Developer's Manual
For Easy Attendance

Table of contents:

Introduction

This project is intended for use by professors to simplify the process of taking attendance of a class and check for proxies. This project can be used to take attendance in universities and schools with about 60 students in class. The software allows instructors to upload photos of the class from which the attendance will be automatically marked, thus reducing the instructor’s tedious task of marking attendance.Though there are many applications that can perform face recognition and identification tasks, there are very few which are specifically used for taking attendance in universities or schools.

Easy Attendance

Easy Attendance is a face recognition tool intended for use in classrooms. In essence, this software tool allows the course instructor to upload pictures of the class to take attendance effortlessly.

Getting Started

System requirements

- Latest version of python 2 & django version - 1.10.6
- To install django run:

```
sudo pip2 install -r requirements.txt
```

Note:

We use the default Django "User" model (in django.contrib.auth.models) to store the users. This makes logging in, logging out and authenticating a request before displaying a page easier. We use the is_staff attribute to check if the user is a professor or a student.

Developer's guide

Server setup

- To run the server:

```
cd Proxy
python manage.py runserver
```

- If running the server for the first time, you may have to run migrations:

```
python manage.py makemigrations
python manage.py migrate
```

Then on your browser, type the address(for example `127.0.0.1:8000`) to go to the login page.

Updating the models

- To make sure that changes in models(models are similar to tables for the DB) take place,

```
python manage.py check
python manage.py makemigrations <module_name>
python manage.py migrate
```

- In case of any errors, resolve them.

DB Access

- To access the DB from command line,

```
python manage.py shell
```

- This will open a python command prompt. In that, you can import the models you want and change the DB.

Modules

AUTH Module

AUTH module features implemented

1. If email already present in DB, doesn't create a new row, just says mail resent and re sends the mail.
2. Reg-Ex matching and giving account type(prof or student)
3. In case email does not match regex, or some failure in connection etc. occurs, takes back to signup page and displays the message.
4. Click on link in email, if the code exists, enter the password and sign up process will be finished.
5. Password is stored as hash.
6. If the verification link is clicked again after successful signup, it will say that the URL has already been modified.
7. If a person attempts to signup for an email which already exists as a user, the appropriate message is displayed.

Things that can be improved in future in auth_module:

1. Delete the entry from SignUp DB in 24 hours(or come up with a hack that stores the timestamp and adds to DB only if the timestamps are > 24 hours)
2. Currently, there doesn't seem to be any way to check if the e-mail ID is valid(For ex: cs14b068@smail.iitm.ac.in is not valid, but our module will send the mail and display "Verification mail sent"). This is a possible source of crashing the DB. Since we only do a regex check, a malicious program could sign up many times with different random addresses that match the regex. If possible, this thing should be looked into. We tried out a few tools, but they did not work.

STUD Module

- Includes functions called when a student is logged in.

PROF Module

- Includes functions called when a professor is logged in.

Testing

- In every module, we have included a file named "tests.py". This file can be used to test for the expected behaviour of the functions in that module.
- To run the tests file:

```
"./manage.py test stud_module.tests" from Proxy folder.
```

Helper files

dummy_entries.py

- This file includes code to add some database entries. To run the script:

```
python manage.py shell
```

- In the shell, run:

```
execfile("dummy_entries.py")
```

- If there are errors, manually copy-paste the lines of the code from the file into the shell to add the database entries.

sendEmail.py

- This file includes code to send an email. This is called as a separate process during the signup process. More details(regarding implementation) can be found in the file.

searchFace.py

- This file includes code to detect faces in an image and search for the detected faces in a course. This is also called as a separate process when a professor uploads photos to take attendance. More details can be found in the file.

Miscellaneous

- Common static includes all the static files such as JS and CSS files used across the various HTML pages.
- All photos uploaded by a professor are included in the "media" folder.
- Issues have been created on github to keep track of what things were to be done. Feel free to raise more issues, if needed.
- If you want to run the server on one machine, and access the page from another machine, make sure to have the IP of the server included in the "ALLOWED_HOSTS" in "settings.py".