Null)
messages (Text)
requirements (Text, Not Null)
payment_amount (Float, Not Null)
status (String, Not Null)  # 'Pending', 'Accepted', 'Rejected'
created_at (DateTime, Default = current timestamp)
```

FlaggedUsers

```
------------
id (Primary Key, Integer)
user_id (Integer, Foreign Key -> Users.id, Not Null)
reason (Text, Not Null)
flagged_at (DateTime, Default = current timestamp)
```

FlaggedCampaigns

```
----------------
id (Primary Key, Integer)
campaign_id (Integer, Foreign Key -> Campaigns.id, Not Null)
reason (Text, Not Null)
flagged_at (DateTime, Default = current timestamp)
```

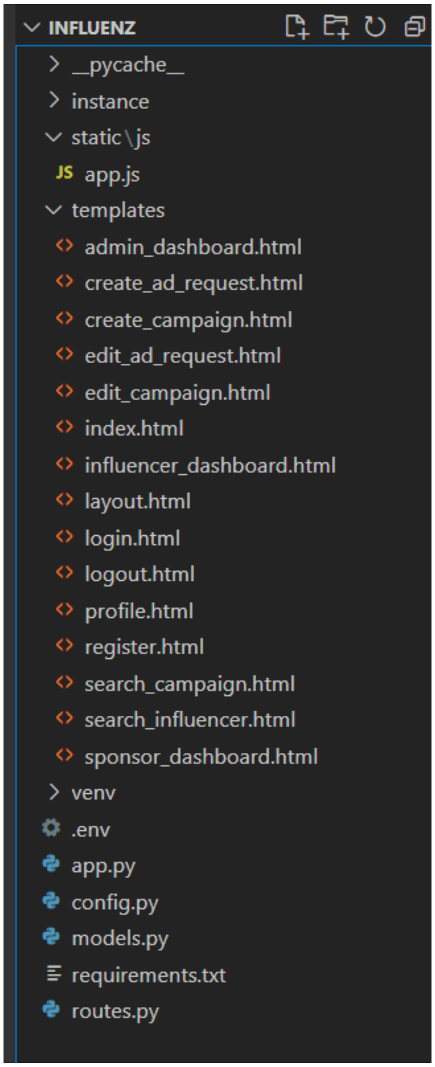
Messages

```
--------
```

id (Primary Key, Integer)
ad_request_id (Integer, Foreign Key -> AdRequests.id, Not Null)
sender_id (Integer, Foreign Key -> Users.id, Not Null)
content (Text, Not Null)
sent_at (DateTime, Default = current timestamp)

Relationships:

- User has a one-to-one relationship with either Influencer or Sponsor (exclusive)
- Sponsor has a one-to-many relationship with Campaign
- Influencer has a one-to-many relationship with AdRequest
- Campaign has a one-to-many relationship with AdRequest
- AdRequest references both Campaign and Influencer (many-to-many)



```
INFLUENZ
  __pycache__
  instance
  static\js
    app.js
  templates
    admin_dashboard.html
    create_ad_request.html
    create_campaign.html
    edit_ad_request.html
    edit_campaign.html
    index.html
    influencer_dashboard.html
    layout.html
    login.html
    logout.html
    profile.html
    register.html
    search_campaign.html
    search_influencer.html
    sponsor_dashboard.html
  venv
  .env
  app.py
  config.py
  models.py
  requirements.txt
  routes.py
```

Project Directory

Video Link -
https://drive.google.com/file/d/1yqheUFEthvvRltcMj2NkBbrdvOmONZv-/view?usp=sharing