ECE 1779

Introduction to Cloud Computing

Assignment 1: Web Development

By

Sai Anirudh Basamsetty [1006042747]
Zobair Ahmed Khan [1005611875]
Shreya Kishore Madathingal [1005620151]

Aim:

To develop a web application using Python and Flask which has the functionality of detecting objects in images.

How to use our app:

1. The web application is being launched on the EC2 instance with instance id “i 0997916775e0b7d71”.

2. The object detection app can be started by using the shell script ‘start.sh’ which is in our EC2 my_app folder on Desktop. To run this script, change directory to ./Desktop/my_app where the file is located and type the given command in the terminal: source start.sh

3. Once the web app starts, you can see the main page which shows a description about the web app and allows you to sign up or sign in.

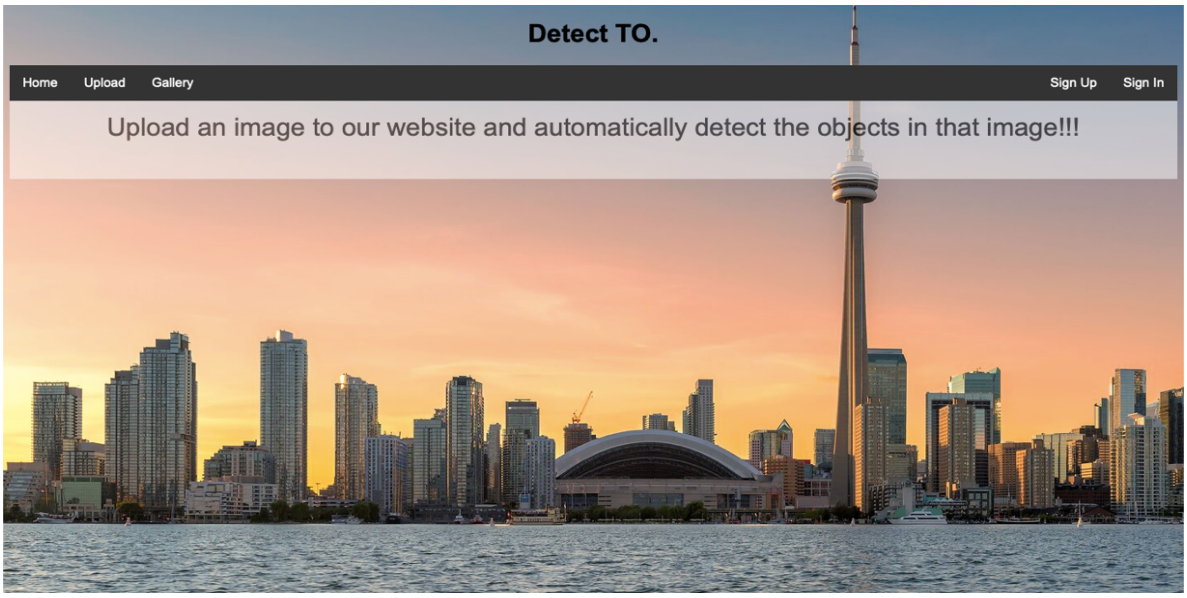

Figure 1: Main page

4. If the user chooses to sign up, they are taken to the registration page. Here, they can fill up the form to sign up new users. The length of username should be less than 30 characters and the password should be at least 8 characters with a mixture of uppercase, lowercase and numbers in order to be accepted. If these rules are not followed, the error message shows up and the user has to fill up the form again.

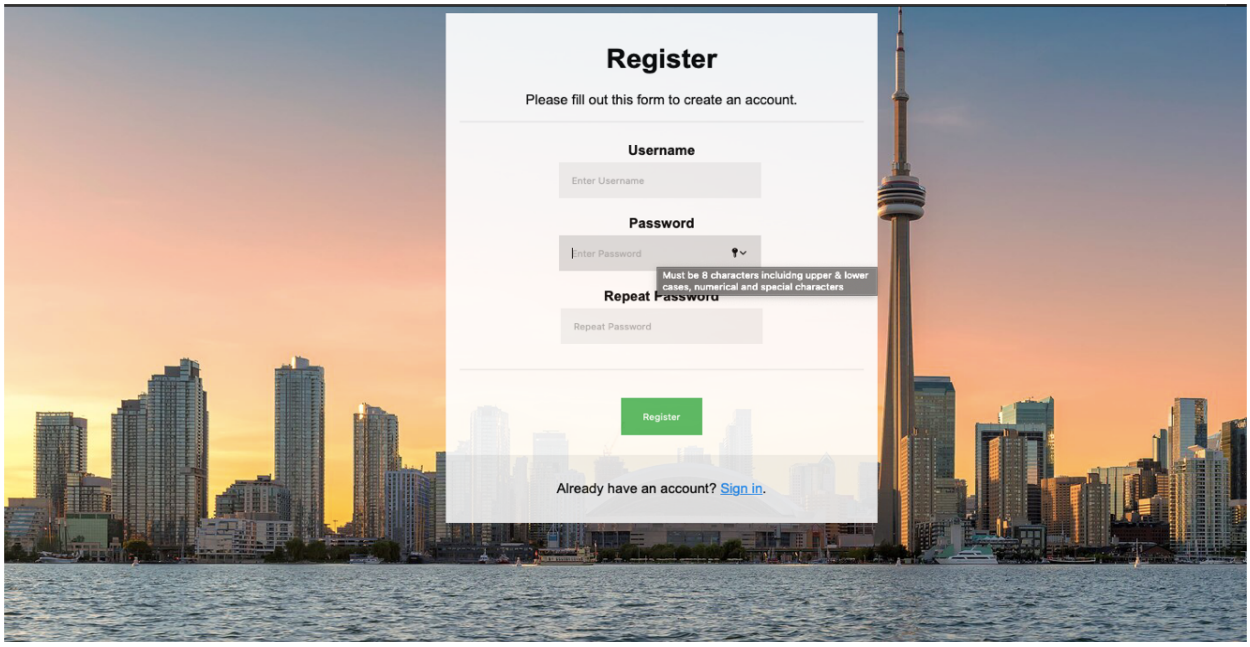

Figure 2: Register page

5. Once the registration is successful, the user is redirected to the login page. Now the user can sign in using the correct credentials which is validated by the back-end.

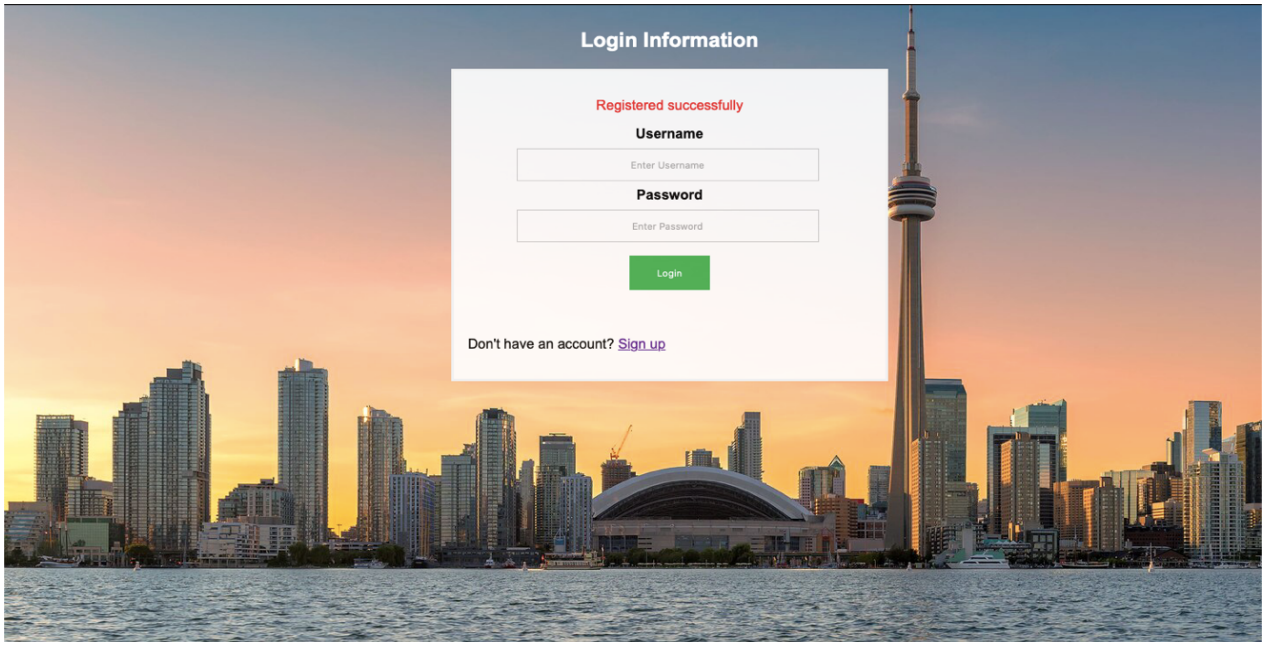

Figure 3: Login page

6. After logging in, the user can see the home page. Here, the navigation bar is available for the user to upload images, see gallery and also logout.

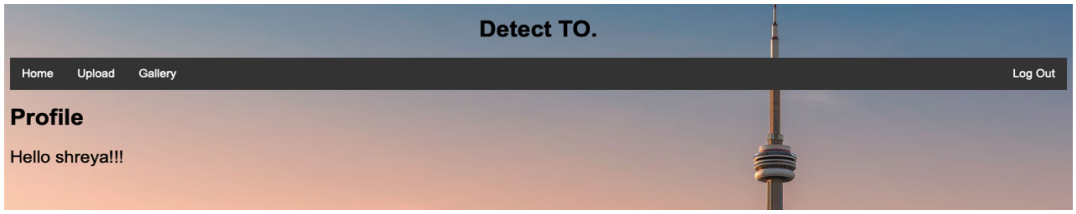

Figure 4: Home page

7. The user can now upload images by clicking on the 'Upload' tab in the navigation bar. This allows the user to upload multiple image files. The size and image formats are restricted by the back-end function. Any exceptions will lead to an error case. If the image is successfully uploaded, the user is directed to the 'Gallery' page.

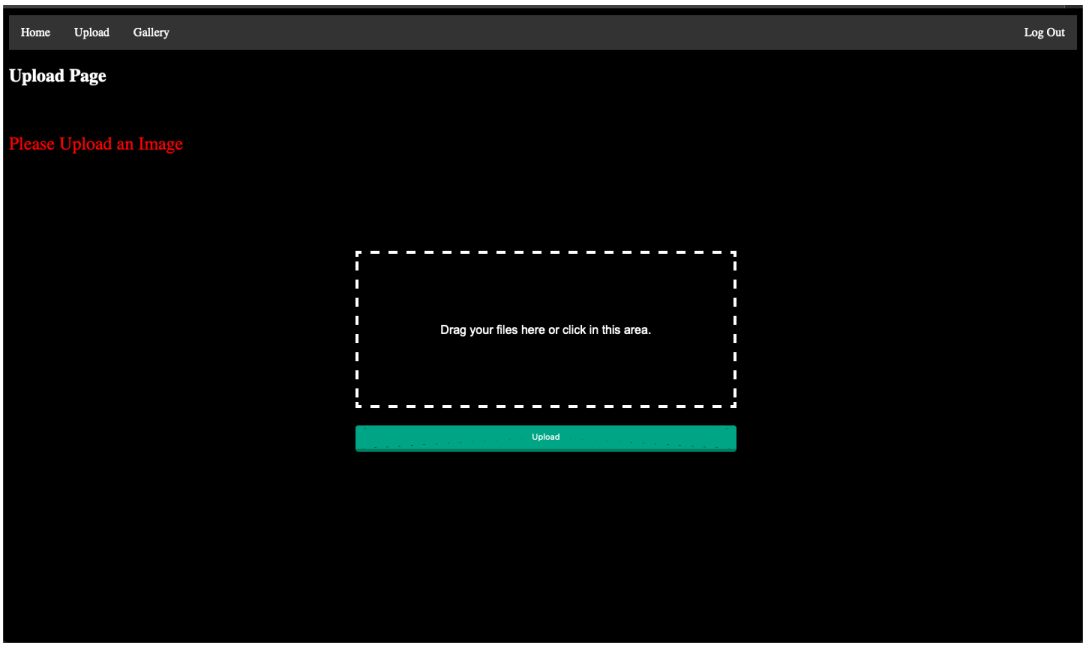

Figure 5: Upload page

8. The gallery page has all the thumbnails of the images uploaded by the user. The user can click on any of the images, which opens a new tab. This new tab is the 'details' page.

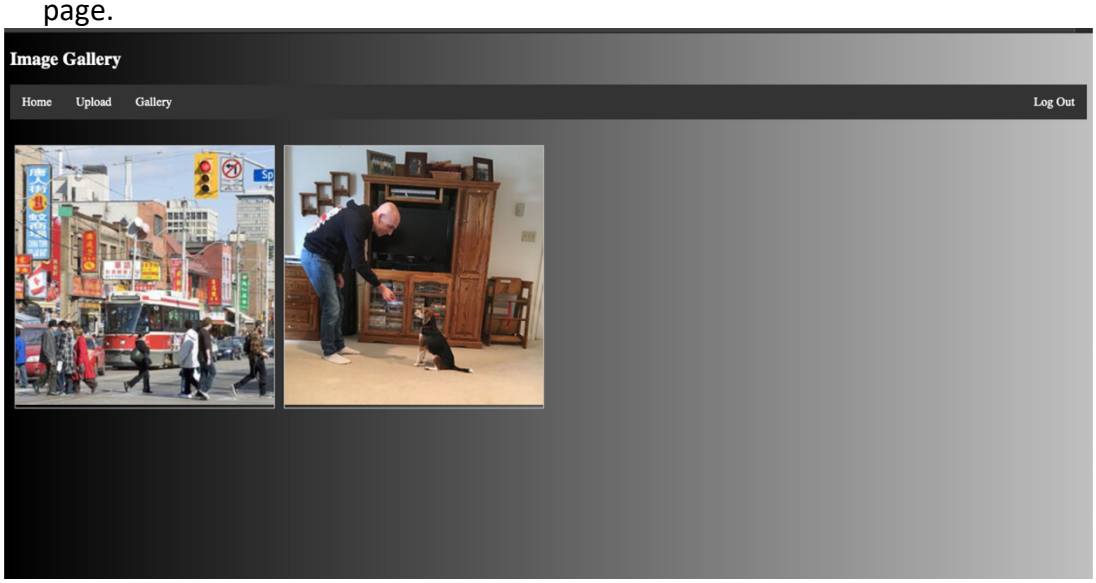

Figure 6: Gallery page

9. The details page for the image automatically detects objects in the image. So, this page displays the original image and the image with all the objects highlighted using rectangular boxes.

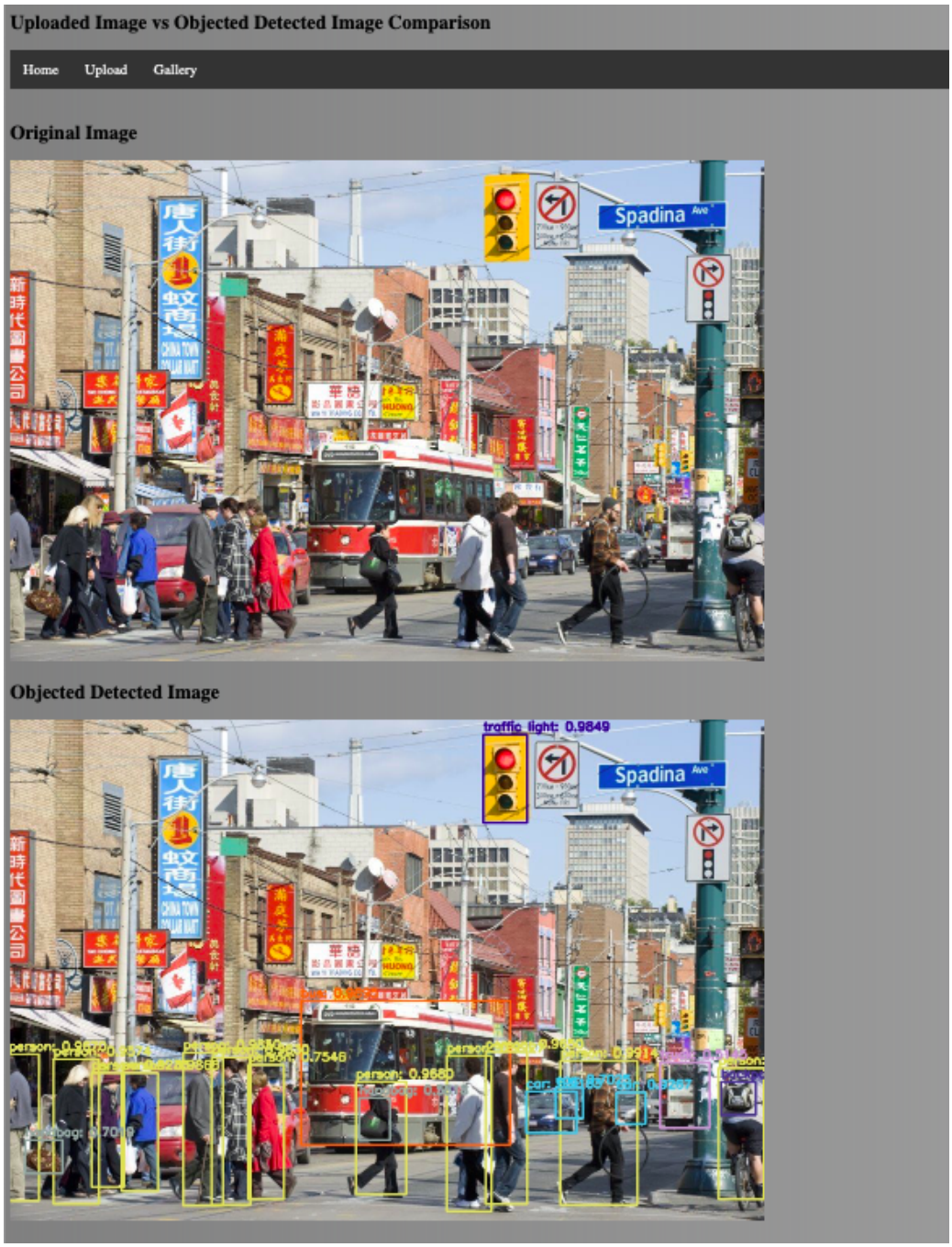

Figure 7: Details page

10. The user can log out using the button in the navigation tab. This clears all the session data for the user and redirects them to the 'Main' page of the web app.

APIs for TAs:

The TAs can access the api for registering and uploading just by using '/api/register' and '/api/upload' which are the two URL end-points.

AWS Credentials:

The AWS credentials used for creating the EC2 instance are:

Username: anirudh.basamsetty@mail.utoronto.ca
Password: Ece1779_pass

General Architecture of application

1. Project schema of web app:

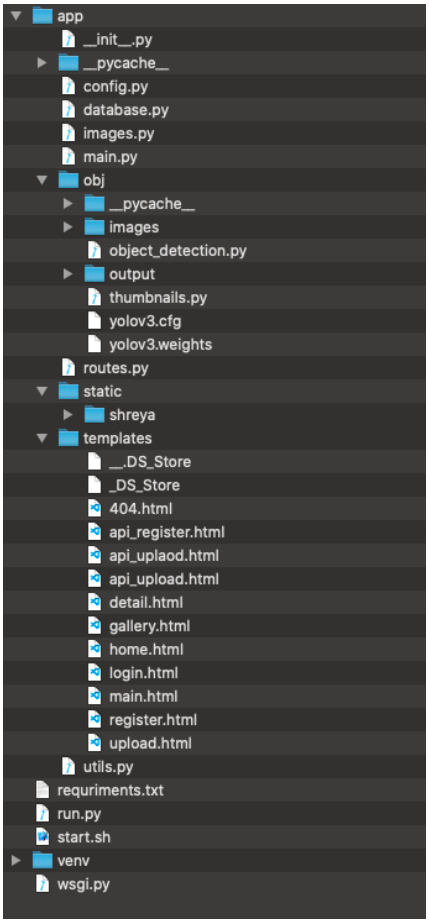

Figure 8: Project Schema

The above figure has the complete structure for the web application. All the templates for html front end are stored in the folder 'templates', the folder 'obj' has all the files related to the object detection functionality of the application. It has the python file for object detection and configuration files of the trained yolo3 model which is used for this purpose. The main python files for the back end of the web application are stored in the folder 'app'. Other files required for setting up and running the application are stored in the my_app directory (main directory of our project).

2. Front end (Presentation layer)

We use html to design a user-friendly front end for the app. The styling is done using CSS in the same html code. The following templates are created for this app:

1. *Main*: The first page for our web application which describes it.

2. *Register*: For registering new users.
3. *Login*: Sign in for returning users.
4. *Home*: The welcome page for users after they log in.
5. *Upload*: The user can upload images on this page.
6. *Gallery*: All user images' thumbnails are displayed here.
7. *Detail*: This page shows original image and image with objects detected.
8. *Error*: This page is used for dealing with errors.

3. Back end (Logic layer)

Python and Flask are used to build the back end. The files that handle the major functionalities of our application are:

1. __init__.py : This file is used for initialization of the application using Flask.

2. main.py: The main page for the web application is built here.

3. Utils.py: This file has all important functions for the app like handling sessions, connecting to database, checking username and password, validating credentials, etc. Some important functions in this file are given below:

Table 1: utils.py functions

Sr. no.	Function name	Function description
1.	Make_session_permanent()	User is logged in for 48 hours.
2	Hashed_password(word, salt)	Password is hashed and then a randomly generated salt is added to it.
3.	save_reg(username, password)	The hashed password is saved in the database along the username. This function also creates a user directory in the static folder, where images are stored.
4.	save_file(username, file, filename)	Save uploaded images in correct user directory. It saves original file, object-detected image file and the thumbnails in their respective folders.

4. Routes.py: The routing functions for all the html pages are written in this file. It includes functionalities like registering new users, logging in, logging out, error handling as well as the api file for registration. The functions in this file are:

Table 2: routes.py functions

Sr. no.	Function name	Function description
1	login ()	Checks if user already signed in otherwise shows login page. Username and password are verified.

2	logout()	Clears session data and allows user to logout.
3	home()	First-page for logged in users. Shows navigation tab.
4	register()	The user can register by filling the form. There are rules enforced on the length of username and password. The password and repassword should both adhere to the given rules.
5	api_register()	Url-endpoint for registration, follows same rules as the register function.

5. images.py: This file contains all the functions related to images in the web application like uploading, object-detection, displaying image thumbnails on gallery, etc.

Table 3: images.py functions

Sr. no.	Function name	Function description
1.	allowed_file(filename)	To restrict accepted file formats to png, jpg and jpeg
2.	upload()	Checks if user is logged in, then allows user to upload image files. User can upload multiple files of allowed format. It also checks the file size to prevent files which are too big.
3.	gallery()	Display thumbnails of uploaded images from user directory.
4.	api_upload()	URL end-point to upload image, works in similar manner to upload function.

6. Obj : This folder has the files used for object-detection in our web app. A trained yolo3 model is used here. The configurations of this model and its weights are saved as *yolo2.cfg* and *yolo3.weights* files respectively. It also has some sample images used for testing stored in 'images' folder while the 'output' folder has the respective outcomes. The object_detection.py file has the code where image is taken as input and the trained model is used to identify the objects. It also draws rectangles around the identified objects.
7. Wsgi.py: This file is used by Gunicorn for binding the app.
8. Requirements.txt : This file has all dependencies required for the web app to run. All of them can be installed using the command 'pip install -r requirements.txt' in the terminal.
9. Start.sh: The shell script for manually launching the app.

10. Run.py: This python file invokes flask to host the webapp on the server 0.0.0.0:5000

4. Database

Our database consists of two tables 'users' with information about the signed-up users and 'images' which contains information about all the user images and their paths.

i. Users table:

1. ID: The unique auto-generated key which is used as the primary key. It's set to auto-increment.
2. Username: The user can choose their username and the only restriction set to maximum length of 30 characters.
3. Password: The hashed password is stored in this field.
4. Salt: Stores the randomly generated salt word and has a length of 16 characters.

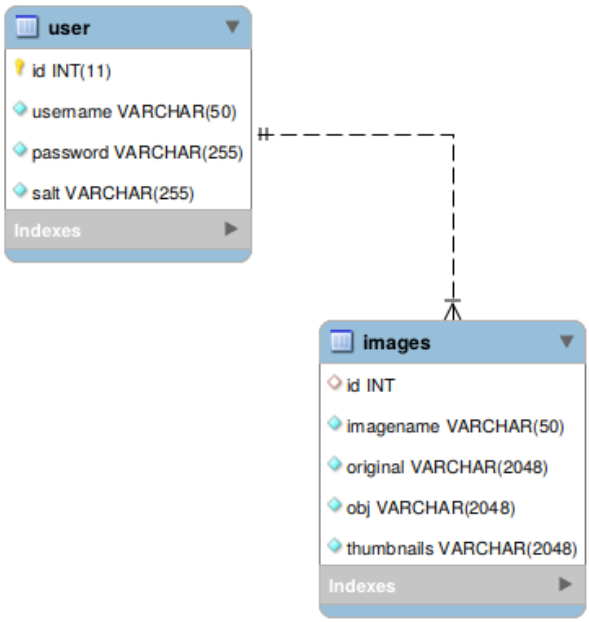

Figure 9: Database Schema

ii. Images table

1. ID: This is the reference key which connects the user column to images.
2. Imagename: It stores the names of the images, uploaded by the user.
3. Original: This contains the path to the original images stored in user directory.
4. Obj: This stores the path of the object-detected image.
5. Thumbnails: It has the information about the path to the image thumbnails.

Load generator test:

The gen.py file was run to test this web application. The results found can be seen in the following figure. This demonstrates the ability of the web application to respond to traffic and ping back.

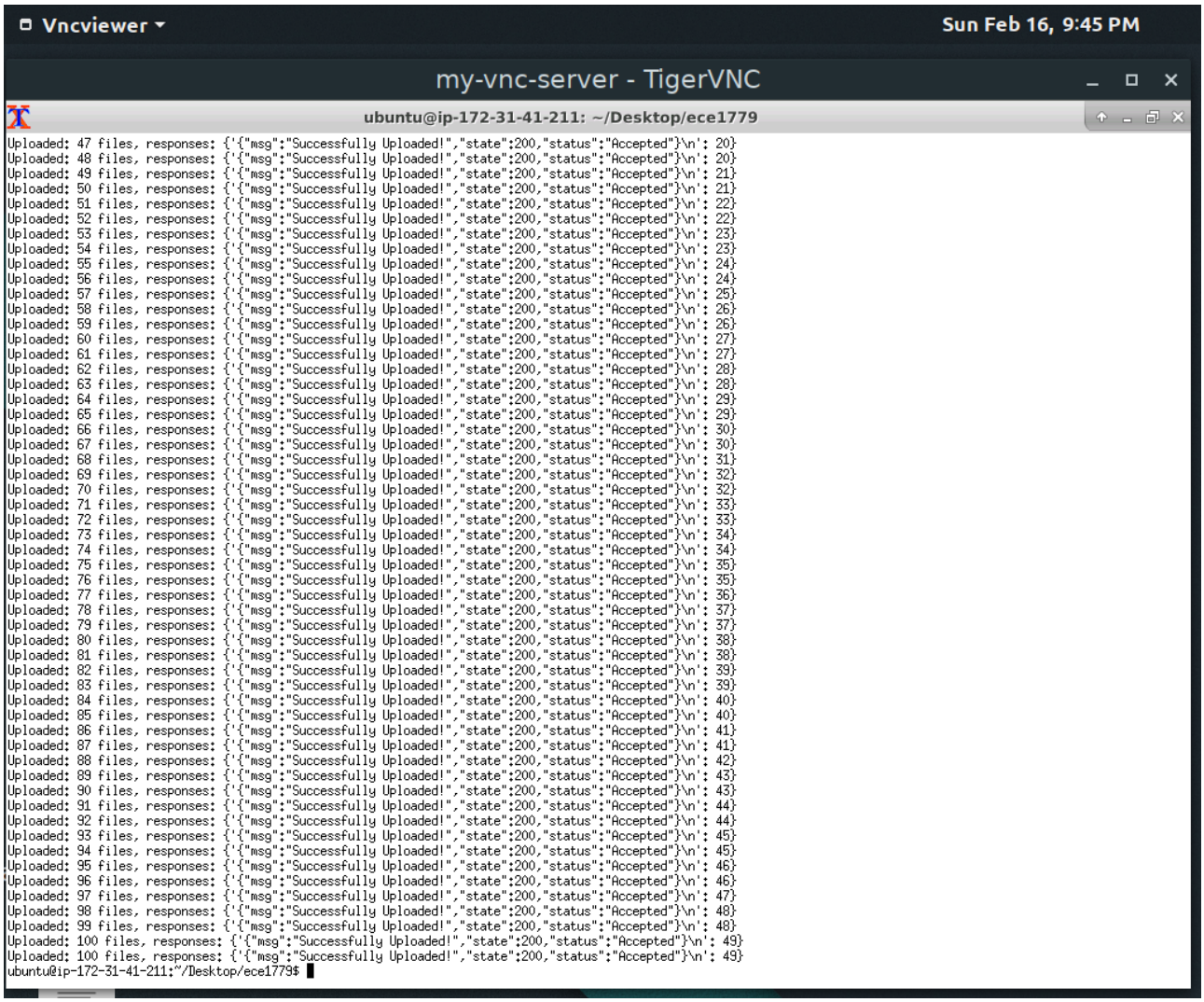

Figure 10: Result of gen.py