**Documentation**

ECE1779 – Introduction to Cloud Computing – Assignment 1

**Project Summary**

The website allows users to register an account with a username and password, and then login with the same credentials. Once logged in, users can upload photos on which a text detection software is automatically applied, and the modified image is hidden from the user. Thumbnails of every photo uploaded by the users are presented upon logging in. Clicking on a thumbnail enlarges the image and presents the modified image with rectangles around the text in the image along with the original image. The basic application flow can be described with the following diagram:

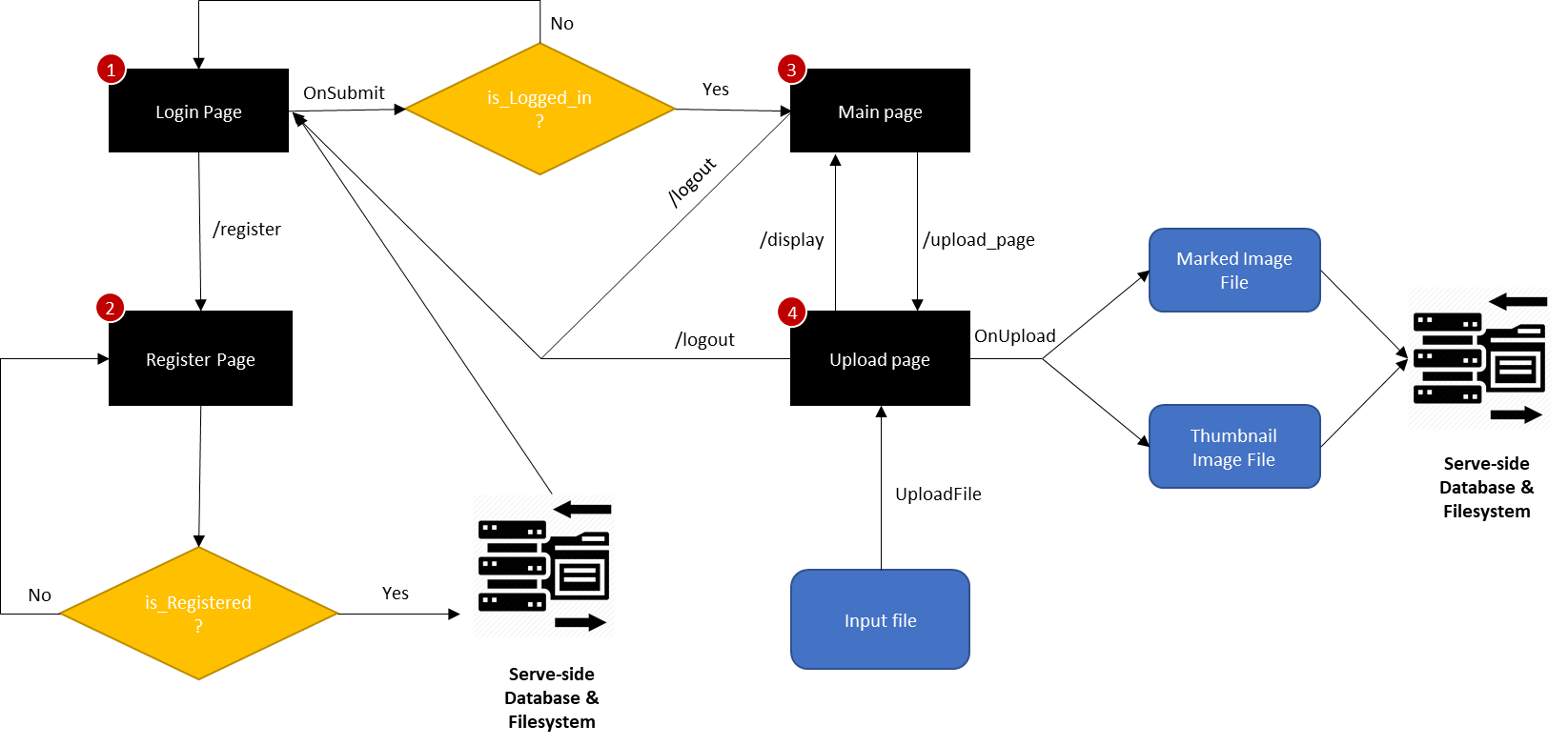


Figure 1: Application Flow diagram of our website (see Appendix for screenshots of each page)

**Features**

1. User registration
2. User login/logout
3. Image uploads
4. Text detection in backend
5. Displaying original image and modified image side-by-side

**Packages Required**

1. Python 3
2. Flask
3. Flask – bcrypt, login, mysqldb, sqlalchemy, wtforms
4. Werkzeug
5. MySQL connector
6. A virtual environment with these packages (not necessary but highly recommended)

All the packages can be installed using pip.

**Running the web application**

1. Assuming the MySQL server is connected, from the desktop, run *bash start.sh* to deploy the web application. If there is no MySQL server connection, connect to the server first before continuing. Alternative steps to manually run the applications are provided in steps 2-3.
2. The main directory is present in *~/ece1779/ece1779assignment1/Assignment1/*. Change to ~/ece1779/ and activate the virtual env present by running ***. venv/bin/activate****.* This environment contains all the packages that are required. Alternatively, created your own virtual environment with the required packages installed.
3. Change to *~/ece1779/ece1779assignment1/Assignment1* and then run ***python3 run.py*** to deploy the web application.

**New Users**

1. Go to <IP:5000/> which opens the login page. Log in using your credentials. If you do not have an account yet, create one by clicking **Register** at the top right.
2. Once logged in, you can see thumbnails of all the photos you have uploaded so far. You can upload new images by click on the **Upload** button.
3. To see the image and the text detection in play, click on any of the thumbnails. You will be presented with the image you uploaded, and the modified image with text in the image surrounded by rectangles.
4. You can logout at any time by clicking the **Logout** button.

**Developers**

For developers, there are 2 main parts that are documented here: the frontend, and the backend.

*Frontend*

The functions of each directory and file that are used in the frontend are listed here:

1. **Static –** In this directory, the *css* and *img* directories are used to hold the main css styling file, and the background image of the website.
2. **Templates –** These are HTML templates that are used when each route is called. There is a template for logging in, registering, uploading an image and displaying thumbnails/images.

*Backend*

The backend uses Flask, a microframework. Each backend file is described here:

1. **\_\_init\_\_.py –** A ‘constructor’ module of sorts that runs every time the application is run. It instantiates a Flask instance, defines a secret key and creates a Bcrypt instance used for hashing the passwords.
2. **routes.py –** This handles all the route requests from the website. This is also where the secret key for the website is stored. A brief description of each route is given below:
   1. **login –** This is the primary homepage and also where the user is taken to when the ‘Login’ button is clicked. If the user’s credentials are correct (i.e., they are already present in the database) the user is logged in and sent to the **display** page. If the credentials are wrong, an error message is displayed, and the user is brought back to the login page.
   2. **register\_page –** If the user passes valid credentials (i.e., the username doesn’t already exist in the database), then a registration successful message is displayed and the user is taken to the **register** route. If any of the validators for the fields fail (explained in **forms.py**), then an error message is displayed, and the user is taken back to the **registration\_page**.
   3. **register –** If the registration is successful, the username and the *hashed* password are stored in the database.
   4. **upload\_page –** If the user is logged in, and clicks on Upload, they are redirected to the **upload\_page**. Here, the image uploaded is stored in */app/static/Input/*with a unique name assigned to it. The unique name is given by the format:

**username + current\_time\_stamp + file\_name**

This ensures that different files with the same name are stored properly, and images with same filenames of different users are also stored properly. This *name* is then inserted into the database along with the username.

Then a thumbnail is created out of the image with the same name and stored in *app/static/thumbnail*, and the text detection software (explained further in **textdetection.py**) is applied to identify text in the images. The modified image is stored under *app/static/Output*.

If the user is not logged in, they are redirected to the login page.

* 1. **display –** If a user is logged in, the thumbnails of all the images the user has uploaded is displayed. If any thumbnail is clicked, an enlarged version of the image, along with the modified image is displayed.

1. **forms.py –** This module defines the format and constraints of the Login, Register, and Upload forms. The constraints used are:
   1. **LoginForm:** Username and password must be present to login.
   2. **RegisterForm:** Username must be between 4 and 20 characters, the password must be between 4 and 20 characters, and the confirm\_password should be equal to the password field. Data is required in all the 3 fields.
   3. **UploadForm:** Files are accepted in **jpg, jpeg and png** formats. If the file size is greater than 5MB, an error is displayed. A file is required to enable the Upload option.
2. **text\_detection.py –** This module performs the text detection on the uploaded image. Using openCV2, rectangles are drawn around the text present in the image, and the modified image is stored in */app/static/Output* with the same filename.
3. **resources.py –** This is a helper module used to create thumbnails and retrieve lists of thumbnails or images required for displaying.
4. **databaseModule.py –** This is the database module used to create a database, the tables necessary, and insert/retrieve information such as the username, hashed password, and the image list. The database schema is presented in Fig. 2. The functions defined in this module are:
   1. **create\_database –** This automatically creates a database if one doesn’t exist with the given name.
   2. **create\_table –** This creates two tables, defined by the schema in Fig. 2.
   3. **insert\_user\_database –** This inserts a username and the hashed password into the *user\_list* table (primary key=username).
   4. **insert\_image\_database –** This inserts the image along with the username into the *image\_list* table.
   5. **verify\_username\_password** – Using Bcrypt module, the entered password for a given username is hashed, and then checked against the password present in the database. If there is a match, it returns True, else a False.
   6. A screenshot of a cell phone

      Description automatically generated**get\_image\_list** – For a given username, all the image names corresponding to that username are extracted and returned as a list.

Figure 2: Database Schema before and after Normalisation

**Appendix**

Login Page:

A picture containing outdoor, sky

Description automatically generated

A picture containing sky, outdoor

Description automatically generatedRegister Page:

A black sign with white text

Description automatically generatedDisplay Page:

A view of a city

Description automatically generatedUpload Page:

Enlarging thumbnail with modified image:

A screen shot of a person

Description automatically generated