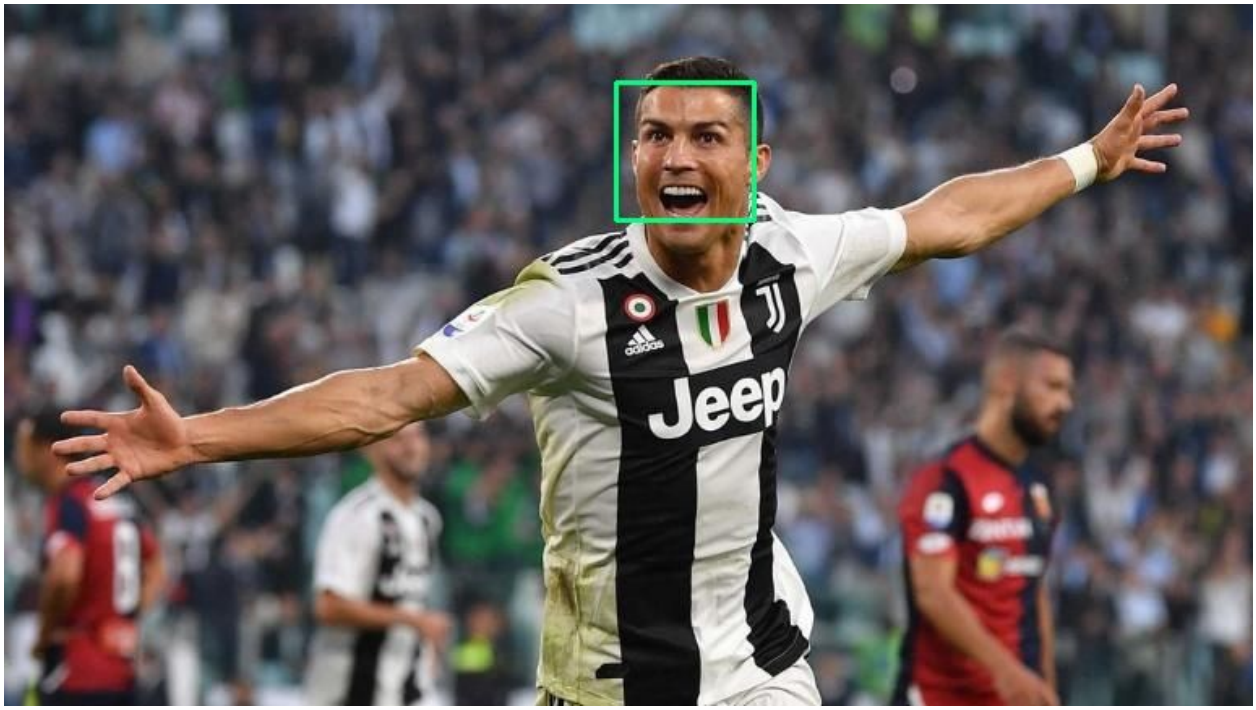
ECE1779: Introduction to Cloud Computing

Winter/Spring 2019

Assignment 1

Web Development



Due Date

February 22

Objective

This assignment will provide you with experience developing a web application using Python and Flask. You will also get experience deploying and running your application on Amazon EC2.

Web Application Description

In this assignment you will develop a simple web application for face detection on photos. Your application should support the following functionality through web:

1. A welcome page with login and register forms. New users can register by providing a username and password. Previously registered users can login using their username and password.
2. A page for uploading a new photo. After a new photo is uploaded, your application should automatically draw a rectangle around human faces in the photo and save the result as well as the original photo.
3. A page that lets logged-in users browse thumbnails of their uploaded photos. Clicking on a photo's thumbnail should produce the original version of the photo as well the photos with rectangles around faces.

Notes for implementation:

1. Use Flask framework.
2. You can use the popular ImageMagick tool to create thumbnails.
3. Use the popular OpenCV library for the face detection and drawing rectangles around human faces. You can find tutorials on how to do so here and here .

Requirements

1. All photos should be stored in the local file system (i.e., on the virtual hard drive of the EC2 instance)
2. Store information about user accounts and the location of photos owned by a user in a relational database. Do **not** store the photos themselves in the database. It is up to you to design the database schema, but make sure that you follow design practices and that your database schema is properly normalized.
3. Follow basic web application security principles. All input should be validated. In addition, user passwords should not be stored in clear text in the database. Instead, store the hash of the password concatenated with a per-user salt value. Details for how to do this are available here.
4. To simplify testing, your application should **also (in addition to your web interface)** include two URLs that makes it possible for the TA to automatically register users and upload photos. The URLs should conform to the following interfaces:

   Register:

```
relative URL = /api/register
method = POST
POST parameter: name = username, type = string
POST parameter: name = password, type = string
```

Upload:

```
relative URL = /api/upload
enctype = multipart/form-data
method = POST
POST parameter: name = username, type = string
POST parameter: name = password, type = string
POST parameter: name = file, type = file
```

5. Sample forms that conform to the above specification are available here and here.
6. A load generator that conforms to the upload interface is available here. You can use the load generator to test your application. **Warning:** the load generator creates a lot of network traffic. To minimize bandwidth charges, you can use the load generator inside EC2 only.

   To run the load generator, run the python script using:

```
python3 gen.py upload_url username password upload_rate(upload_per_second)
files_folder number_of_uploads
```

for example:

```
python3 gen.py http://9.9.9.9:5000/api/upload user pass 1 ./photos/ 100
```

7. Your application should be deployed on a single AWS EC2 instance of type t2.small
8. All code should be properly formatted and documented

Deliverables

We will test your application using your AWS account. For this purpose, your application should be pre-loaded on an EC2 instance. Please suspend the instance to prevent charges from occuring while the TA gets around to grading your submission.

You should write a shell or bash script called **start.sh** that initializes your web application. This script should be put in Desktop folder inside the EC2 instance. Your web application should run on port **5000** and be accessible from outside the instance. So, make sure that you open this port on your EC2 instance. Documentation about how you can open the port can be found here .

Submit the assignment only once per group. Clearly identify the names and student IDs of group members in the documentation.
To submit your project, upload to Quercus a single tar file with the following information:

1. User and developer documentation in a **PDF** file (documentation.pdf). Include a description of how to use your web application as well as the general architecture of your application. Also include a figure describing your database schema. Click here for tips for

how to write documentation. Your documentation will be marked based on how cohesive it is and how well you are able to explain the technical details of your implementation.
2. AWS credentials (credentials.txt). The TA will need these to log into your account. You can create credentials with limited privileges using AWS IAM if you don't want to share your password with the TA; however, make sure that you include permissions to start and stop EC2 instances. Test the credential to make sure they work.
3. Key-pair used to ssh into the EC2 instance (keypair.pem).
4. Anything else you think is needed to understand your code and how it works.

Marking Scheme

- UI/UX: Being able to navigate through all pages easily (using sensible menus and buttons), properly showing the photos **(4 points)**
- Functionality: Face detection functionality, handling exceptions, no crashes or bugs, correct database design, proper photo storage **(7 points)**
- Security: Input validation, implementing sessions, using salts and hashes for passwords, correct mechanism for logging in and registering users **(4 points)**
- Test on load generator: We shall test your web application using the load generator provided. The load generator requires the upload API(explained above) to function. Make sure that your web application works using the load generator **(3 points)**
- Documentation: All code should be properly formatted and documented, explanations about how to log in to your amazon account, how to start the instance and how to run the code should be included. You should also write about the general architecture of your code and how different elements of your code are connected to each other. Don't forget about the database schema or group members **(7 points)**

Resources

Amazon provides free credits for students to experiment with its cloud infrastructure, including EC2. To apply for an educational account go to Amazon Educate.
The following video tutorials show how to :

- Create an EC2 instance
- Connect to an instance using a VNC Client.
- Suspend an instance

To help you get started, an AMI (id: ami-0cb01fcf9cf16705c) is provided with the following software:
Note: the AMI is only available in the N. Virginia AWS Region

- Python 3.5
- PyCharm IDE
- Firefox
- MySQL Server (root password ece1779pass)

- mysql-workbench
- vncserver (password ece1779pass)
- Database and AWS Flask examples covered in lectures and tutorials.
- ImageMagick
- OpenCV libraries
- Python Wand ImageMagick bindings
- Python OpenCV binding (cv2)
- gunicorn
  This is high performance server for Python-based applications. For an example of how to run it, look at the run.sh file inside Desktop/ece1779/databases
- In addition the directory in Desktop/ece1779 contains the following:
    - **databases:** A PyCharm project with all examples from the databases lecture and tutorial
    - **extras:** A PyCharm project with code for transcoding photos and a sample form that conforms to the load testing interface