EvaDB Project 2 Report

Name: Chi-Ting Liu, GTID: 903925209

Github Link: evaDB_faceDate

Topic: FaceDate, The Revolutionary Dating App for You!

Abstract

It's a common observation that couples often resemble each other. This phenomenon sparks discussions about our inclination to be drawn to those who share similarities with us. Research studies [1] delve into this intriguing aspect of attraction. Consequently, I **build a dating app that suggests potential matches based on similarities**.

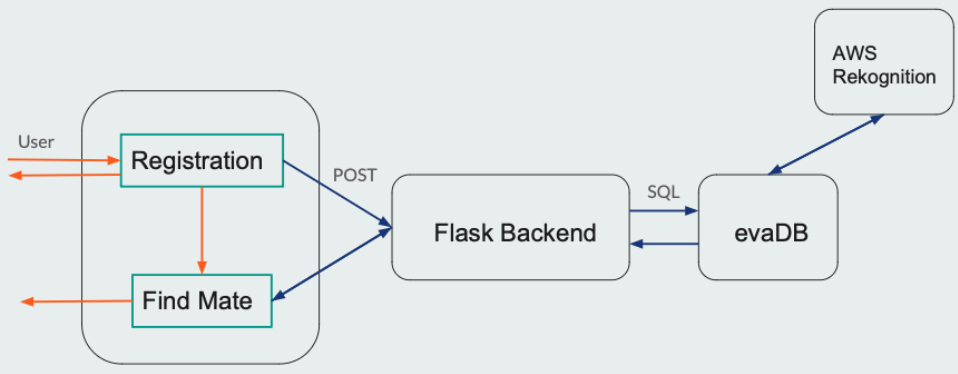
In my last project, I integrated evaDB with AWS Rekognition services, augmenting this database with robust computer vision capabilities. The seamless SQL interface of evaDB coupled with image search capabilities from AWS Rekognition sets the ideal foundation for constructing this dating app.

To achieve this, I explore the REST API of evaDB. I chose React for the frontend framework, Flask for the backend, and leveraged evaDB as the core database to develop the prototype of FaceDate—a dating app centered around identifying similarities.

Deliverable

1. Prototype of FaceDate
2. A small bug in REST API of evaDB

Big Picture



When users visit our app, they will first be asked to register. Then we will store their pictures into our database. After registration, he/she can click the find mate button, then the front-end will render two results in database that looks most similar to them.

Sample Input / Output

For this demo, let's assume our database already contains profiles of 4 individuals. I'll illustrate the process of a new user registering and finding a potential match."

At first, the database has the following information:

Chou
chou@gmail.com

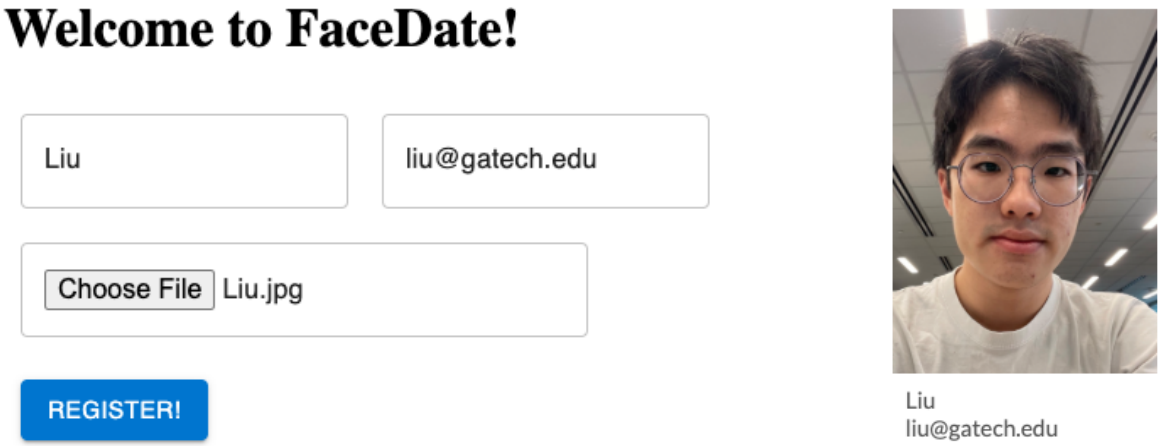
Gal
gal@gmail.com

Chen
chen@gmail.com

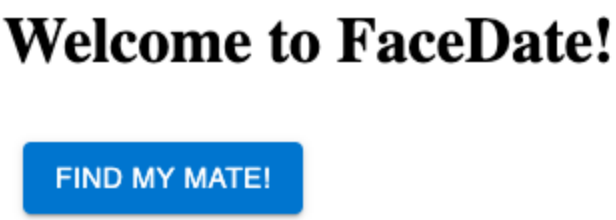
Wang
wang@gmail.com

_row_id	name	gmail	filename
1	chou	chou@gmail.com	files/Chou.png
2	gal	gal@gmail.com	files/Gal.png
3	chen	chen@gmail.com	files/Chen.jpg
4	Wang	wang@gmail.com	files/Wang.png

Then, I (Liu) register through the registration page:



After registration, I can see the “Find My Mate” button as following:



Click on the button, then I can see the top 2 result!

	data	gmail	name
[[[141, 146, 114], [160, 165, 133], [185, 191,...		chen@gmail.com	chen
[[[173, 180, 178], [172, 180, 178], [173, 180,...		gal@gmail.com	gal

Welcome to FaceDate!

FIND MY MATE!

Your Mate!

Name: chen

Mail: chen@gmail.com

Name: gal

Mail: gal@gmail.com

(Whew… Chen is my girlfriend. It's nice this app didn't betray me lol)

Small bug in REST API

- In line 70 of `evadb/apps/rest/run_evadb.py`, directly saving the results into folder `app.config["UPLOAD_FOLDER"]` will results in an error.
- We can simply use `os.path.exists` to fix this issue.
- Since this is just a very subtle issue, I just leave it here and did not open a PR.

Implementation Detail

I set up my server at 127.0.0.1:8803, then I run my client app on localhost:3001

I set up 2 tables, "userDemo" is for user information, "imageDemo" is for store the file.

- userDemo
  - Columns: id, name, gmail, filename
- imageDemo
  - Columns: name, data

I built 2 API for this prototype, the detail is as following

1. Registration
   - store the uploaded file into `files/` folder
   - Load the image into `imageDemo`
   - Store the user info into `userDemo`

   It can be done easily through the SQL instruction provided by evaDB
2. Mate
   - Given the user's name, find the top 2 similar faces in the `imageDemo`

   It can be done easily through the SQL instruction provided by evaDB:

```
SELECT i.data, u.gmail, u.name
FROM userDemo AS u
JOIN (
    SELECT name, data, AWSCompareFaces(data, Open(\'{filename}\'))
    FROM imageDemo
    WHERE name != \'{filename}\'
    ORDER BY similarity DESC
    LIMIT 2
) as i
ON u.filename=i.name;
```

Challenge

1. How did evaDB and flask communicate with each other. (marshaling/unmarshaling data)
2. How to store the file correctly at evaDB.
3. How to render the image at client.
4. How to search the image efficiently in evaDB.

Lesson Learned

1. How to design the API and database
    - I've chosen to separate user information from files. Given that files tend to be large, the column pertaining to these files is optional unless there's a need to retrieve the actual image.
2. EvaDB is a very powerful AI-powered app
    - With the help of evaDB and AWS Rekognition function, I can develop this app without any knowledge of underlying CV function.
3. REST API communication
    - It took me some time to learn Flask and React online, which is pretty fun actually.

Future Work

- CREATE INDEX to optimize image search

Reference

1. Objectively measured facial traits predict in-person evaluations of facial attractiveness and prosociality in speed-dating partners