Documentation

Web Application Introduction

Welcome to the documentation of our object detection application. This web application enables users to:

- **Registration and login:** Create a personal account and login. Log out if they want.
- **Upload and detection:** A logged-in user can upload a new photo with objects and automatically get a detected photo with rectangles around objects. Then we will also save a thumbnail for the corresponding uploaded photo.
- **Browsing uploaded photos:** Each user will have a personal page allowing them to browse the thumbnails of uploaded photos and he or she can view the original version of the photo as well the detected photos when clicking on the thumbnail.

Agenda of the documentation

The structure of the documentation is basically in two parts: users guide and developers reference

- **Users guide：** instruction for users to use the web application with its functionality of object detection.
- **Developers reference:** instruction for developers to understand the architecture of the application.

Users guide

This guide will give the user a tutorial on how to use this application.

Login page

Open the browser and enter the dynamic URL Server's IPv4 address + :5000 to access the login page which also is the homepage of the website. The login page contains a login block which consists of two inputs: Username and Password. Username and password are required to log in.

Login process：

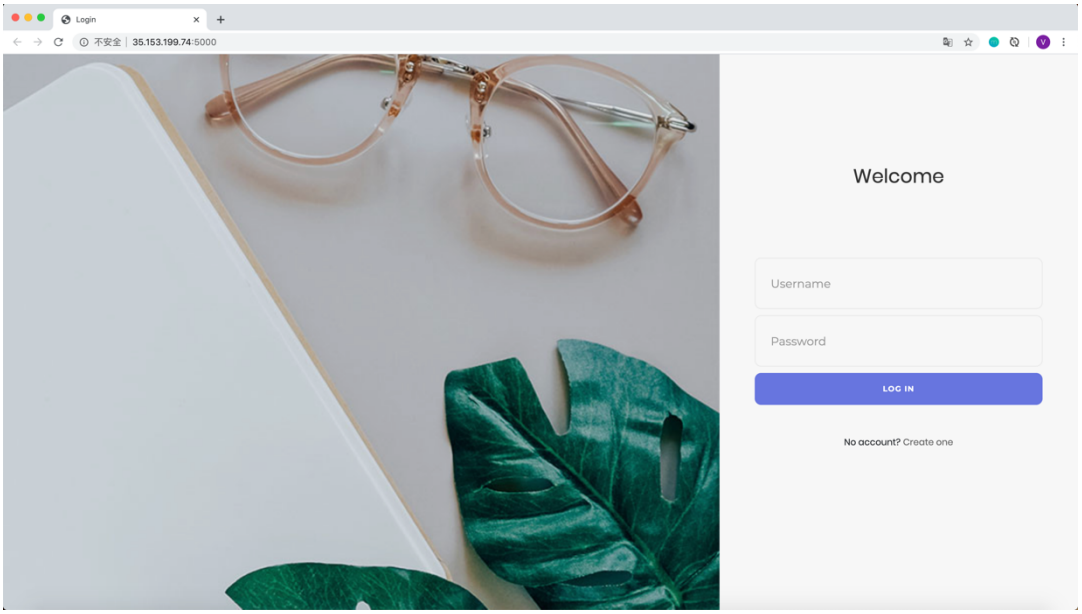
You should enter your registered username and correct password in the inputs of Username and Password and click on the **LOG IN** button to finish the login process and then you will jump to your personal page. If you do not have an account, please click on **Create one** to jump to the register page.

Error messages:

- You will get the reminder of “Username is required” or “Password is required” if one of them is empty.
- You will get the reminder of “The username does not exist. Please try again” if the username you entered does not exist.
- You will get the reminder of “Password error” if you enter an incorrect password.

Success messages:

- You will see the message of “Welcome back + your username” on the top of your personal page to show that you have successfully logged in.



Login Page

Register page

You can access to register page when you click on the **Create one** or directly visit it by entering the URL of the Server's IPv4 address + :5000 + '/register'. Register page contains a register block that consists of four inputs: Username, Email, Password and Confirm password. All the inputs are required.

Register process：

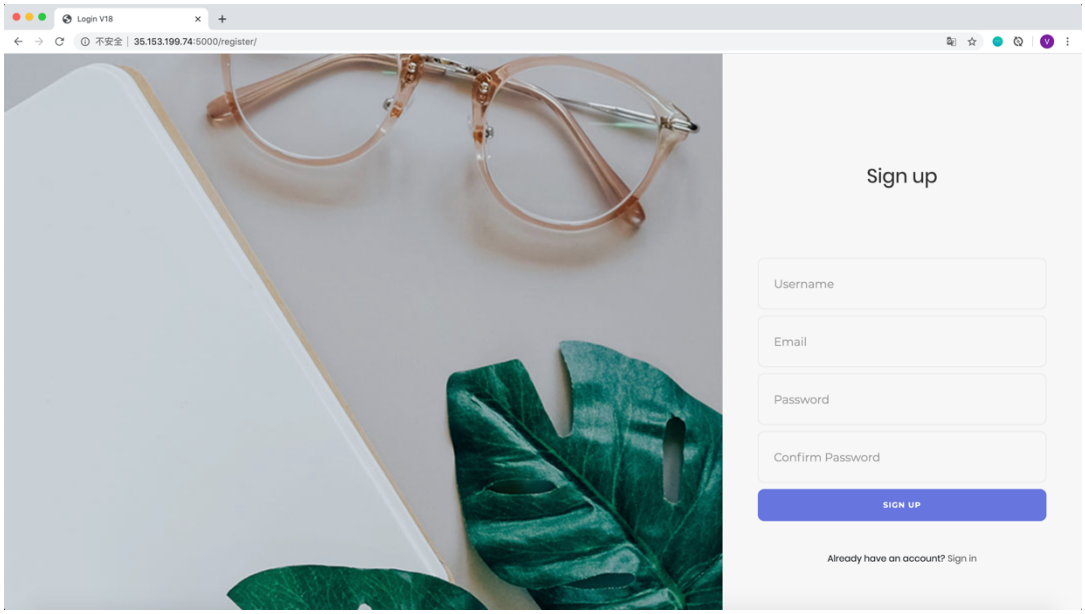
You should enter a legal username, email address, password and confirmation password in four inputs and click on the **SIGN UP** button to finish the registration process and then you will jump to the Login page to login. If you already have an account, please click on **Sign in** to jump to the Login page.

Error messages:

- You will get the reminder of “Username is required”, “Email is required”, “Password is required” and “Please enter your password” if one of the inputs is empty.
- You will get a reminder of “The length of the username is illegal!” if your username is more than 100 characters.
- You will get a reminder of “The name already exists!” if your username is the same with someone else.
- You will get a reminder of “Two passwords are inconsistent!” if your original password and confirmation password is inconsistent.

Success messages:

- You will get a reminder of “You have successfully registered!” if you have done the registration process correctly.



Register Page

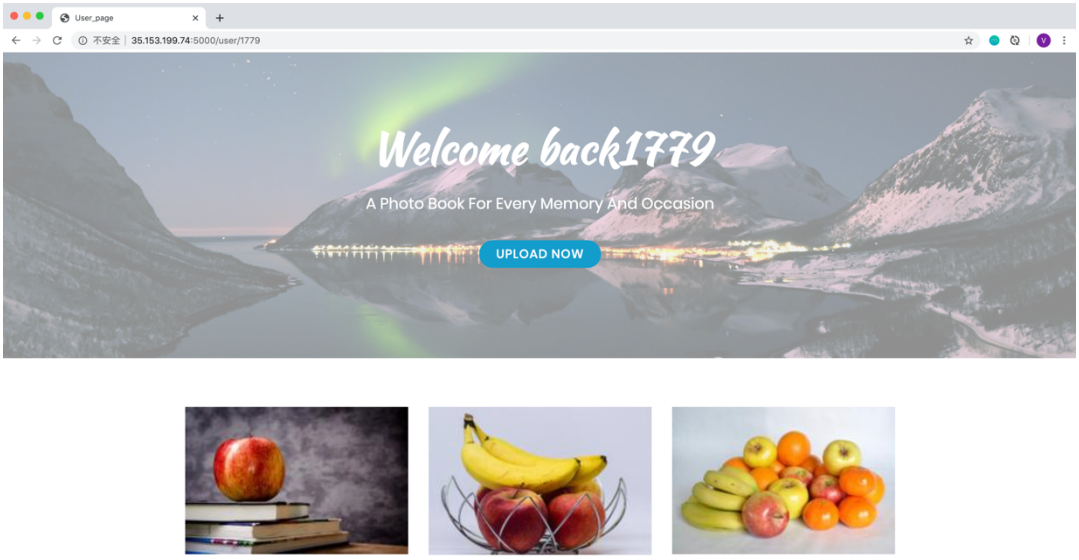
Personal page

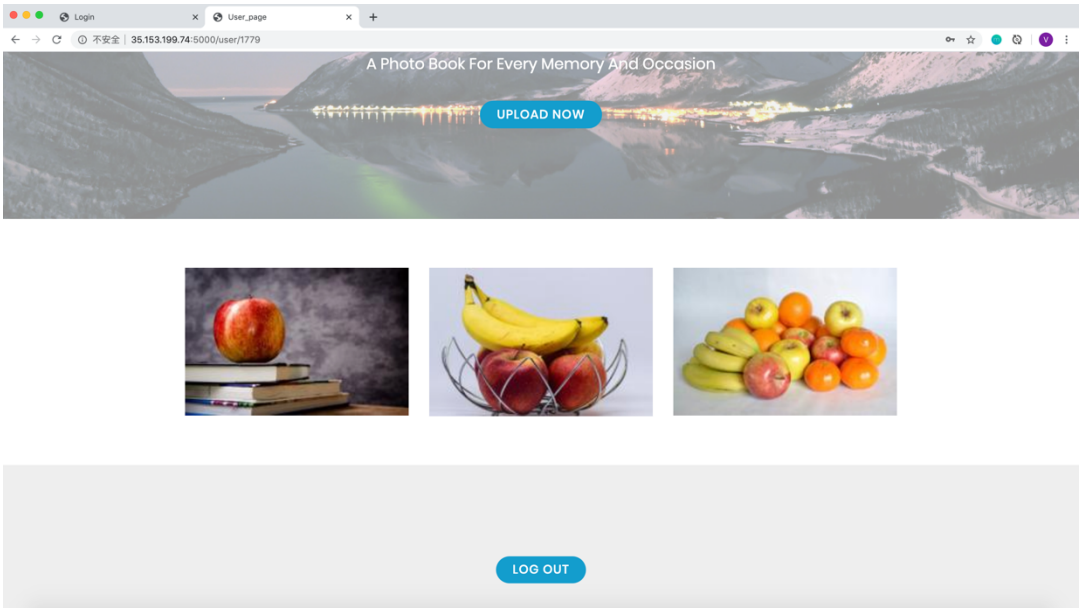
You can access your personal page when you have successfully logged in. You will see the message of “Welcome back + your username” on the top of your personal page to show that you have successfully logged in.

Personal page contains a gallery of the photos you have uploaded and two buttons.

Photo gallery: You will get a reminder of “You haven't uploaded any image” if you haven’t uploaded any photos. If you have already posted photos on the website, you will see the gallery of the thumbnails of your photos on the personal page.

Upload and Log out button: Click on the **Upload** button, you will jump to the upload page which enables you to upload a new photo. Click on the **Log out** button, you will directly log out your current account and back to the homepage of our website which is the Login page also.

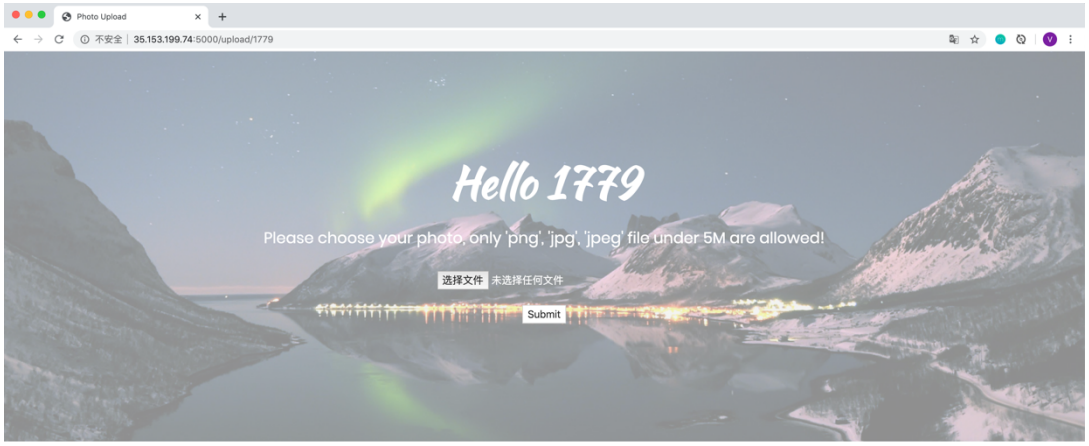




Personal Page

Upload page

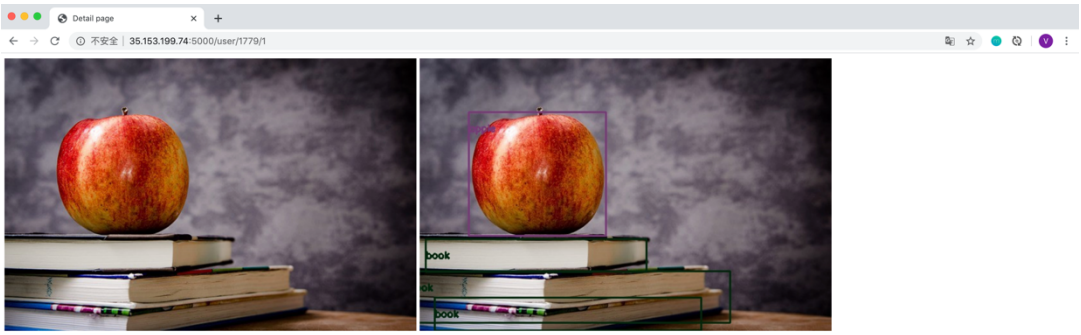
You can access to Upload page by click on the **Upload** button on the personal page. You will see the welcome of “Hello + your username” on the top of your upload page to show that you have successfully logged on the upload page. Please note that only 'png', 'jpg', 'JPG', 'PNG','JPEG’ are supported on this application. If the photo you provided are not supported the website will not upload the file and stay on the upload page until you upload a legal photo.
Choose file and Submit button: Click on the **Choose file** button**,** you can choose the photo you want to upload from your local document. Click on the **Submit** button, you will upload the photo you chose to our website.



Upload page

Detail page

You can access to the Detail page by click on the thumbnail in your personal page. The Detail page will display the original image and result of the object detection of this specific image.



Detail page

Developer’s Reference

Programming environment

Language: Python 3.7
Framework: Flask
IDE: Pycharm
Database: Sqlite
VNC client: Tiger
Program structure:

```
ECE1779Assignment1                                  # Directory for the whole project
├ static                                                    # Static file directory
│ ├ org_photo                                              # Raw picture directory
│ ├ det_photo                                          # Detected picture directory
│ └ thumbnail                                              # Thumbnail directory
│
├ templates                                                 # Webpage file directory
│ ├ detail_page.html
│ ├ login.html
│ ├ upload.html
│ ├ register.html
│ ├ upload_page.html
│ └ user_page.html
│
├ venv                                                # Virtual environment directory
├ app.py                                                                 # Main function
├ model.py                                                    # Create class for app.py
├ object_detection.py                          # The function for the object detection
├ savephoto.py                  # Functions for the operation of the table user_photo
├ thumbnail.py                                  # Functions to create the thumbnail
├ create_database.py                                                     # Init database
├ schema.sql                                    # Create tables in the database
├ database.db                                                              # Database
├ coco.names
├ yolo.py
```

```
├ yolov3.cfg
├ yolov3.weights
```

Inside Folder **statics**, **org_photo** contains the images that the user uploaded, **det_photo** contains the images with the detected objects and the **thumbnail** contains the thumbnail images which will display on the user page.
Folder **templates** contains all the .html files which are used to build the webpages.
The **app.py** is the function that implements most of the functionalities. Inside the file app.py, the main implementations are as follows:

Main program

Open Module

sqlite3	#a C-language library that implements SQL database engine
flask	#a micro web framework based on the Werkzeug and Jinja2.
flask_wtf&wtforms	#Operator that reads the user's input from the web page
os	#Operator that is used to obtain operating address in the system
OpenCV	#Generator that generates the detected photo
hashlib	#The module provides with algorithms to implement the hashes for passwords

Customized Module:

object_detection	#The function that read the original photo and conduct the object detection and write the detected photo to the aimed address
thumbnail	#The function that read the original photo and conduct the thumbnail transformation and write the thumbnail to the aimed address
savephoto	#The module that implement the operation of the table *user_photo* in the database

Customized Class

UploadForm( )	#Define a WTF form for reading the user's input in the personal upload page
User( )	#Define a Class for users to support the table *users* and operations of this table in the database
User_photo( )	#Define a Class for users' uploaded photo to support the table *user_photo* and operations of this table in the database

Customized Function

user_register( )	#The rooting function that render the register page and operates the input from register page

user_register_api( )	#The rooting function for TA to register automatically
user_login( )	#The rooting function that render the login page and operates the input from login page
user_logout( )	#The rooting function that enable users to log out and back to the login page
upload_photo( )	#The rooting function that render the upload page and operates the input from upload page
upload_photo_api( )	#The rooting function for TA to upload photos automatically
user_visit( )	#The rooting function that render the user page and display the photos user has uploaded as thumbnail
detail_function( )	#The rooting function that render the user detail page and display the original photo and detected photo
object_detection( )	#The function that implement the object detection (We use YOLO Object Detection as the API to implement object detection)
Thumbnail( )	#The function that implement the thumbnail transformation

Templates:

login	#Display the log in page (the home page of the website)
register	#Display the register page
user_page	#Display the user personal page with thumbnails
upload_page	#Display the upload page
detail_page	#Display the detail page with original photos and detected photos

Database

Database name: database

Table name: users

Column Name	Data type	Characteristic	Default
id	INTEGER	Primary key, Not Null, Auto increment	
name	STRING	Not Null	NULL
email	STRING	Not Null	NULL
password	STRING	Not Null	NULL
salt	STRING	Not Null	NULL
photo_id	STRING	Not NUll	NULL

Table name: user_photo

Column Name	Data type	Characteristic	Default
id	INTEGER	Primary key, Not Null, Auto increment	
name	STRING	Not Null	NULL
org_photo	STRING	Not Null	NULL
thumbnail	STRING	Not Null	NULL
det_photo	STRING	Not Null	NULL

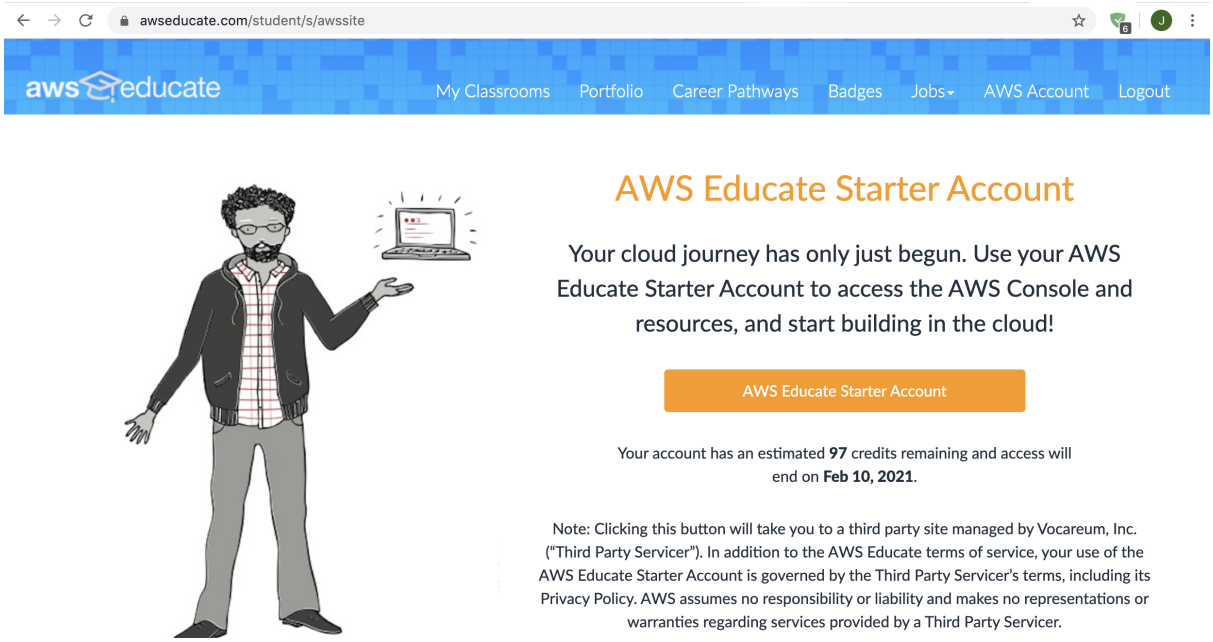
Two tables could be connected by query using id or name.

AWS

Log in: we are using the educational account provided in class. To login to the account, you can use:

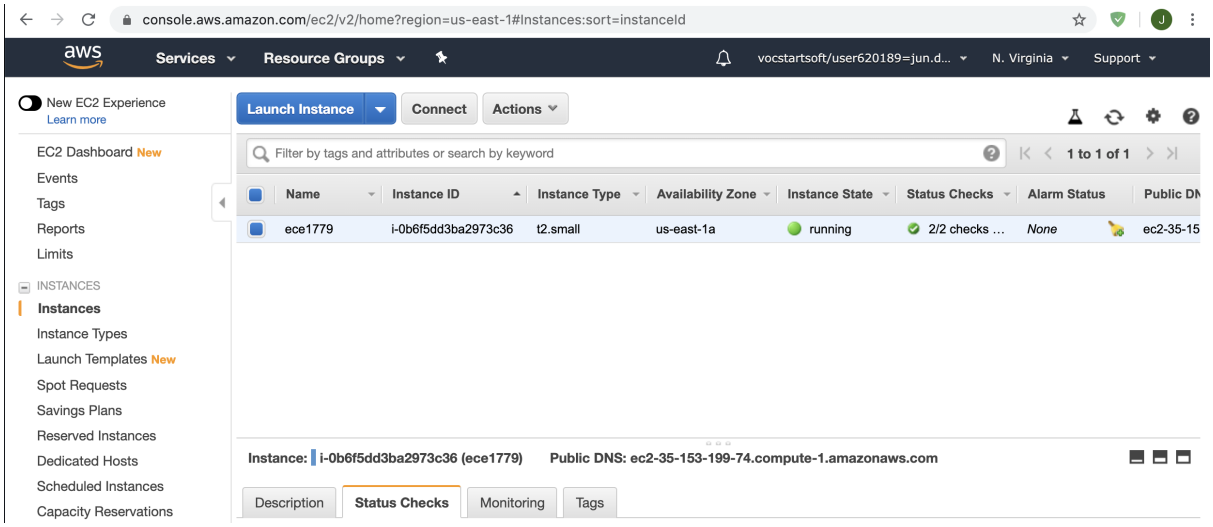
```
Account name: jun.deng@mail.utoronto.ca
Password: 960105Jd!
```

Once you logged in, you can choose **AWS account** on the top right corner and click on the **AWS Educate Starter Account** button.



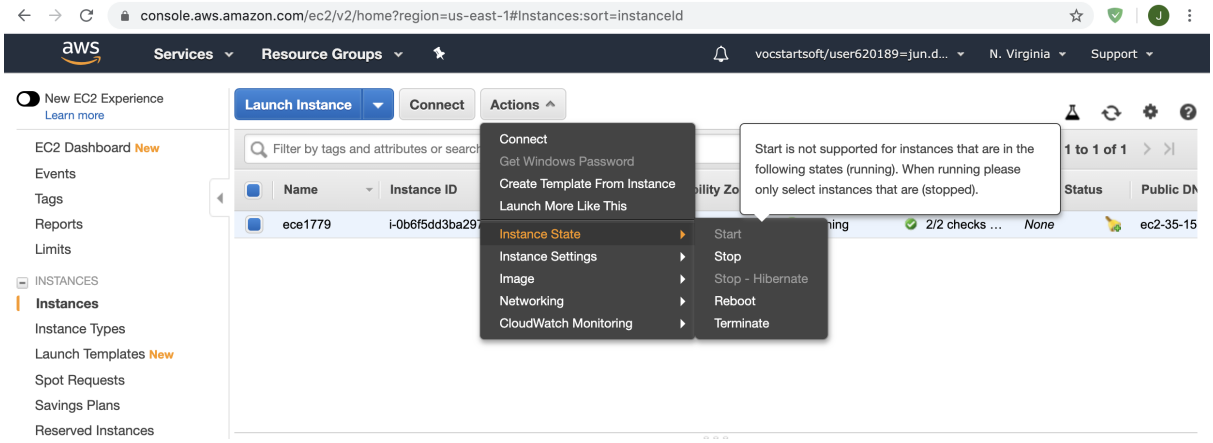
AWS educate starter account login page

Then in Vacareum welcoming page, click on **AWS console** and choose **EC2** in the following page. Clicking on the **Instances** on the left, then all the instances are displayed, as shown below.



Instance

The instance named **ece1779** is our instance we are using. To start this instance, just select this instance and go to **Actions** button. Then select Instance **State->Start** in the drop-down list, as shown below. Now the instance should be started.



Connect to the instance by SSH

If check on the aws account, we can see that the public IP in this instance is 35.153.199.74. Key-pair is inside the file ece1779_a1.pem. Knowing this information, we us the following command in the terminal to connect the server:

```
% ssh -i ece1779_a1.pem ubuntu@35.153.199.74 -L 5901 :localhost: 5901
```

To start running the program, just go to the Desktop and bash run the start.sh file as following:

```
ubuntu@ip-172-31-36-147: ~/Desktop$ bash start.sh
```

Test on load generator

The application has been tested on the load generator. The load generator file is "gen.py" in home directory. The command for the terminal console is python3.7 gen.py http://35.153.199.74:5000/api/upload 1779i 123 1 Downloads/pictures/ 10". The response of the terminal console is shown below:

```
ubuntu@ip-172-31-36-147:~$ python3.7 gen.py http://35.153.199.74:5000/api/upload
 1779i 123 1 Downloads/pictures/ 10
url: http://35.153.199.74:5000/api/upload, rate: 1.000000, files_folder: Downloa
ds/pictures/, n_uploads: 10
Uploaded: 1 files, responses: {}
Uploaded: 2 files, responses: {}
Uploaded: 3 files, responses: {}
Uploaded: 4 files, responses: {}
Uploaded: 5 files, responses: {}
Uploaded: 6 files, responses: {}
Uploaded: 7 files, responses: {'{"Success":"You have uploaded successfully."}\n'
: 2}
Uploaded: 8 files, responses: {'{"Success":"You have uploaded successfully."}\n'
: 2}
Uploaded: 9 files, responses: {'{"Success":"You have uploaded successfully."}\n'
: 2}
Uploaded: 10 files, responses: {'{"Success":"You have uploaded successfully."}\n
': 2}
Uploaded: 10 files, responses: {'{"Success":"You have uploaded successfully."}\n
': 2}
```