ECE1779 Assignment I
Documentation

Hao Jin (1000303405)
Louis Chen (1000303502)

Table of Contents

1.0 Description

This documentation serves as a general guideline for ECE1779 assignment 1. End deliverable of this assignment is a web server which provides online photo storage service. The web server is preloaded on Amazon's EC2 server. It provides features such as new account registration, login authentication, photo uploading, and face detection in the uploaded photos. Please refer to subsequent sections for prerequisites, installation, server architecture, database architecture, and APIs.

2.0 Prerequisites

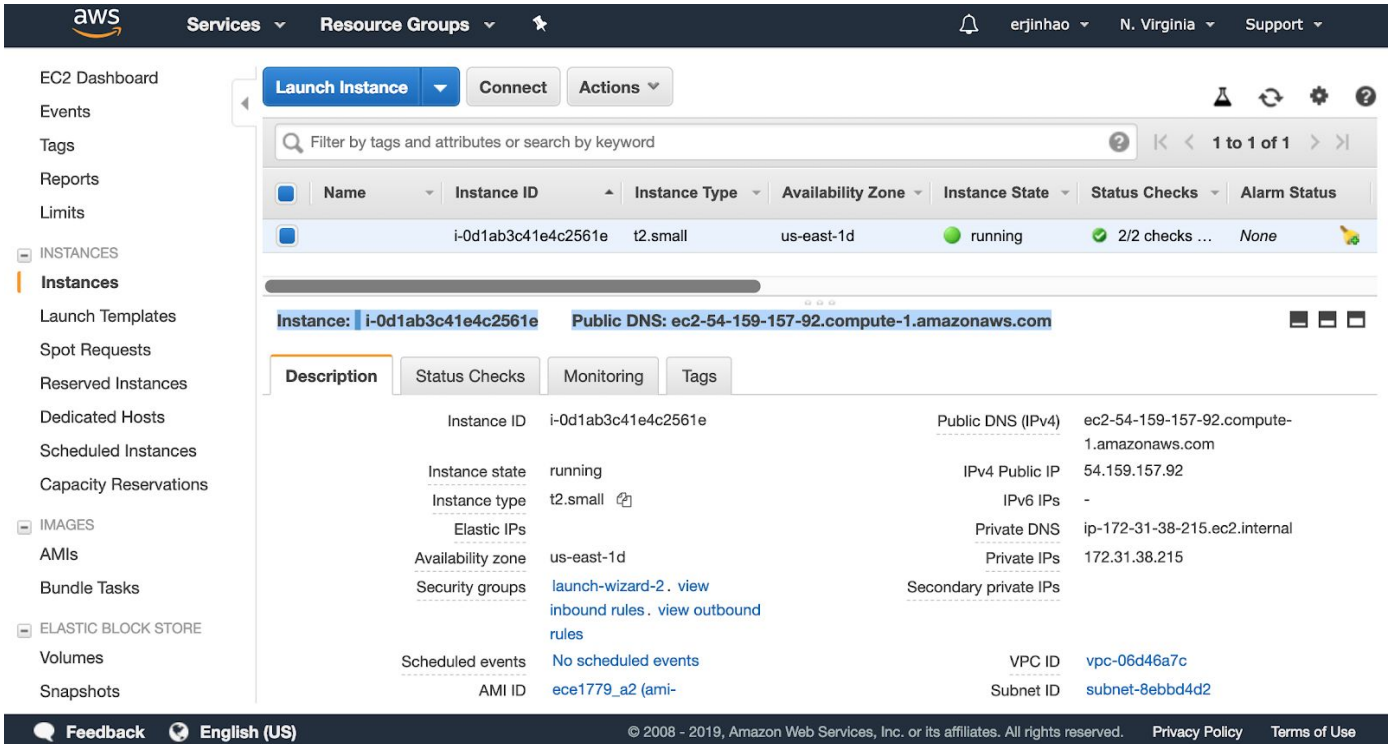
The following table outlines software prerequisites needed to launch the web server.

Interpreter	Python 3.5+ (most preferably Python 3.7)
Libraries	Flask, mysql-connector, OpenCV
Database	MySQL

3.0 Installation

This section shows the step-by-step procedures for launching the web server.

1. Open a browser, navigate to https://aws.amazon.com/.
2. Click “My Account” on the top navigation bar, and choose “AWS Management Console”.
3. Sign in using the email and password provided in the file “credentials.txt” (provided in the zipped package).
4. Under “Services” on the navigation bar, click “EC2”.
5. Under “EC2 Dashboard” on the left, click “Instances”.
6. You should see only 1 EC2 instance is present, and stopped. Now select the instance, click “Actions” button on the top, choose “Instance State”, and click “Start”. When prompted, click “Yes, Start”.
7. Wait for the “Instance State” to turn green (running). Once it turns green, the EC2 instance has been launched and ready to run the web server.
8. Locate the “keypair.pem” file provided in the zipped package. Open a terminal, type “chmod 400 <path_to_keypair.pem>” to set the correct access permissions.
9. Now press “Connect” on the top menu. You should see an example command to ssh into the EC2 instance. In the terminal, type the command in. The command should look similar to the following: “ssh -i <path_to_keypair.pem> ubuntu@ec2-54-159-92.compute-1.amazonaws.com”.
    a. Make sure to specify the correct path to the pem file “keypair.pem” (provided in the zipped package).
    b. Make sure to change “root” to “ubuntu”.
10. Now your terminal should be directed to the EC2 instance’s terminal. Type the following command to launch the web server: “~/Desktop/start.sh”.
11. Your web server is now running. On your local computer, open a browser and type the following url to connect to the web server: “http://ec2-54-159-157-92.compute-1.amazonaws.com:5000”.
    a. Make sure you use the correct DNS, as it changes once the instance is rebooted. Correct DNS can be found in the main EC2 console, shown as follows.
    b. Make sure to stop the web server and the EC2 instance once you are done. In the terminal, type the following command to stop the web server: “~/Desktop/stop.sh”. To stop the EC2 instance, you may take step 6) and select “Stop” instead of “Start”.



4.0 Server Architecture

The following is the architecture for the web server.

- app/
  - static/
    - assets/
    - css/
    - users/
  - templates/
    - image.html
    - login.html
    - profile.html
    - register.html
  - train_file/
    - eye.xml
    - face.xml
  - __init__.py
  - api.py
  - db.py
  - image.py
  - login.py

- ❏ macro.py
- ❏ main.py
- ❏ profile.py
- ❏ register.py

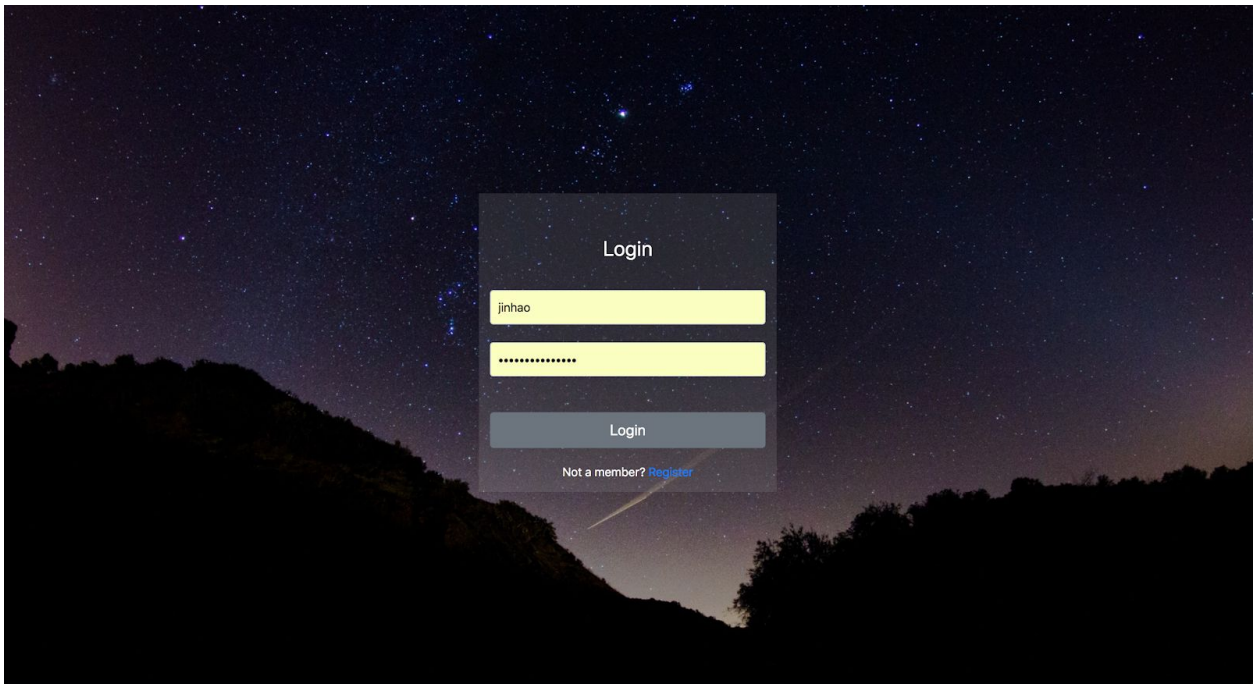
❏ run.py

Static folder contains all static resources for the web server categorized into three folders which are assets, css and users.

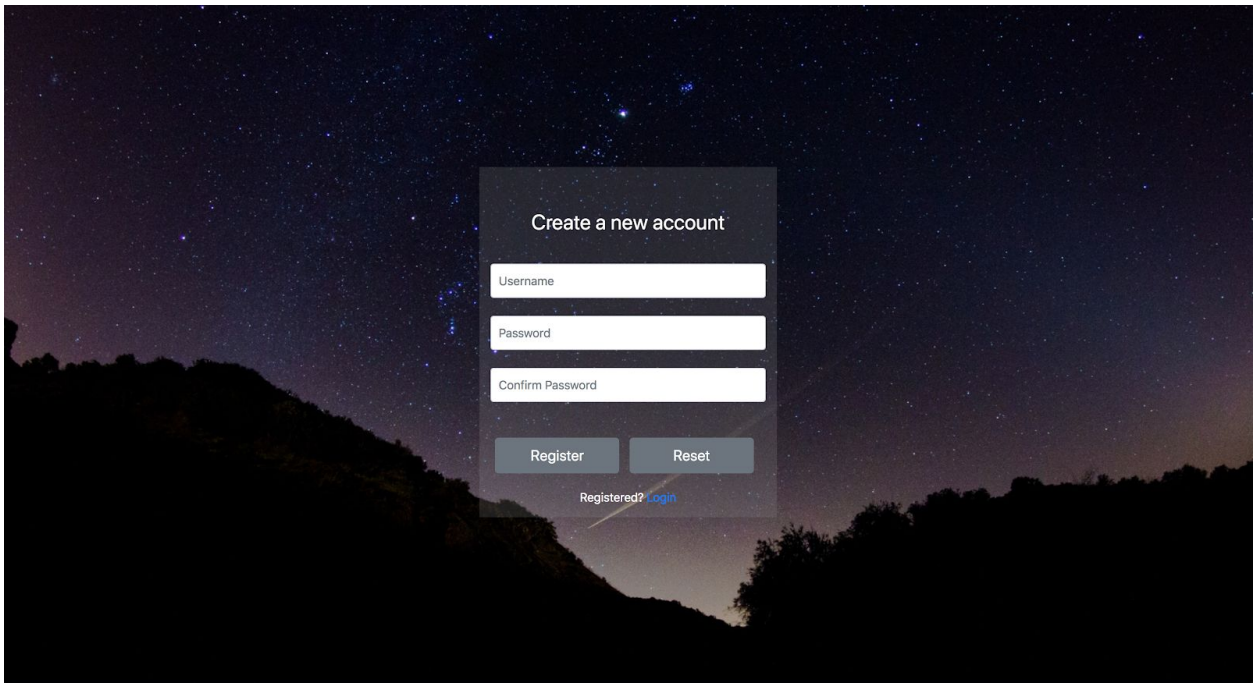
- assets
  - This folder contains all the static images used for front-end UI. Now it contains one background image used in all the pages.
- css
  - This folder has all the CSS files used to decorate the HTML files in the templates. Login.css is linked in login and register pages and profile.css is linked in the profile page. Except for customized CSS file, pages are also linked with bootstrap CSS and font awesome CSS for designing the front-end UI.
- users
  - This folder contains all the images uploaded by the users and corresponding face-detected images. All uploaded images are stored and categorized into specific folders named after the username of uploaders.

Templates folder contains four HTML templates, which are login.html, register.html, profile.html, and image.html. Following describes what users are able to do in these pages.

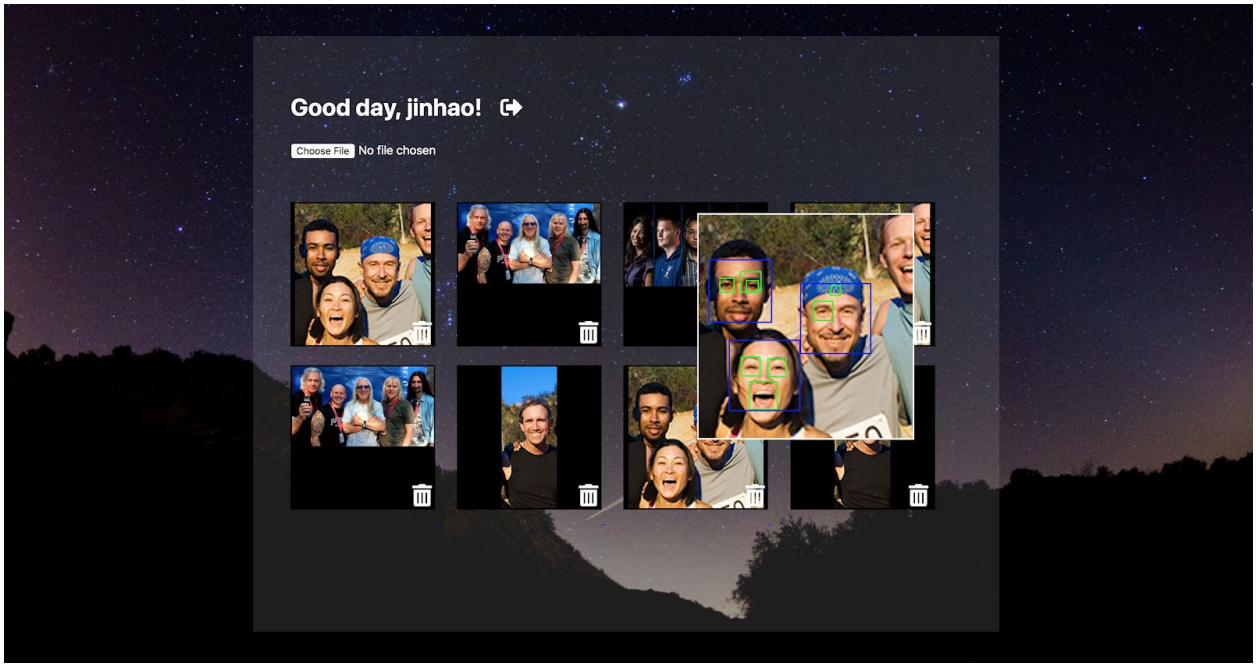
- login.html
  - This template renders a page for users to log in with their credentials if they have registered before. It has two input. One is the username. Another one is the password. If users never registered, there is a button linked to the register page. If users input wrong username or password, a proper message will be displayed in red color.



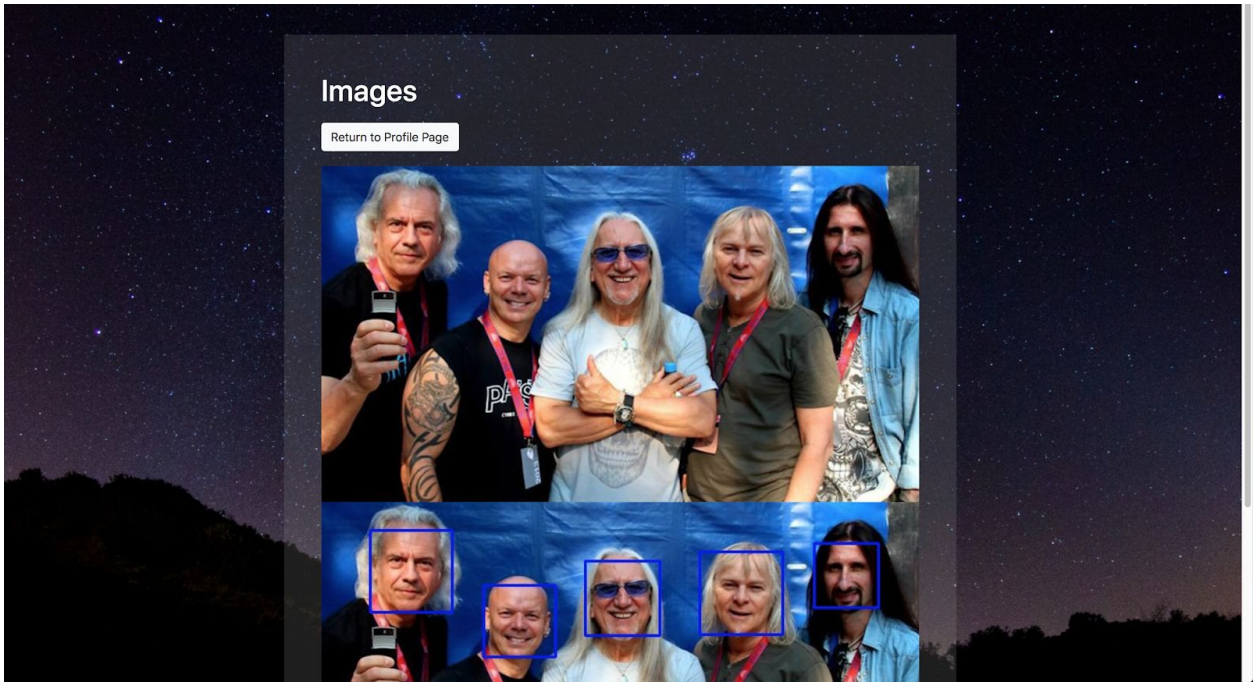
- register.html
    - This template renders a page for users to register a new account. It has three inputs which are username, password, confirm password. If users input a username that has been registered or mistyped the password, an error message will be displayed. There is also a button below these inputs to redirect the URL to the login page.



- profile.html
    - This template renders a profile page that enables users to upload images and also displays thumbnails of the images uploaded by users. It starts with a welcome sentence including username. There is also a logout button beside the welcome sentence for users to sign out. Underneath the welcome sentence, there is a “choose file” button for users to choose images to upload. After users click on the button, a system window will pop up to prompt to select a file to upload. After being selected, the file will be automatically uploaded to the server. A feedback message will also be displayed on the page telling users if the file is uploaded or not. The file will be processed on the server and generated a new file with rectangles drawn around faces. Users can hover on the thumbnail to see the face-detected image. Users can also click on the thumbnail to redirect to the image.html to see images with full resolution. Moreover, there is a button with a trash icon at the corner of images for users to delete the corresponding image.



- Image.html
    - This template renders a page for displaying images in full resolution. After a user clicks on the thumbnail of an image in the profile page, he can view the details in the image page. This template displays two images. One is the original image. Another one is the new image around faces detected by the application. This template also has a button that enables users to the profile page.



Train_file folder contains two training files for face and eye detection, provided by OpenCV.

- eye.xml
    - This training file trains the OpenCV model to detect human eyes.
    - Due to assignment requirements, this file is not used.
- face.xml
    - This training file trains the OpenCV model to detect human faces.

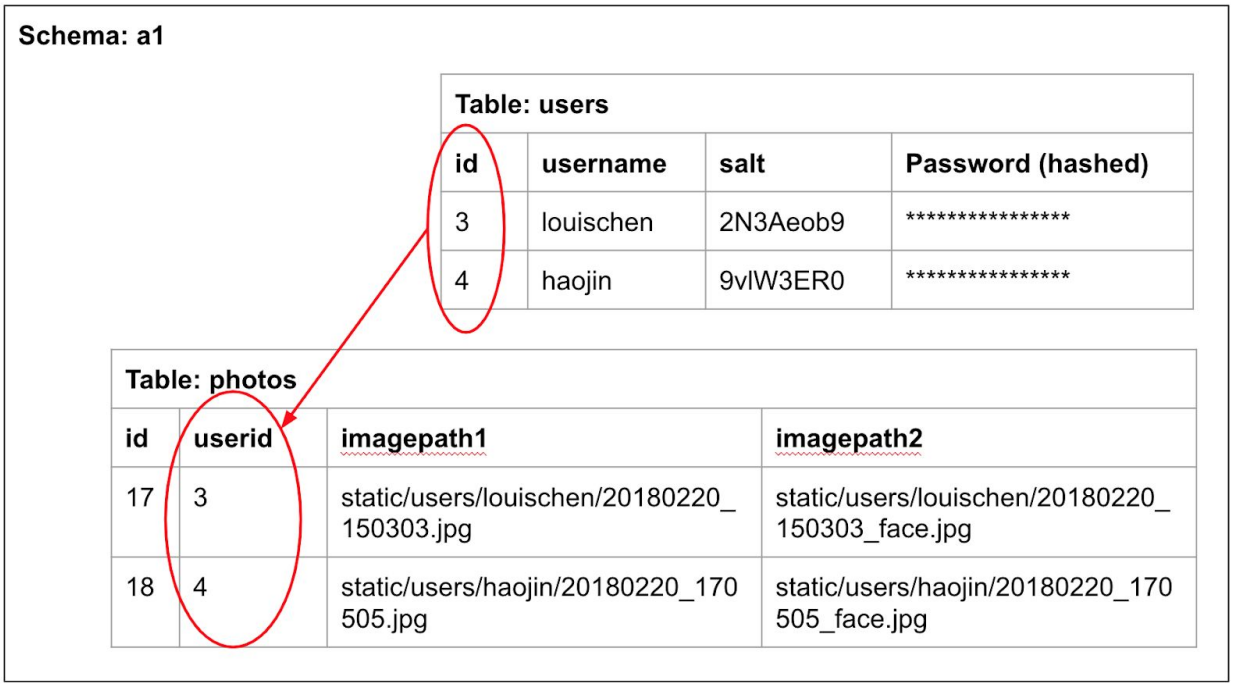
Python files that contribute to the main functionality of the web server are listed below.

- __init__.py
    - Initializes the web server.
    - Imports python files.
- api.py
    - Handles API requests (refer to section 6.0 for API details).
- db.py
    - Contains database configuration parameters (user, password, host, database name).
    - Contains functions to connect/disconnect to database.
    - Contains function to generate a random salt for account registration.
- image.py
    - Handles image URL and receives image paths through URL
    - Renders image.html to show images in full resolution
- login.py

    - Handles user login POST form. It first validates the input strings to check if they have an improper character to avoid SQL input hack. It then queries the database to check if the username is used and if the password is correct. It returns error messages if there is any problem.
    - Redirects to profile page if the user successfully logged in.
- macro.py
    - Stores secret key for Flask session as *secret_key*.
    - Stores valid characters for username as *username_char*
    - Stores valid characters for password as *password_char*.
    - Stores valid characters for salt as *salt_char*.
- main.py
    - Renders the main page of the web server. It checks whether user has previously logged in using Flask session. If so, it renders the page to profile.html. If not, it renders the page to login.html to prompt user to log in.
- profile.py
    - Handles photo upload POST form. It first validates all parameters (username, password, file). If they are not valid, the function will return error message with proper HTTP status code. It then retrieves the photo, performs face detection on it, and makes a copy of the photo with face detection marked in blue rectangles. Finally, it stores the original photo and the photo marked with face detection onto EC2 instance's local disk, under the relative directory "app/static/users/". The photos are named after timestamp (in the format of yymmdd_hhmmss).
    - Handles photo deletion POST form. It first deletes the photo files stored on the EC2 instance and deletes the corresponding row entry in the database (under table *photos*).
    - Handles user logout POST form. It removes username from the Flask session and redirects the browser to main login page.
    - Contains function to retrieve all photo paths associated with currently logged in user.
    - Contains function to retrieve extension of the uploaded image.
- register.py
    - Contains function to render the account registration form (register.html).
    - Handles account registration POST form. It first checks if username, password, and confirm_password contain valid characters as listed in macro.py. It then checks if the username has been registered. If there is no error, it will redirect the user to the profile page. Otherwise, it will display the corresponding error on the webpage in red.
- run.py
    - Serves as top-level python file for the user to launch the web server.

5.0 Database Architecture

The web server uses a relational database to store all users' username, salt, hashed password, and their photo paths. The database is constructed using MySQL. The following figure shows the architecture of the database.



The database contains one schema, *a1*, which internally consists of two tables, *users* and *photos*. Table *users* stores information about user login credentials. It has columns *id* (primary key), *username*, *salt*, and *password* (SHA hashed value of password concatenated with salt). Table *photos* stores paths of the photos uploaded by the user. The paths are relative to web server's main directory, and therefore allows for quick search of photos. Table *photos* consists of columns *id* (primary key), *userid* (foreign key), *imagepath1* (path to original photo), and *imagepath2* (path to photo with face detection). The relationality of the database takes place in between table *users*'s primary *id* and table *photos*'s *userid*.

6.0 APIs

Despite of direct access to the web server via browser, the web server also provides APIs to enable direct URL access. Two features that are accessible via APIs are account registration and photo upload.

The following figure shows the URL and the required parameters for account registration.

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

The following figure shows the URL and the required parameters for photo 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
```

Users may use the command-line tool, curl, to post API requests. Example curl commands for posting account registration and photo upload are as follows.

curl -D “username=<username>&password=<password>” http://127.0.0.1:5000/api/register
curl -F “username=<username>” -F “password=<password>” -F “file=@<path_to_file>” \
http://127.0.0.1:5000/api/upload

If the API requests are handled successfully, users should receive a return message “Success”. If not, users will receive “Error: ” followed by a reason of failure.

7.0 Revision Control

The web server has been revision controlled in GitHub. The following table outlines the information of contributors who have co-worked during the development process.

Contributor	**Email**	**GitHub ID**
Hao Jin	hao.jin@mail.utoronto.ca	erjin
Louis Chen	louis.chen@mail.utoronto.ca	louistjchen

The repository is currently set in private mode. However, it will be accessible to the public at https://github.com/louistjchen/Photo-Web-Server sooner or later.