MODERN APPLICATION DEVELOPMENT I

SPONSOR COORDINATION PLATFORM

BY RISHABH INDORIA

Student Details

Name: Rishabh Indoria

Roll No: 21F3001823

Email: 21f3001823@ds.study.iitm.ac.in

About Me: My passion lies in Data Science. I'm captivated by the world of data and its endless possibilities, striving to unlock insights that drive meaningful decisions. Additionally, my curiosity extends to the realm of Artificial Intelligence, where I'm dedicated to pushing the boundaries of what's possible with intelligent systems.

Introduction

The project aims to provide a platform where sponsors and influencers can coordinate with each other. Sponsors can find influencers to promote or advertise their product, and Influencers can choose who to collaborate with and receive monetary compensation for the same.

Approach

For the project, I started by building the backend. Since, I had previous experience with Flask, I was able to make a basic functional app quickly. As I had a bit of time, I learnt Flask-Login for user authentication. Flask-Login, bcrypt and HTML5 validation were used for ensuring safe and secure login functionality. As I have worked majorly on Jupyter notebooks, I was used to coding everything in a single file, but as the application grew the file became very large which necessitated a more structural approach, where I separated CRUD functionality of each model into its own file for better organization and

maintainability. I also explored pagination from Flask-sqlalchemy which allows limiting the number of results shown in the page, this made sure that the user is not overwhelmed with a huge number of campaigns or ad requests.

For the front end, Jinja was used for HTML templating, while Bootstrap was used for styling. Since, I had never used bootstrap before, this is where most of my time was spent. Trying to find different ways to display all the details and forms. I ended up choosing cards for displaying all the details and modals to display all the forms, I also used various alerts and buttons to give the application a more lively feel.

I have also implemented different graphs using matplotlib, where I convert the images generated by matplotlib to a base64 encoded string which is displayed by the HTML file.

Frameworks and Libraries used

- Flask: Backend framework used for building the backend
- Flask-sqlalchemy: Part of Flask framework used to easily build and manage SQL tables
- Flask-Login: Part of Flask framework used to authenticate and validate users
- Flask-Bcrypt: Part of Flask framework used for password hashing
- datetime: Used for handling date and time operations
- re: Regular Expression library used for pattern matching in form validation.
- dotenv: Used for easy loading of variables stored in .env file
- os: Used for accessing variables loaded using dotenv
- HTML: Frontend technology for displaying user interface
- Jinja: A Template engine used for providing pythonlike syntax in HTML
- Bootstrap: Frontend framework used for styling
- Matplotlib: Used for making graphs
- faker: Python library that generates fake data

ER Diagram


flagged
id integer
type varchar
type_id integer
reason varchar
created_on date
campaign
id integer
sponsor_id integer
name varchar
description varchar
goals varchar
category varchar
budget float
remaining float
start_date date
end_date date
created_on date
user
id integer
role varchar
name varchar
email varchar
password varchar
industry varchar
reach float
created_on date
ad_request
id integer
influencer_id integer
campaign_id integer
messages varchar
requirements varchar
payment_amount float
status varchar
negotiate boolean
accepted_on date
rejected_on date
created_on date
dbdiagram.io

API Endpoints

GET /api/user

- Query Parameters: id or email

- Responses: User details or error message.

POST /api/user

- Form Parameters: name, email, password, role, industry, reach, category

- Responses: User created message or error message.

PUT /api/user

- Form Parameters: email, password, name, change_password, industry, reach, category

- Responses: User updated message or error message.

DELETE /api/user

- Query Parameters: email, password, delete_email, id

- Responses: User deleted message or error message.

GET /api/users

- Responses: List of users.

GET /api/campaign

- Query Parameters: id

- Responses: Campaign details or error message.

POST /api/campaign

- Form Parameters: email, password, name, description, start_date, end_date, category, budget, visibility, goals

- Responses: Campaign created message or error message.

PUT /api/campaign

- Form Parameters: email, password, id, name, description, start_date, end_date, category, budget, visibility, goals

- Responses: Campaign updated message or error message.

DELETE /api/campaign

- Form Parameters: email, password, id

- Responses: Campaign deleted message or error message.

GET /api/campaigns

- Responses: List of campaigns.

GET /api/ad_requests

- Responses: List of ad requests.

Presentation video

https://drive.google.com/file/d/13z1eo89HrwxTkzTo9LWP5xKdlEL77abJ/view?usp=sharing