ECE1779 Assignment 2

# Workers

## Description

Each individual worker application is written in python and html using the flask library and it performs as an online text detector. It automatically draws rectangular boxes around the text on the photos which users uploaded. Users can explore all the photos they uploaded and compare the photos before and after text detection in pairs.

## Prerequisite

A Python 3.7 interpreter is required.

The following python packages are required in order to start the server.

* Click
* PyMySQL
* Flask
* Flask-SQLAlchemy
* Jinja2
* MarkupSafe
* SQLAlchemy
* Werkzeug
* aiofiles
* aiohttp
* async-timeout
* attrs
* chardet
* idna
* imutiles
* itsdangerous
* multidict
* opencv-python
* pip
* setuptools
* yarl
* awscli

## EC2

The worker applications are running on separate EC2 instances. Each instance is given a custom role so that it is allowed to access objects stored in the S3 bucket without specifying an access key and a secret access key. This role is immediately attached to the instance when it is created to make sure each worker instance can read and write objects in the S3 bucket.

Moreover, the worker instances are assigned to a custom security group for two reasons. First, these instances are open to public so that anyone from anywhere can create accounts and upload images. Second, accesses to the database are only granted to instances in such security group.

## Initialization

The individual worker application is initialized throughout executing the h.sh script on the Desktop. This application is designed to run at port 5000. Therefore, please make sure port 5000 is available on the client side.

## Using the application

If users are not log in, the nav bar has a login button as well as a sign-up button. Users can use the log in button to log in and use the sign-up button to sign up.

While user has logged in, nav bar has home, log out, and upload button. Log out button will delete the user information in sessions and let the user log out. Home button is directed to home page at which users can view and upload the images. On the upload page, users can upload images with ‘png’, ‘jpg’, ‘jpeg’ and ‘fig’ extensions. After uploading, web app will detect and use green rectangles to mark the text in the images. Users can view images they uploaded on the view page. By click the ‘view’ button under the images, users are able to see the original image and the processed image at same time.

## API

Tests can be done by sending API requests. You are allowed to create new users and upload images using an existing account.

The register request requires the following interface.

URL: <http://0.0.0.0:5000/api/register>

method = POST

POST parameter: name = username, type = string

POST parameter: name = password, type = string

The upload request requires the following interface.

URL: <http://0.0.0.0:5000/api/upload>

enctype = multipart/form-data

method = POST

POST parameter: name = username, type = string

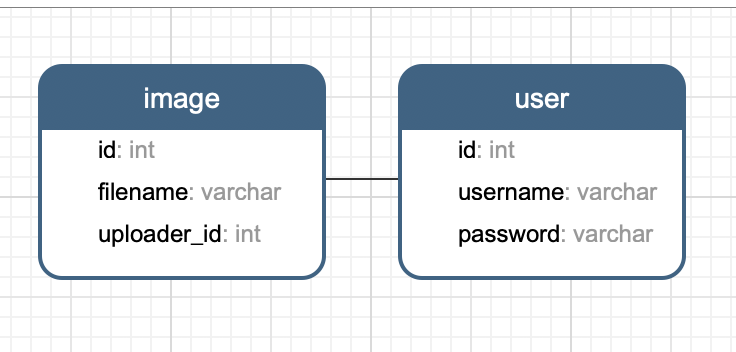
POST parameter: name = password, type = string

POST parameter: name = file, type = file

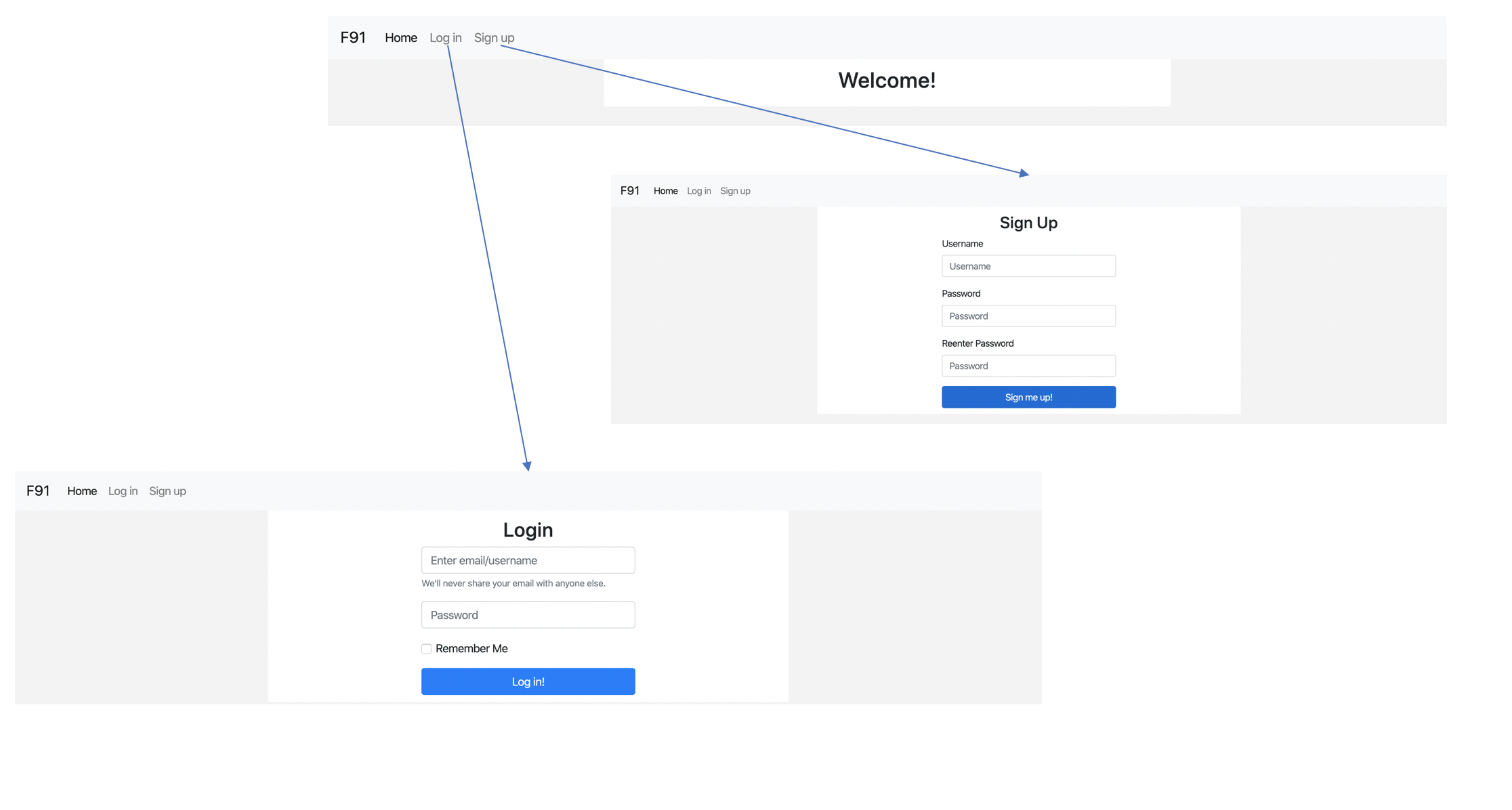
## Database

Our database is powered by the MySQL engine and stored using the AWS RDS service. It can only be accessed by the AWS EC2 instances within a specific security group and all other accesses will be denied.

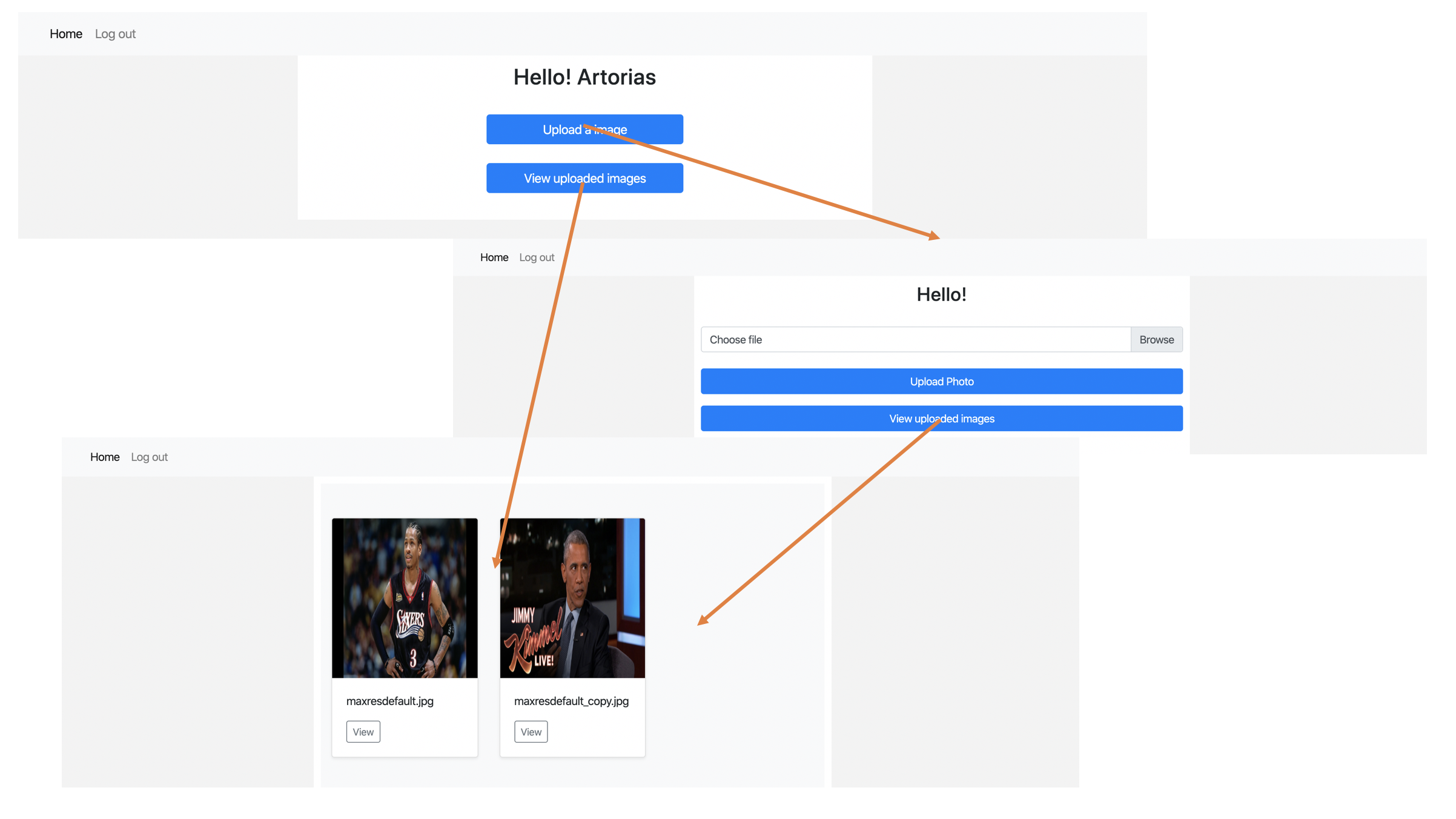
The database consists of two tables, image and user, which are connected to each other by sharing the user id. Hence, each user is only allowed to access the image he or she uploaded. Moreover, since each image is given a unique image id, different images with the same filename will be treated as different images. The detailed database structure is shown below. The uploader\_id in table ‘image’ is refer to id in table ‘user’.



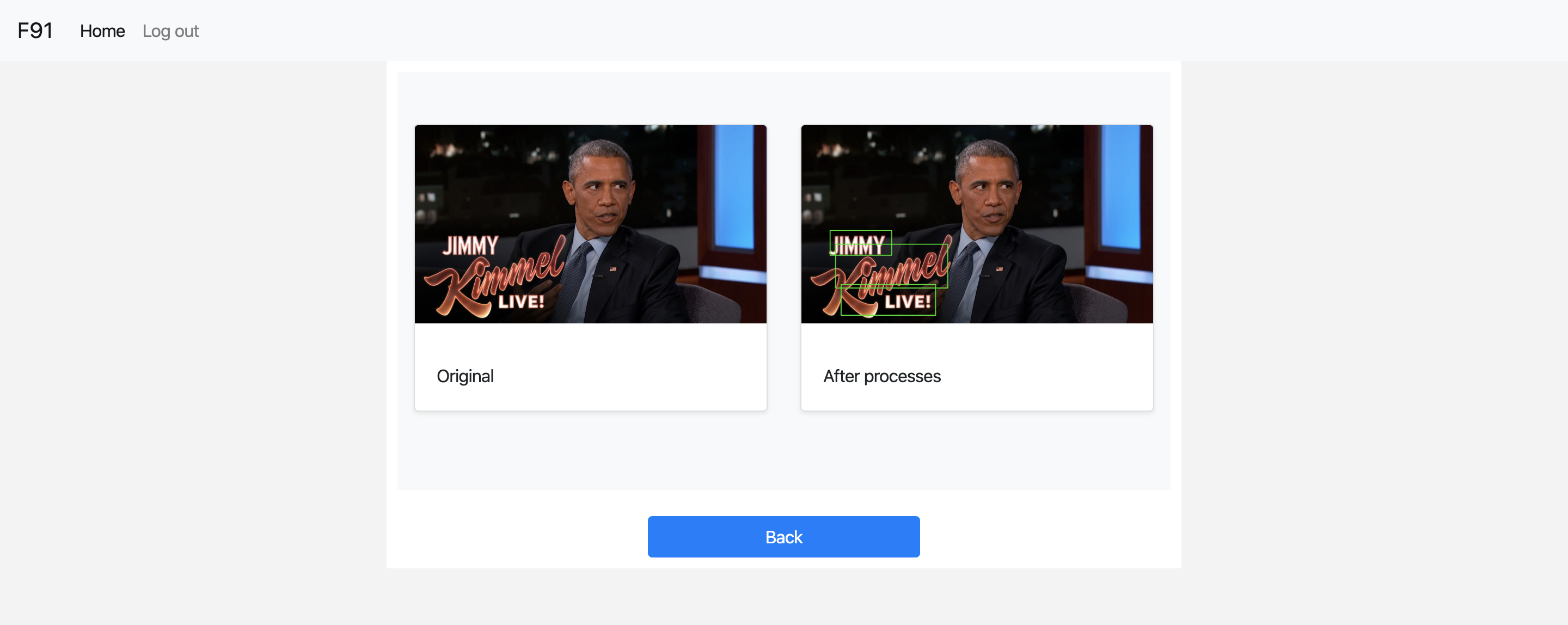
The application structure is shown as follows.



There are two buttons on the main page. The ‘Login’ button will direct the user to the login page while the ‘Sign up’ button will lead the user to the sign-up page. After logging in or creating an account, the user will be directed to the user page.



When clicking the ‘view’ button, users can see the difference between the original photos that they upload and the photos after text detection.



## Code Structure

The body of the application is packed in the “app” directory. When the application starts, “\_\_init\_\_.py” will be executed and basic parameters and settings in order to run the program will be initialized. The core of the application is the “views.py” file. This file is written to handle all the requests received from the clients, such as showing users the photos they uploaded. The structure of the database is defined in “model.py”.

A static directory is created to store all the css files as templates.

“text\_detection.py” contains the function to draw rectangles around texts in the photo. When an image is uploaded, another image with rectangles around texts will also be created and saved to the user’s directory so that he or she can see the difference.

All the html templates are stored in the “template” folder under the “app” directory. This allows us to separate the html files from the python files. Moreover, one html template can be used in several different situations.

## S3

When a user creates an account, a directory with his or her username will be created in the S3 bucket to store the images he or she uploaded later. If the user uses clicks the delete button in the web app, the image under the corresponding directory will be removed.

The session encryption key is stored in a text file in the S3 bucket to make sure that each worker app has the same key.

## Cookies

For security purposes, cookies are used so that we can know the status of the user. When the application starts, a *session* object is created to store the login information of the user. All the data in the *session* object is encrypted using a secret key, which is stored in a text file in the S3 bucket. Therefore, users can stay logged in using any worker instances. Once the user logged in, the username will be stored in *session* for further verification. Users are required to reenter their usernames and passwords every 24 hours for the purpose of safety. They can also manually log out to drop their information in *session*. The application will automatically redirect the user to the login page if it cannot find the user’s corresponding information in *session*.

## File restriction

Users are only allowed to upload images with the extensions mentioned above. A file is verified twice before being posted on the user’s page. First, when the server receives the file, it will check the file’s extension. If the extension is not allowed, the client request will not be accepted. For instance, the server will not let a user upload a file with a pdf extension. Second, the server will check the readability of the file during image processing. If the file is unreadable, it will be rejected by the server. For example, the user can manually change the extension of a file from others to jpg, making it look an image file. However, such file cannot be read by the text detection function and hence these kinds of files are blocked from uploading.

The application only accepts files with sizes less than 10MB. Otherwise the load of the server will be too heavy.