**ECE1779: WEB DEVELOPMENT**

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
* VNC server
* OpenCV
* Python OpenCV binding (cv2)
* gunicorn\*\*
* Flask

WORKING:

--- steps to log in to the aws account, start the instance and run the code (how to use the app)---

GENERAL ARCHITECTURE:

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

* db\_table.sql
* config.py
* run.py
* .vscode

--- settings.json

* app

--- \_\_init\_\_.py

--- api.py

--- forms.py

--- login.py

--- upload.py

--- utils.py

--- viewImage.py

----- text\_detection.py to be added -----

--- static/styles

--- ui.css

--- 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 2 tables:

1. users - This contains the information about a user i.e., their userID and password.
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).

Both tables use InnoDB as engine, charset as utf8mb4 (4 byte, UTF-8 Unicode encoding) and collate as utf8mb4\_0900\_ai\_ci (accent and case insensitive collation algorithm).

The database schema can be represented as:

--- add image of database schema ---

* 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” databased used.

* run.py

Used to run the web application.

* 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

The script uses the modules “upload” and “utils” and the path of the local storage.

The way the different modules interact is defined here.

Using the “post” method, for the “/register” uri, the username and password are obtained. Checking if the prerequisites conditions are satisfied, the app gives appropriate response messages. The username and the encrypted password are added to the database.

Using the “post” method, for the “/upload” uri, the user is allowed to login if correct password is provided for the given userID.

Then, the user can upload an image that, if in correct format and matches the sizing specifications, is performed text detection and resized to obtain the thumbnail. These 3 are saved according to the format specified in db\_table.sql and the thumbnail is displayed with a success response.

* 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.

The 2 forms are defined as a class with the requisite fields defined with certain conditions.

The submit fields for register, reset and submit are defined.

* 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.

The base html page is used for the “/index” uri and the index function is used to display the page initially and if login is false.

The register html page is 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.

The login html page is 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.

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.

The allowed file types are defined.

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 the thumbnails of all the images previously uploaded by the user. On selecting a specific thumbnail, the original image and the text detected image are retrieved from the local storage.

The upload html page is used for the “/upload” uri and is returned by the upload\_form function.

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 used to connect the mysql connector. Boto3, client errors, logging and hashlib packages are imported. The config and app modules are also imported.

The mysql.connector package is used to connect to the database through the connect\_to\_database function.

The database attributes are obtained through the get\_db function.

The teardown\_db function is defined to close the database in case of exceptions.

A url is created for the local storage in the create\_presigned\_url function, returning nothing in case of error.

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

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.

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.

It is used for defining the page for “/viewImage/<key>” uri with the viewImage function.

The function is used to obtain the 2 images, original and text detected, from the storage using the generate\_presigned\_url function using the “get\_object” client method and defined parameters.

* ui.css

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

* base.html

This html script defines the base page for the user. It has url for static and uses the ui.css file.

The navigation can be done tab using through “home” and “album”.

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. It has url for static and uses the ui.css file.

The navigation can be done tab using through “home” and “album”.

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 the thumbnail 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. It has url for static and uses the ui.css file.

The navigation can be done tab using through “home” and “album”.

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. It has url for static and uses the ui.css file.

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.

ISSUE TRACKER

CITATION

CONTACT INFORMATION