Cracow, 17.01.2024

Faceauth.

The project is carried out as part of the course: Python in the enterprise.

Authors: Paweł Poręba, Zuzanna Szczelina, Aleksandra Chenczke

INTRODUCTION

The project aimed to create an authorization system using facial recognition. The goal was to establish three foundations that evolved over time. The first was to create an authorization model, followed by the development of a frontend where user-entered data would be collected and sent to the backend for processing.

PROJECT STAGES

To coordinate our work, we regularly organized meetings where the team leader (Paweł) assigned specific tasks. During these meetings, we discussed current issues and consulted any problems that arise. Additionally, we met with the lecturer every week to report on our progress.
The first meeting took place remotely. During this meeting, the outline of the entire project was created, and Paweł set up a repository on GitHub before it. It mainly focused on organizational matters, and we also discussed the vision for the project. Next meetings were held in person, where we continuously updated our to-do list.

Our list of accomplishments is as follows:

1. **Creation of a login feature**: We successfully developed a login feature. It was done with the use of default Django User - but we created custom views for that purpose, and custom templates. During that stage, Zuzanna Szczelina was creating backend views, Aleksandra Chenczke was responsible for the frontend part, and Paweł Poręba was coordinating the whole thing
2. **Docker deployment**: We set up Docker for our project. This was useful, because we didn't have to set up a PostgreSQL database locally - and we needed PostgreSQL for vector extension. In this stage, mainly Paweł Poręba was involved in the development, because he knew what it was about. Later, he explained what he had done, and how it was working, to the rest of the team.
3. **Image capturing functionality**: Implemented the capability to capture images, which are then stored in a vector database. Image was captured using simple web API available in the browser, then transformed to blob type using canvas element (it was the fastest way to do that), and then sent

to the backend, where its hash was created and stored in a vector database, and the file was stored in the filesystem. Here, Aleksandra Chenczke was creating the frontend side, Zuzanna Szczelina was responsible for the backend part, and Paweł Poręba was helping both of them.

4. **Testing and familiarity with PyTorch**: Conducted testing and gained proficiency in the PyTorch framework. We were aiming to understand the logic behind PyTorch API, how it works, why it works like that, how to train models, what is overfitting, how to transform images, how to store and retrieve models, and how to use them in a production environment. All of us were involved in this step.
5. **Styling of the login page**: Designed and styled the login page. Aleksandra Chenczke created the design, wrote styles and JavaScript logic for the pages, and Paweł Poręba reviewed that, corrected some things, and integrated it with Django static files.
6. **AI model for face detection**: Developed an AI model capable of detecting faces. Firstly, we wanted to create our own PyTorch model, but we realized that it would be better to use a previously trained model, check if it is working with our setup, and then retrain it some more. Ultimately we ended up using 2 different neural networks: MTCNN for face detection and cropping of an image, and InceptionResnetV1 for face recognition. We also have 3 different network parameters for the InceptionResnetV1 network: one that was trained previously using VggFace2 dataset, one that was a retrained version of the first one, and one that was only trained using our own dataset. In this step, mainly Paweł Poręba and Zuzanna Szczelina were involved.
7. **Integration of the entire project**: Successfully integrated all components of the project.

BLOCKERS

1. Docker running on Windows. It turned out that now it is necessary to install Windows Subsystem for Linux, to be able to run Docker on Windows. What more, Windows uses a different byte for line ending than Unix systems.
2. Lack of experience with neural networks. We had to learn it from the beginnings

NEURAL NETWORK SCORES

Table 1. Scores of 2 different neural networks after 6 epochs of training

	Loss	**Accuracy**
Fine tuned model	0.06	1
Model built from scratch	4	0.4792

After more epochs, the Fine tuned models loss started rising, so we decided that there is no point in training it further. Model built from scratch showed similar results after 12 epochs, but then it still wasn't working on other data.