Project Name: Influencer Engagement & Sponsorship Coordination Platform - V2

Author:

Kaushal P Prajapati

Roll No: **22f2001073**

Email: 22f2001073@ds.study.iitm.ac.in

Course: **Modern Application Development II**

Description:

The **Influencer Engagement & Sponsorship Coordination Platform** is designed to create a dynamic ecosystem that connects sponsors and influencers, facilitating a seamless process for advertising products and services. This platform serves as a bridge between companies looking to promote their offerings and influencers seeking to monetize their social media presence.

Technologies Used:

The **Influencer Engagement & Sponsorship Coordination Platform** leverages a diverse set of technologies to ensure a robust, efficient, and user-friendly application. The following technologies were employed:

- **Vue CLI**: Utilized for developing the front-end user interface, enabling a responsive and interactive design that enhances user experience.
- **Bootstrap**: Implemented for styling and aesthetic purposes, providing a clean and modern look to the application.
- **Flask-CORS**: Facilitates the management of cross-origin requests, ensuring smooth interactions between the front end and back end from different domains.
- **Flask-RESTful**: Employed to develop a RESTful API for the application, allowing for effective communication between the client and server.
- **Jinja2**: Used for rendering HTML templates, particularly in the context of sending emails, ensuring a dynamic content delivery.
- **SQLite**: Chosen for data storage, providing a lightweight and reliable database solution for the application.
- **Flask-SQL Alchemy**: Utilized as an Object-Relational Mapping (ORM) tool, enabling seamless interactions with the SQLite database while simplifying database operations.
- **Flask-Celery**: Implemented to manage asynchronous background jobs on the backend, allowing for efficient task processing without blocking the main application flow.
- **Redis**: Used as an in-memory database to cache API responses, significantly enhancing performance. It also serves as a message broker for managing task queues with Celery.

- **Flask-Caching**: Employed to cache API outputs, further optimizing performance and reducing load times for frequently accessed data.

API Design:

The project includes APIs for managing books and sections. It allows creating, updating, and deleting records using RESTful endpoints.

 User Authentication and Management

- POST /api/sponsor-register - Sponsor Registration
- POST /api/influencer-register – Influencer Registration
- POST /api/login – Admin/Sponsor/Influencer Role based Login

 Admin Management

- GET /api/admin/search – Search Sponsors, Influencers, Campaigns
- POST /api/approve-sponsor/<int: sponsor_id>' - Approve Sponsor
- GET /api/admin/sponsors – List of all the sponsors
- GET /api/admin/influencers – List of all the influencers
- GET /api/admin/campaigns – List of all the campaigns

 Campaign Management (for Sponsors)

- POST /api/campaigns - Create Campaign
- GET /api/campaigns - Get Campaigns
- POST /api/campaigns/<int: campaign_id>/review-influencer' – Review Influencer for particular campaign
- PUT /api/edit-campaign/<campaign_id> - Update/Edit Campaign details

 Influencer Management

- GET /api/influencers - Get All Influencers
- GET /api/influencer/<id> - Get Influencer Details
- PUT /api/influencer/edit-profile/<id> - Update influencer profile details
- GET /api/influencer-details/<id> - Get influencer profile details
- POST /api/influencer/campaign-details/<campaign_id> - Provide review to particular campaigns

 Ad Request Management (for Influencers)

- GET /api/influencer/ad-requests - Get Ad Requests for Influencer
- PUT /api/influencer/ad-requests/<id>/accept - Accept Ad Request
- PUT /api/influencer/ad-requests/<id>/reject - Reject Ad Request
- PUT /api/influencer/ad-requests/<id>/negotiate - Negotiate Ad Request

 Ad Request Management (for Sponsors)

- GET /api/sponsor/ad-requests - Get Ad Requests for Sponsor
- PUT /api/sponsor/ad-requests/<id>/accept - Accept Ad Request
- PUT /api/sponsor/ad-requests/<id>/reject - Reject Ad Request
- PUT /api/sponsor/ad-requests/<id>/negotiate - Negotiate Ad Request

 Search Functionality

- GET /api/search/influencers - Search for Influencers
- GET /api/search/campaigns - Search for Campaigns
- GET /api/search/sponsors - Search for sponsors

 Dashboard Statistics

- GET /api/sponsor-stats - Get Statistics for sponsors
- GET /api/admin-stats – Get Statistics for Admin

Architecture:

- /application/: This directory houses all server-side code necessary for the operation of the application. It includes configuration files, models for database interactions, route definitions for handling API requests, and utility functions for repetitive tasks.
- models.py: Contains the SQLAlchemy models that define the database schema, encapsulating the structure of various entities such as sponsors, campaigns, and ad requests.
- view.py: Defines the API endpoints for the application, facilitating interaction between the client and server.
- /background_tasks/: Contains background tasks managed by Celery, such as sending reminders and generating reports for sponsors.
- /utils/: A collection of utility functions, including those for sending emails, enhancing code reusability and maintainability.
- /templates/: This folder holds HTML templates utilized for rendering views and emails, ensuring a separation of design from logic.
- /frontend/: This directory is dedicated to the Vue.js frontend application. It contains the source code, public assets, and configuration files required for the frontend development.
- main.js: The entry point for the Vue.js application, initiating the app and its components.
- /components/: Contains reusable Vue components that are utilized throughout the application, promoting modular design.
- requirements.txt: Lists the Python dependencies required for the backend, ensuring that the development environment can be easily replicated.
- main.py: Serves as the main entry point for launching the Flask application, initiating the server and making the application accessible.

Features:

The **IESCP** platform offers a comprehensive suite of features designed to facilitate seamless interaction between sponsors and influencers. Below are the key features of the application:

1. User Roles and Authentication

- **Role-Based Access Control (RBAC)**: Supports three distinct user roles—Admin, Sponsor, and Influencer—each with specific access privileges and functionalities.
- **Secure Login and Registration**: Provides a secure login and registration process for all users, ensuring data protection and user authentication.

2. Admin Dashboard

- **User and Campaign Monitoring**: Enables admins to monitor all users and campaigns, ensuring compliance and facilitating intervention if necessary.

- **Statistics Overview**: Displays key statistics such as active users, campaign statuses, and flagged content for easy tracking of platform activity.
- **Sponsor Approval Workflow**: Automatically routes new sponsor sign-ups to the admin for approval, ensuring a controlled onboarding process.

3. Campaign Management (for Sponsors)

- **Create and Manage Campaigns**: Allows sponsors to create new campaigns, categorize them by niche, and manage existing campaigns, including updating details such as start/end dates and budgets.
- **Visibility Settings**: Offers options to set campaigns as public or private, controlling access to ad requests.

4. Ad Request Management (for Sponsors)

- **Create and Edit Ad Requests**: Sponsors can generate ad requests tied to specific campaigns, specifying requirements, payment amounts, and influencer targets.
- **Track Ad Request Status**: Provides the ability to view and manage the status of ad requests (Pending, Accepted, Rejected) for effective campaign oversight.

5. Influencer Interaction

- **Ad Request Notifications**: Influencers receive notifications for new ad requests, allowing for timely responses.
- **Accept/Reject and Negotiate Ad Requests**: Influencers can accept or reject ad requests and negotiate payment terms, ensuring fair compensation for their services.

6. Search Functionality

- **Search for Influencers**: Sponsors can search for relevant influencers based on niche, reach, and follower count, streamlining the ad request process.
- **Search for Public Campaigns**: Influencers can explore available public campaigns based on their interests, increasing opportunities for collaboration.
- **Search for Admin**: Admin can search for all the sponsors, influencers and explore available public campaigns.

7. Backend Jobs

- **Daily Reminders**: Scheduled reminders sent to influencers via Google Chat, SMS, or email to prompt them to check for pending ad requests.
- **Monthly Activity Reports**: Automated generation and emailing of monthly reports to sponsors detailing campaign performance, including metrics such as advertisements completed, budget usage.
- **CSV Export of Campaign Details**: Allows sponsors to download their campaign details in CSV format, facilitating offline record-keeping and analysis.

8. Performance and Caching

- **API Caching**: Implements caching mechanisms to enhance API performance and reduce response times for frequently accessed data.
- **Efficient Data Retrieval**: Utilizes Redis for in-memory data storage and caching, ensuring rapid access to critical information.

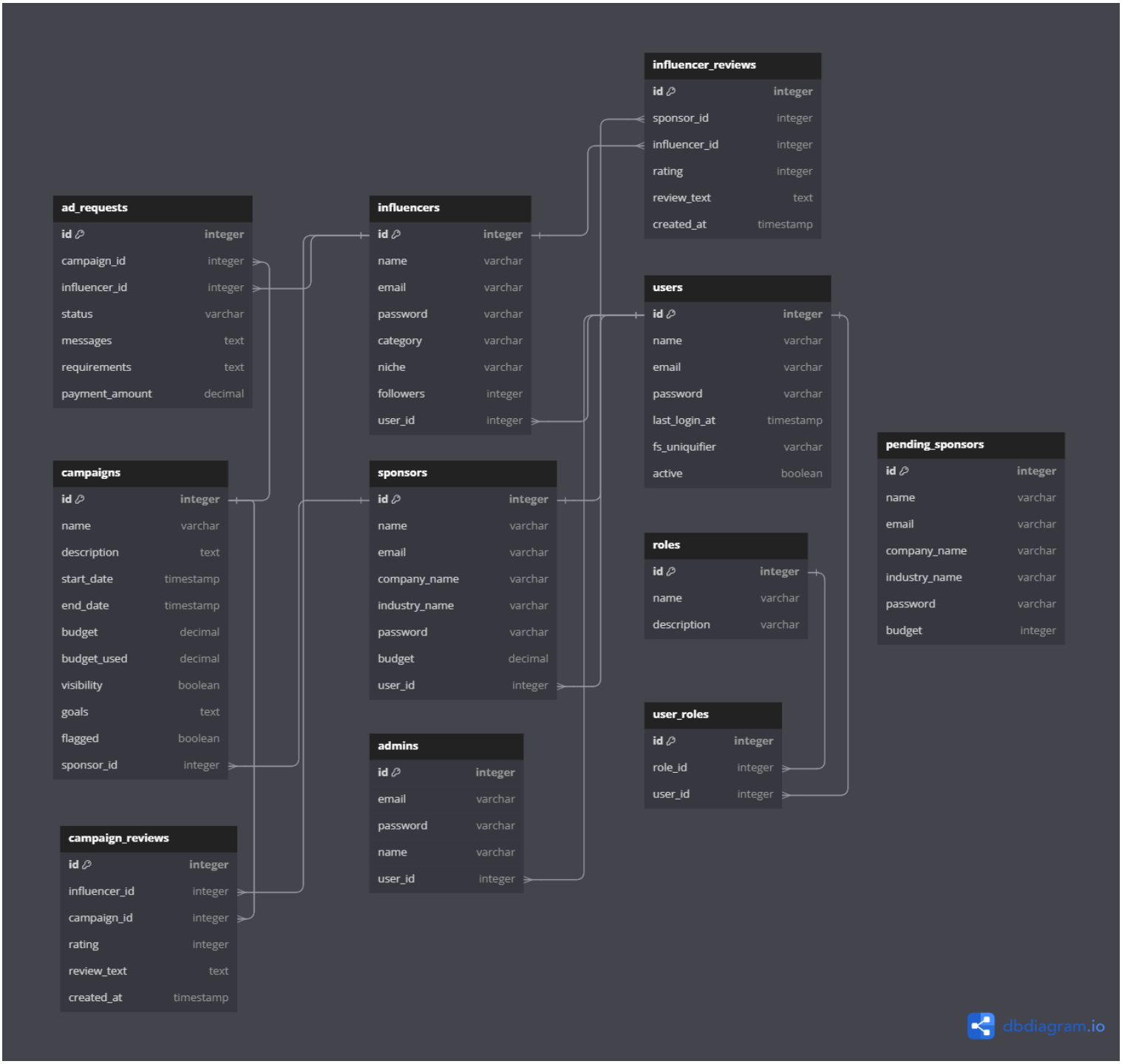
9. User-Friendly Interface

- **Responsive Design**: Ensures a consistent and user-friendly experience across devices, from desktops to mobile phones.
- **Bootstrap Integration**: Utilizes Bootstrap for aesthetically pleasing and intuitive UI components, enhancing user engagement.

10. Robust Reporting and Analytics

- **Dynamic Reporting Features**: Generates reports and statistics on user activity, campaign performance, and ad request outcomes, aiding decision-making for sponsors and influencers alike.

Database Schema Design:



Video:

https://drive.google.com/file/d/1dc2dRrNCJ3SFdg5V6pAJqHWc_B5a2Fj5/view?usp=drive_link