The webAPP - Influencer Marketing Platform

Author

Name: Rahul Kumar Singh
Roll No: 22f3001530
Email: 22f3001530@ds.study.iitm.ac.in

About me

I'm pursuing my BS in Data Science at IIT Madras. Through this Influencer Marketing Platform project, I've enhanced my skills in web development, particularly focusing on technologies like Python, Flask, SQLAlchemy, HTML, and CSS.

Description

Developed The webAPP, a Flask-based web application for connecting sponsors with influencers. Implemented user authentication, campaign management, influencer search, ad request system, and an admin dashboard for platform oversight.

Technology used

1. **Python:** Backend development
2. **HTML:** Web page structure
3. **CSS:** Styling web pages
4. **Flask:** Web framework for routing, rendering templates, and handling requests
5. **Flask-SQLAlchemy:** ORM for database interactions
6. **Jinja2:** Template engine for dynamic HTML rendering
7. **SQLite:** Database engine for data storage

Database Schema Design:

User Table:

1. id as the primary key
2. username (unique and required)
3. password (required, stored in hashed form)
4. email (unique and required)
5. role (required: admin, sponsor, or influencer)
6. is_active (boolean, default True)

Sponsor Table:

1. id as the primary key
2. user_id as a foreign key linking to the User table
3. company_name
4. industry
5. budget

6. description

Influencer Table:

1. id as the primary key
2. user_id as a foreign key linking to the User table
3. category
4. niche
5. reach
6. bio
7. is_flagged (boolean, default False)

Campaign Table:

1. id as the primary key
2. sponsor_id as a foreign key linking to the Sponsor table
3. title (required)
4. description
5. budget
6. category
7. status (required: active, completed, flagged)
8. start_date
9. end_date
10. is_public (boolean, default True)

AdRequest Table:

1. id as the primary key
2. campaign_id as a foreign key linking to the Campaign table
3. influencer_id as a foreign key linking to the Influencer table
4. status (required: pending, accepted, rejected, negotiated)
5. details
6. negotiation_terms
7. communication_log
8. created_at (timestamp)
9. updated_at (timestamp)

Architecture and Features:

The application follows the MVC (Model-View-Controller) structure:

1. Models: Defined in models.py, representing database tables and relationships.
2. Views: HTML templates located in the templates folder, styled with CSS.
3. Controllers: Routes and logic in app.py, handling user requests and data flow.

Video Demonstration: MAD1 Demo