ECE1779: WEB DEVELOPMENT

INTODUCTION

A simple web application for text detection in images has been created using flask framework. The application has been deployed on an EC2 instance. Through the web, the application lets new users register and allows already registered users login and logout. The page “Album” contains thumbnails of all the images previously uploaded by the user and allows the user to upload new images. When the user uploads a image, the application automatically detects the text in the image and draws rectangles around it. This image, along with the original image and the thumbnail is stored locally on the virtual hard drive of the EC2 instance. When the user wishes to see any previously uploaded image, they can click on the thumbnail present in the album to view the original and the text detected image. The information regarding the users and the location of the images are stored in a database “newdb”.

PREREQUISITES:

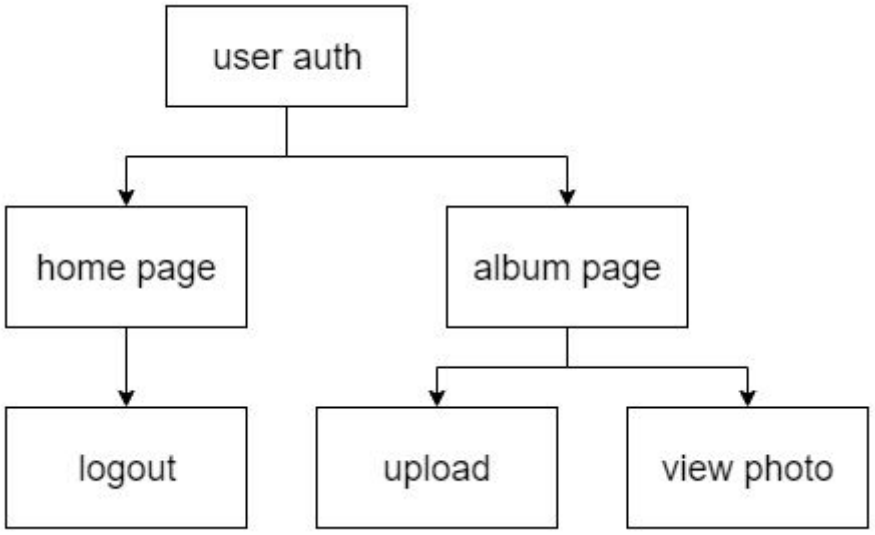
- Python 3.7
- PyCharm IDE
- Web browser
- MySQL server
- MySQL workbench
- mysql_connector
- Flask
- flask_bootstrap
- flask_uploads
- OpenCV-python
- numpy
- boto3
- botocore
- WTForms
- gunicorn

TA TESTING GUIDE:

1. Click here (https://735141600372.signin.aws.amazon.com/console) to console login page.
2. Login with IAM user name and password.
   User name: TA / Password: 123123
3. Go to *Services* and find *EC2*.
4. Choose “ece1779-a1” instance and start it.
5. SSH to the instance with *keypair.pem* as provided.
6. Go to Desktop/ and run the application with the command: ./start.sh

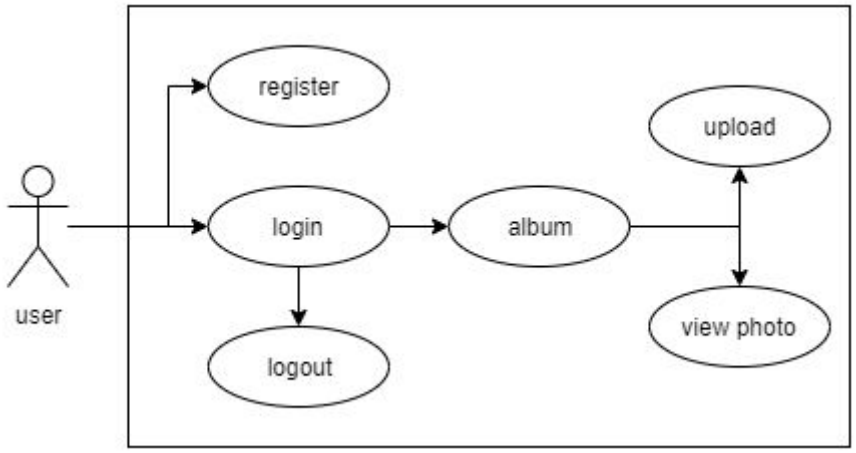
GENERAL ARCHITECTURE:

Project structure:



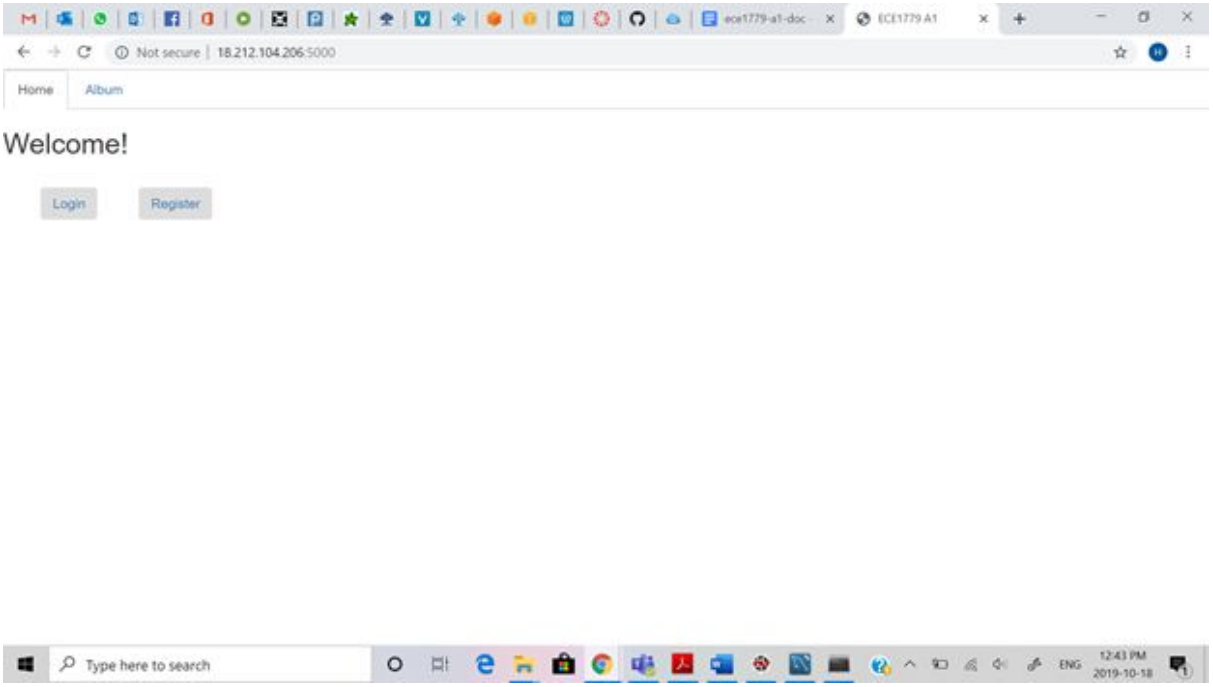
User can sign up own account. Then login to access to own album. In album page, all photos are listed and user is able to upload new photos. After clicking a thumbnail in album, system will redirect to photo page, and origin photo and text-detected photo will be displayed.

Use case diagram:

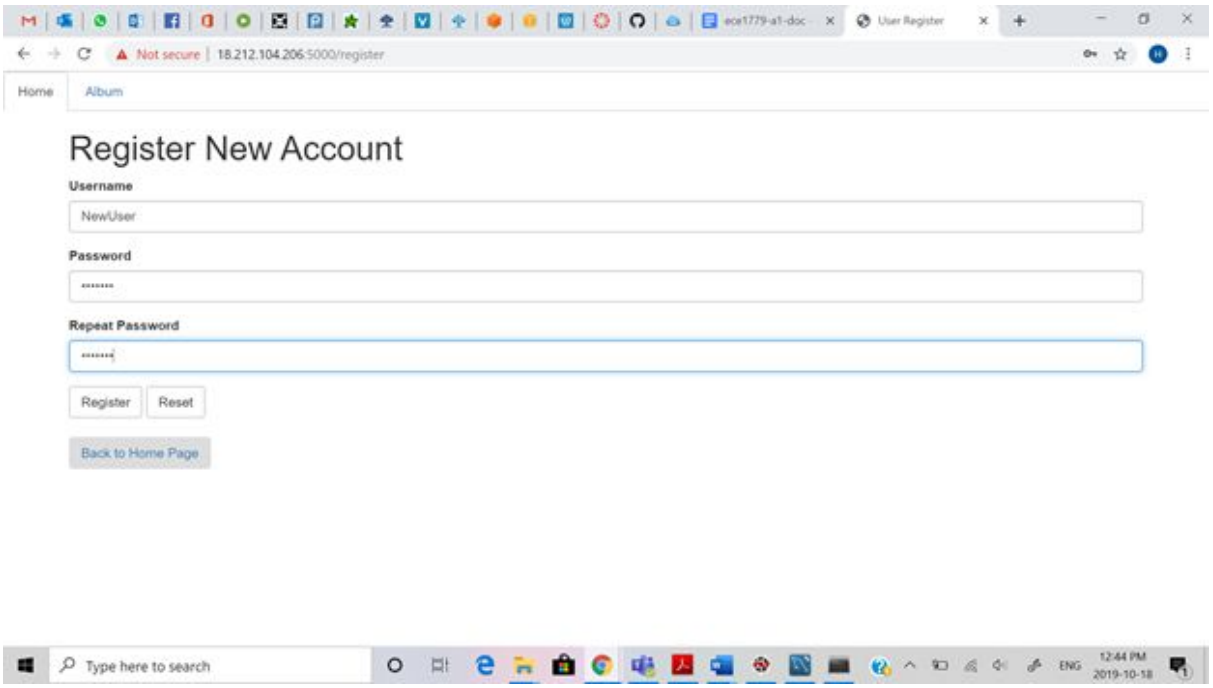


PAGES DISPLAYED:

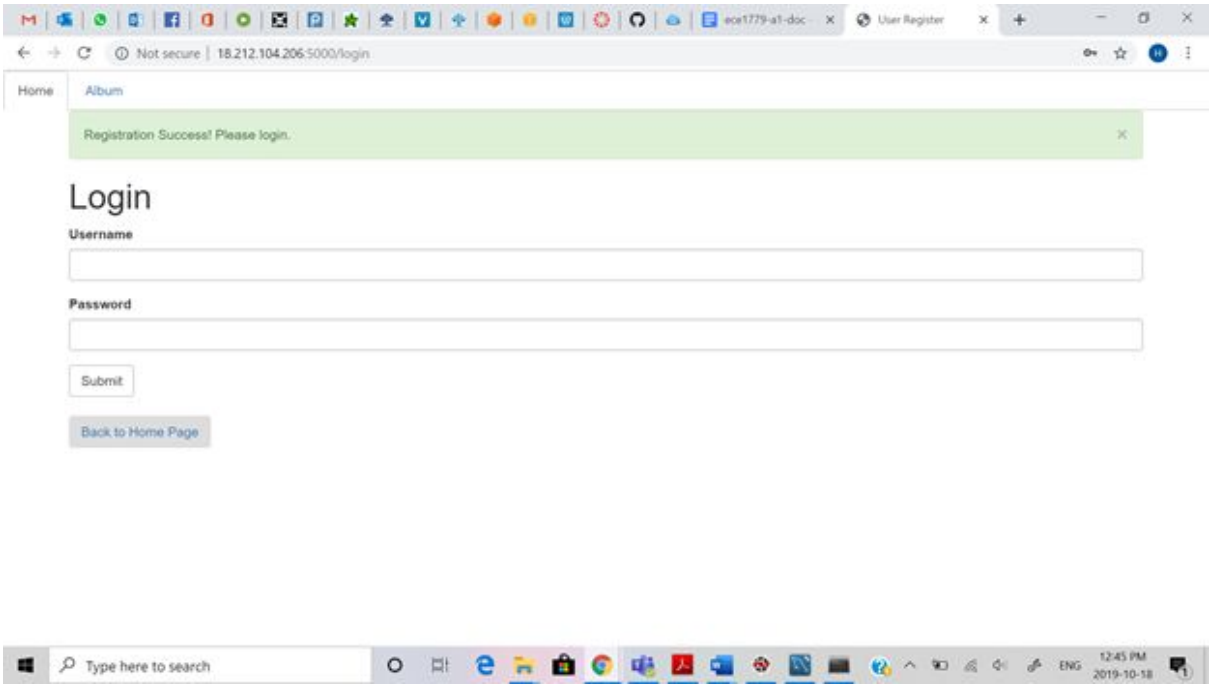
The base web page:



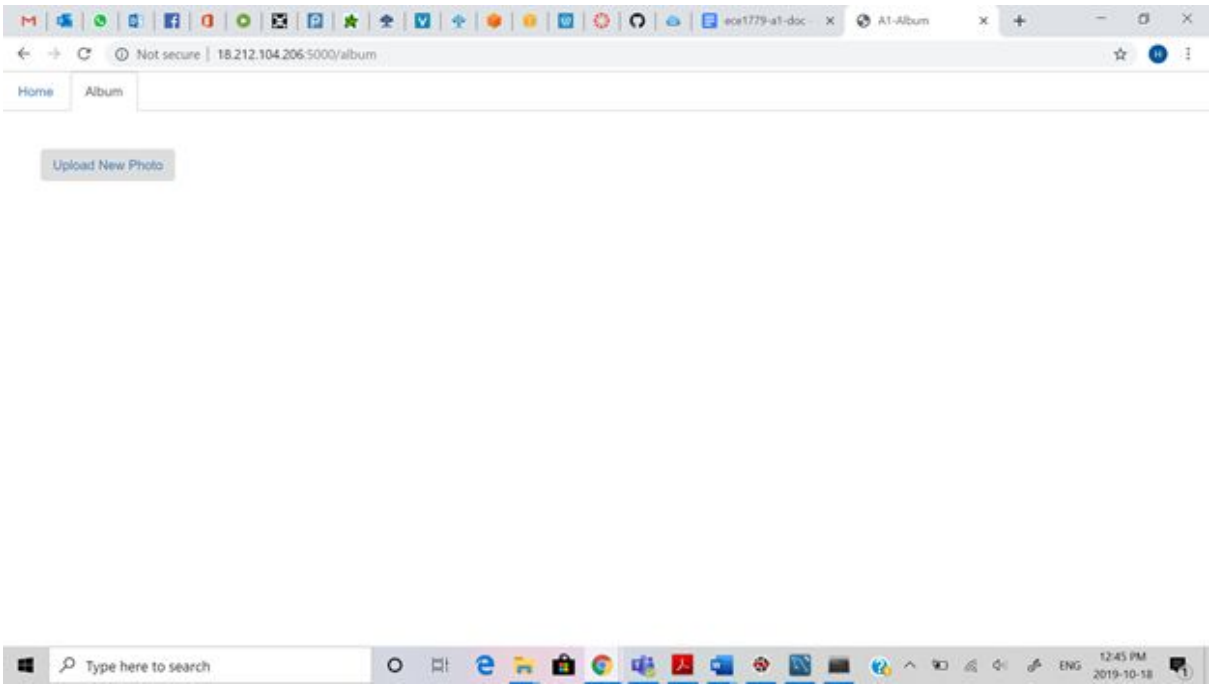
Register a new user: (Click the “register” button)



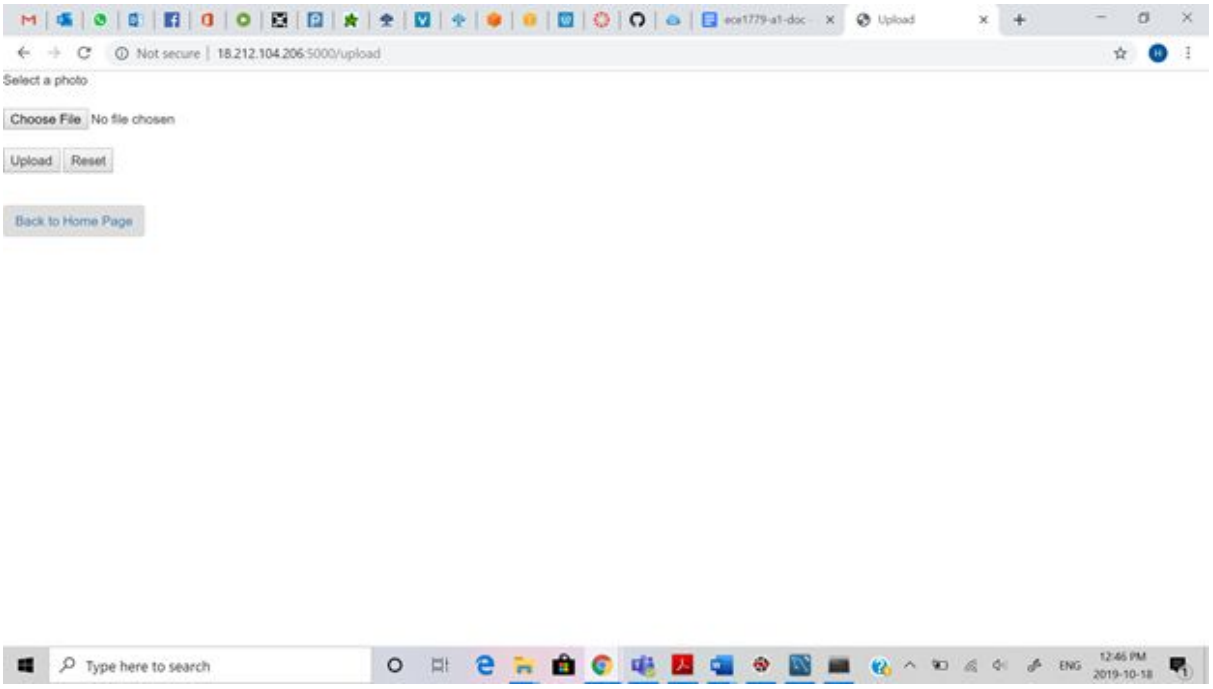
Login: (After registering or on clicking “login” button)



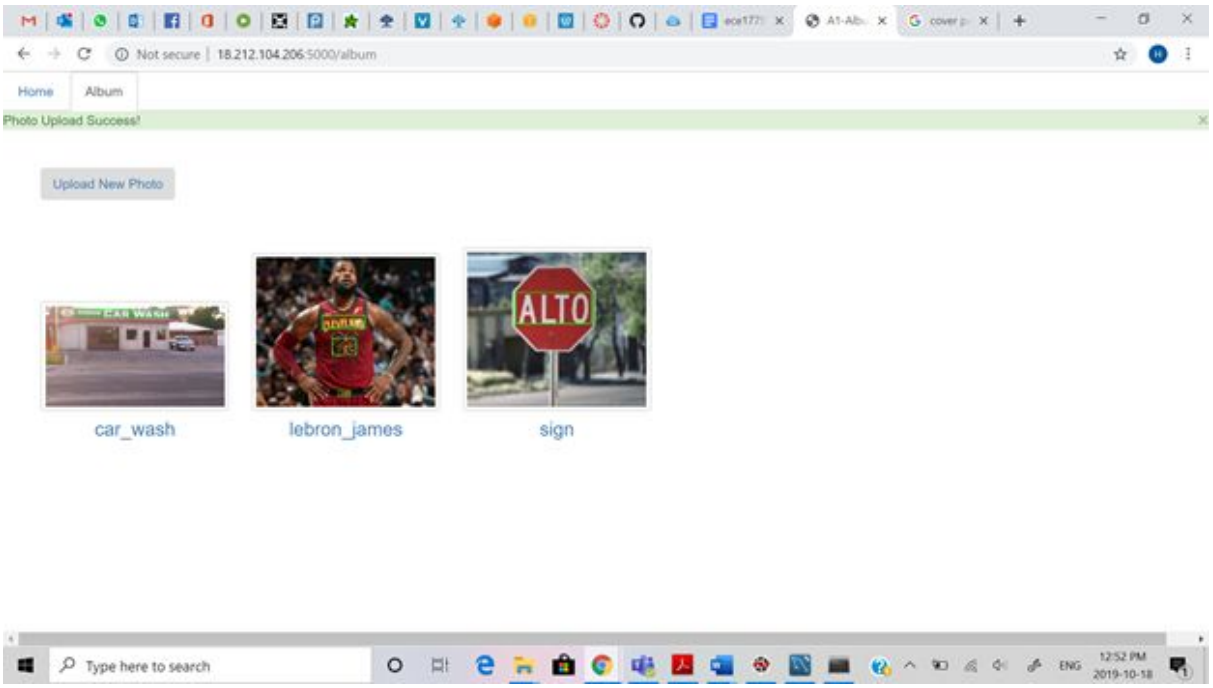
Album: (On clicking on the “album” tab)



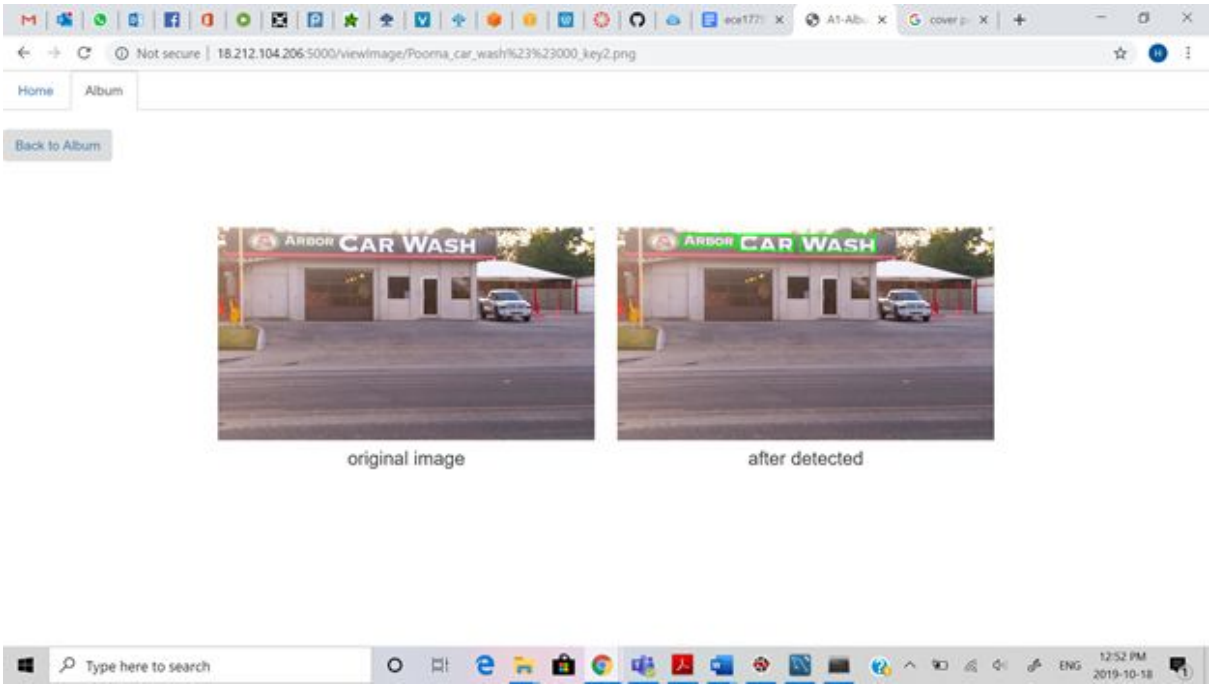
Upload new photo: (After clicking on the “Upload new Photo” button)



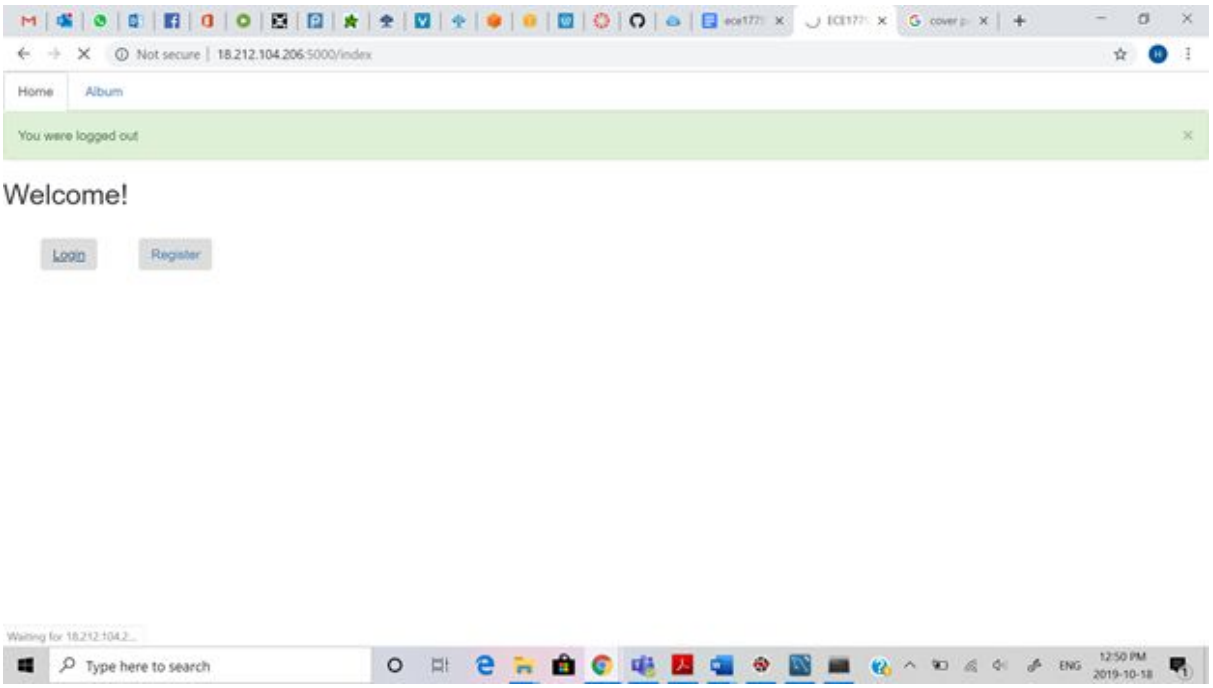
Album: (After uploading photo)



Select an image from album:



Logout:



CODE DOCUMENTATION:

The code is divided into sections in the “a1” directory as follows:

- db_table.sql
- config.py
- run.py
- frozen_east_text_detection.pb
- text_detection.py
- start.sh
- .vscodE

  --- settings.json

- app

  --- __init__.py

  --- api.py

  --- forms.py

--- login.py

--- upload.py

--- utils.py

--- viewImage.py

- static/styles

  --- ui.css

- static/uploads
- templates

  --- base.html

  --- login.html

  --- myalbum.html

  --- register.html

  --- twoImages.html

  --- upload.html

- **db_table.sql**

This code defined the tables that are to be created in the database “newdb”.

There are two tables:

1) users - This contains the information about users, including userID, password and salt value.

Field	Type	Null	Key	Default	Extra
userID	varchar(20)	NO	PRI	NULL	
password	varchar(64)	NO		NULL	
salt	varchar(12)	NO		NULL	

2) photos - This contains the user information (userID) along with the information about the 3 images saved i.e., the original photo, the text detected photo and the thumbnail using 3 different additions to the photo name (username_filename_key0, username_filename_key1 and username_filename_key2 respectively).

Field	Type	Null	Key	Default	Extra
userID	varchar(20)	NO		NULL	
key0	varchar(100)	NO	PRI	NULL	
key1	varchar(100)	NO	UNI	NULL	
key2	varchar(100)	NO	UNI	NULL	

- **config.py**

This code defines the secret key and the location of the images locally.

It also defines the configuration of the database setting a user, password and host for the "newdb" database used.

- **run.py**

Used to run the web application.

- **frozen_east_text_detction.pb**

Code for the deep learning based text detector that is the EAST detector.

- **text_detection.py**

This script is used to detect text in images. It uses NumPy and OpenCV.

detect_text(): This function takes input argument "image", with a default set for others and is used in the upload.py script. OpenCV is used to read image, make a copy and resize it. Then, the EAST text detector is loaded and the image is passed through its layers. NumPy functions

are used to create and identify the correct bounding boxes for the text. This function returns the text detected image.

- **settings.json**

This code is used to generate appropriate http responses in JSON format.

This code contains the path for executable python.

- **__init__.py**

This script creates the application object as an instance of flask. The module passed under the "__name__" variable is used as the starting point.

The config module is called to set the secret key and session lifetime for 24 hours. i.e, the logged in user stays logged in for 24 hours.

Then all the requisite modules are imported, and the blueprint is registered for the api with a defined prefix.

- **api.py**

register(): Function created for TAs, included in api module. Username and password are obtained. Checking if the prerequisites conditions are satisfied, the app gives appropriate response messages. The username and encrypted password are added to the database.

upload(): Function created for TAs, included in api module. Username, password and file are required as form-data. Return a json object with result and an http status code. This function is able to store image into local folder and store path into database if all input are valis.

- **forms.py**

This script defines the "register" and the "login" forms that are available to the user using FlaskForm, flask_wtf.file, wtforms and validators functions that are predefined.

registerForm(): validate input from registration form.

loginForm(): validate input from login form.

- **login.py**

This script uses the forms and utils modules. It also uses the render_template function for usage of base.html, register,html and login.html.

index(): redirect to main page. Display the page initially and if login is false.

register(): used for the "/register" uri with "get" and "post" methods and the register function get the register form created in forms.py. The user can register based on a few constraints and if successful the userID and encrypted password are added into the database and the user is redirected to the login page.

login(): used for the "/login" uri with "get" and "post" methods and the login function validates the user information on submission. Appropriate error and success messages are shown for incorrect password and successful login.

logout(): The logout function is used for "/logout" uri. It clears the session and redirects the user to the opening page.

- **upload.py**

This script uses flask, flask_uploads, boto3, pillow and tempfile functions. It uses the utils and the config modules.

go_album(): The myalbum html page is used for the "/album" uri and the defined go_album function first checks if the user is logged in. If the user is logged in the album page displays thumbnails of all the images previously uploaded by the user.

upload_form(): The upload html page is used for the "/upload" uri and is returned by the upload_form function.

upload(): The "/upload_sumbit" uri is used with a "post" method and defined by the upload function. For a logged in user, the function checks the uploaded file name, format and size, and if they follow the constraints the uploaded file is saved in the format as defined in db_table.sql. On the uploaded image text detection is performed and the thumbnail is created and saved along with the original image. The user gets a success message and is redirected to their album.

If any of the constraints for the image are not met, the user gets an error message and is redirected to the upload page.

If no file is selected, the user gets a warning message and is redirected to the upload page.

- **utils.py**

This script is a collection of utilities.

connect_to_database(): Connect to the database.

get_db(): The database attributes are obtained through the get_db function.

teardown_db(): Close the database in case of exceptions.

encryptString(): The encryptString function is used for encrypting using sha256 from the hashlib.

keynameFactory(): The keynameFactory function is used to create and store the filenames for different images uploaded with suffixes such that a user can upload images with similar file names. This is also done for the images obtained and stored as result as well.

normalName(): The normalName function is used to obtain the original filename without the suffixes that were added in the keynameFactory function.

- **viewImage.py**

This script is used to return the twoImages html template. It uses the utils and the config modules.

viewImage(): used for defining the page for "/viewImage/<key>" uri with the viewImage function. On selecting a specific thumbnail, the original image and the text detected image are retrieved from the local storage.

- **ui.css**

This script is used to define the various styles of display that are used in the html template files.

TEMPLATES:

- **base.html**

This html script defines the base page for the user.

The user can navigate between “home” and “album” pages. A part of the webpage is left to display the alert messages that can be closed with a button. The page has a greeting for the user if logged in, else it displays buttons for “Login” and “Register”.

- **login.html**

This script defines the login page.

The user can navigate between “home” and “album” pages. A part of the page is left to display alert messages that can be closed with a button. The form fields are displayed using “post” method from the login form, with fields username, password and submit. A “Back to Home Page” button redirects the user to the base html page.

- **myalbum.html**

This html script defines the album page for the user.

The user can navigate between “home” and “album” pages. A part of the webpage is left to display the alert messages that can be closed with a button. The album page has button for uploading new photos. It displays thumbnails of all the uploaded images that on click, lead to the url for viewImage.

- **register.html**

This script defines the register page.

The user can navigate between “home” and “album” pages. A part of the page is left to display alert messages that can be closed with a button. The form fields are displayed using “post” method from the register form, with fields username, password, confirm, submit and reset. A “back to Home Page” button redirects the user to the base html page.

- **twoImages.html**

This html script defines the images page for the user.

The user can navigate between “home” and “album” pages. The page has a Back to Album button that redirects to album page. It displays the original image along with the text detected image from the local storage.

- **upload.html**

This html script defines the upload page for the user.

The navigation can be done tab using through “home” and “album”. A part of the page is left to display alert messages that can be closed with a button. The uploaded image gives size error based on constraints, else using the “post” method for /upload_submit, the file is uploaded, or the page is reset. The page has a Back to Home Page button that redirects the user to base page.

CODE:

https://github.com/judyliou/ECE1779-CC/tree/master/a1