

FUN DAY WEATHER APP

Installation and test instructions

Prerequisites-

The application backend has been developed with Python 2.7.12 (should work with Python 2.7.12 and above, but not tested with Python 3).

It is recommended that you setup a python virtualenv. Following are the steps.

1. Install virtualenv

```
$ pip install virtualenv
Collecting virtualenv
```

Downloading

```
https://files.pythonhosted.org/packages/b6/30/96a02b2287098b23b875bc
8c2f58071c35d2efe84f747b64d523721dc2b5/virtualenv-16.0.0-py2.py3-non
e-any.whl (1.9MB)
```

```
100% |██████████████████████████████████████████████████████████████| 1.9MB 67kB/s
```

```
Installing collected packages: virtualenv
Successfully installed virtualenv-16.0.0
```

2. Create virtualenv

```
$ virtualenv ~/ENV2714
New python executable in /Users/mawagle/ENV2714/bin/python
Installing setuptools, pip, wheel...done.
```

Above command would pick up the system's default python interpreter for virtualenv. The virtualenv is created underneath ~/ENV2714 directory. Alternatively you could specify the python interpreter using -p option. E.g.:-

```
$ virtualenv -p /usr/local/bin/python2.7 ~/ENV2715
Running virtualenv with interpreter /usr/local/bin/python2.7
New python executable in /Users/mawagle/ENV2715/bin/python2.7
Also creating executable in /Users/mawagle/ENV2715/bin/python
Installing setuptools, pip, wheel...done.
```

3. Activate virtualenv

```
$ source ~/ENV2714/bin/activate
```

Above command should add a prefix (ENV2714) to your command line prompt.

Following is the link to the git repo for the code:-

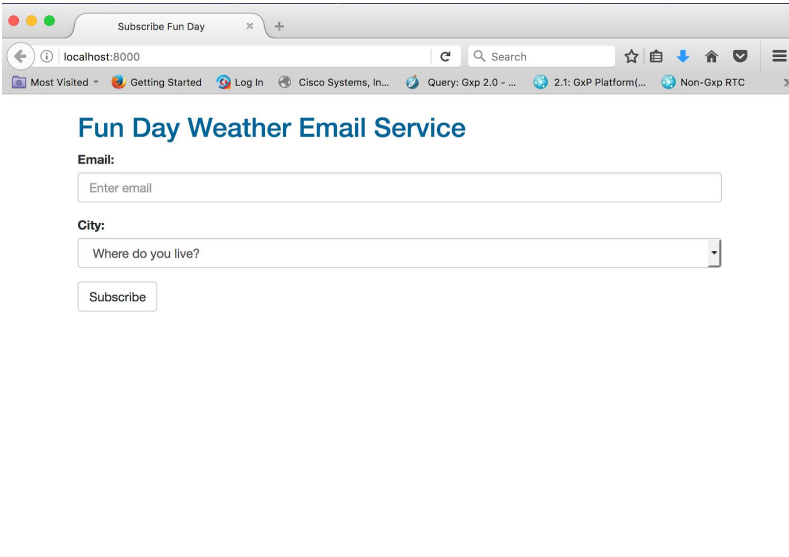
https://github.com/vaglomangirish/weather_update_app

Steps to run the application

1. Download the code using the following link:-
https://github.com/vaglomangirish/weather_update_app/archive/master.zip
2. Extract the zip to a directory that you prefer.
3. Change the working directory to the extracted directory
4. Change the directory to WeatherApp which is inside the extracted directory.
`$ cd WeatherApp`
5. Install the requirements packages
`$ pip install -r requirements.txt`
6. Change directory to src.
`$ cd src`
7. To Run the HTTP Server and the application, run the following:-
`$ python subscribe_api.py`
 - * Running on `http://0.0.0.0:8000/` (Press CTRL+C to quit)
 - * Restarting with stat
 - * Debugger is active!
 - * Debugger PIN: 647-257-729

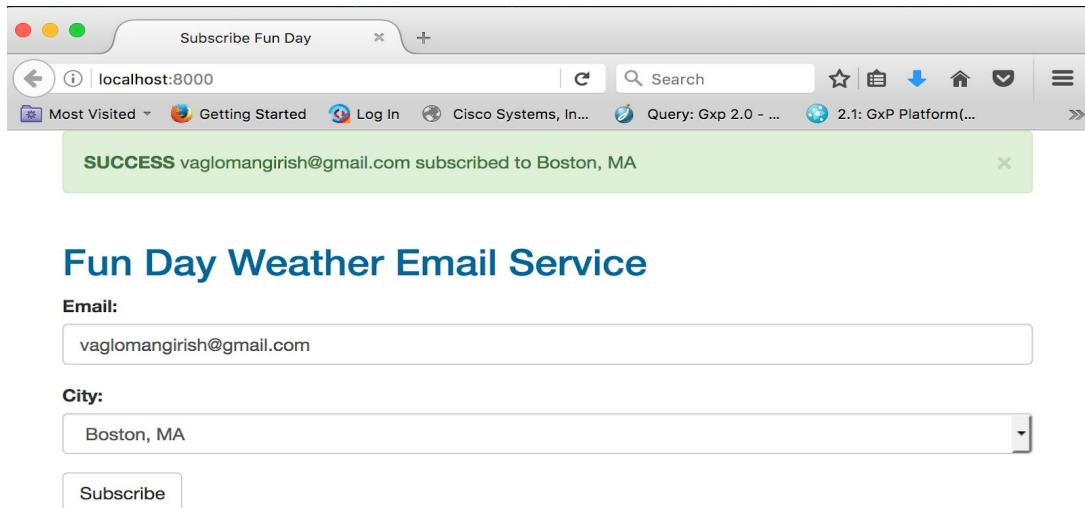
Server is now up and running. The server activity will be logged here.

8. Hit <http://localhost:8000> in web browser (Chrome/ Firefox) and you should see the following page.



The screenshot shows a web browser window with the title 'Subscribe Fun Day'. The address bar shows 'localhost:8000'. The page content includes the heading 'Fun Day Weather Email Service' in blue. Below the heading, there is a form with two input fields: 'Email:' with a placeholder 'Enter email' and 'City:' with a placeholder 'Where do you live?'. A 'Subscribe' button is located below the 'City' field. The browser's address bar and tabs are visible at the top.

9. Enter the email to subscribe and select the City and click Subscribe button. A message should appear stating the subscription was successful.



The screenshot shows a web browser window with the title "Subscribe Fun Day". The address bar displays "localhost:8000". A green success message banner at the top reads: "SUCCESS vaglomangirish@gmail.com subscribed to Boston, MA". Below this, the heading "Fun Day Weather Email Service" is displayed. The form includes an "Email:" label with a text input field containing "vaglomangirish@gmail.com", a "City:" label with a dropdown menu showing "Boston, MA", and a "Subscribe" button.

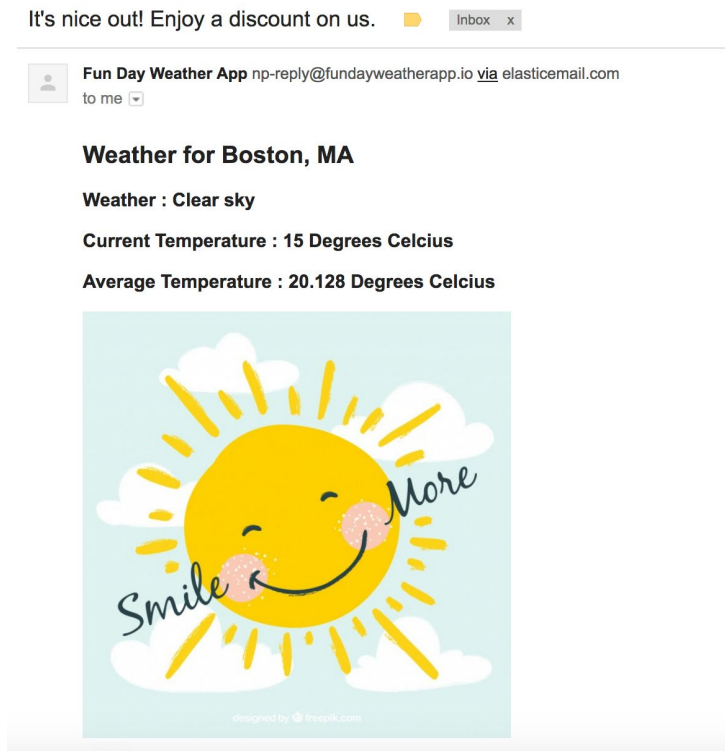
10. Repeat for as many locations and users.

11. To send the emails, do the following.

With current src directory run following:-

```
src $ python send_weather_emails.py
2018-06-11 21:19:40,042 - WeatherAppLogger - INFO - City: Boston, MA
Current Temperature: 15 Average Temperature: 20.128
2018-06-11 21:19:40,460 - WeatherAppLogger - INFO - Email sent to
vaglomangirish@gmail.com for city Boston, MA
```

12. You should receive an email on every subscribed id and for every city which should look similar to the following.



Running Unit tests

With current src directory run following:-

```
$ nosetests -v
Test to add record to store. ... 2018-06-12 14:46:07,151 -
WeatherAppLogger - INFO - data_store/store.json
2018-06-12 14:46:07,152 - WeatherAppLogger - INFO - Loaded the current
stored data.
2018-06-12 14:46:07,154 - WeatherAppLogger - INFO - Loaded the current
stored data.
2018-06-12 14:46:07,154 - WeatherAppLogger - INFO - Loaded the current
stored data.
ok
Test get records from store. ... 2018-06-12 14:46:07,155 -
WeatherAppLogger - INFO - data_store/store.json
2018-06-12 14:46:07,155 - WeatherAppLogger - INFO - data_store/store.json
2018-06-12 14:46:07,155 - WeatherAppLogger - INFO - Loaded the current
stored data.
2018-06-12 14:46:07,155 - WeatherAppLogger - INFO - Loaded the current
stored data.
```

2018-06-12 14:46:07,156 - WeatherAppLogger - INFO - Retrieved existing data dump

2018-06-12 14:46:07,156 - WeatherAppLogger - INFO - Retrieved existing data dump

ok

Test to send an email. ... ok

Test to get average temp. ... ok

Test to get weather. ... ok

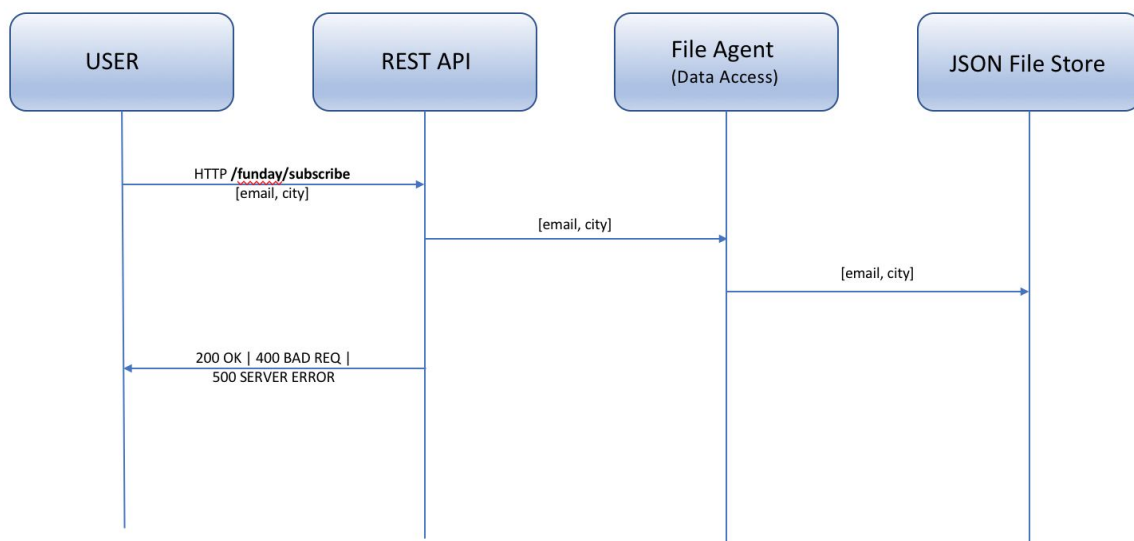
Test email content subject based on parameters. ... ok

Ran 6 tests in 1.834s

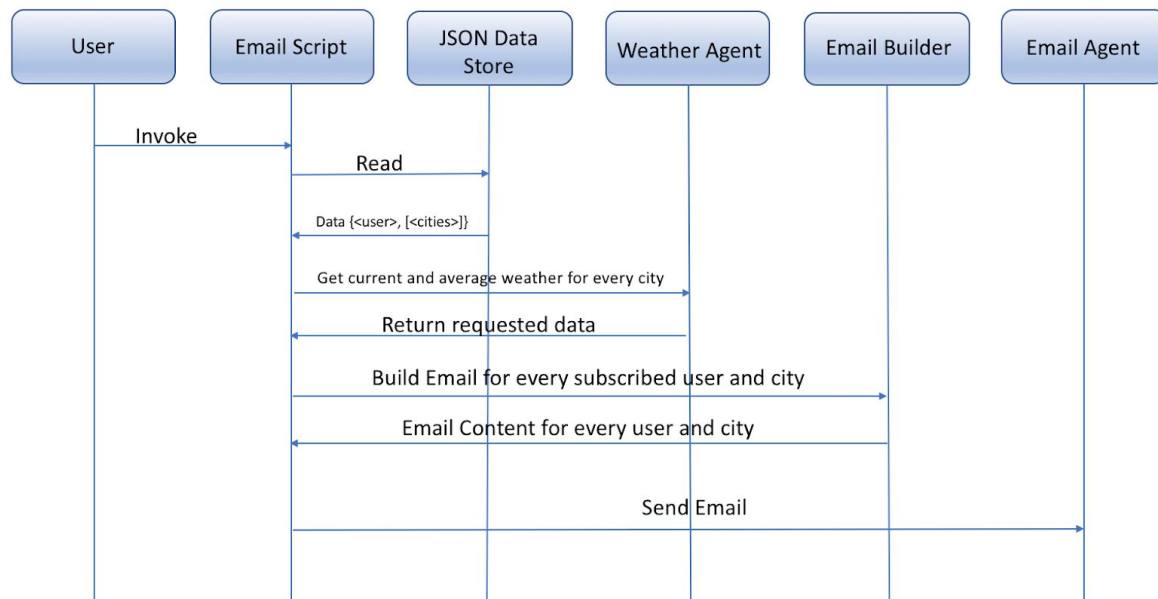
OK

Implementation Notes

Subscribe Workflow-



Send Email Workflow-



The backend has been developed with Python 2.7. The REST APIs have been developed with flask. The two REST APIs exposed are as follows:-

1) Subscription APIs

URL Pattern: **/funday/subscribe**

Form mandatory attributes:

"email": <email>

"city": <city>

Status code:

200 : Success

400: Bad Request

500: Server Error

2) API to retrieve top 100 populated cities

URL Pattern: **/funday/getcities**

Status code:

200 : Success

400: Bad Request

500: Server Error

The API for top 100 cities is used to populate the dropdown in the subscription form. The cities are not necessarily ordered by population. For better UX, it is ordered alphabetically. The list of cities is maintained in a json list at: WeatherApp/src/data_store/top_city_list.json

The list of cities has been retrieved from <http://www.city-data.com/top1.html>

The subscription data is persisted in a json data store file located in the project directory at the following path: WeatherApp/src/data_store/store.json

JSON store has been used for better structured data and avoiding portability issues and complexity in testing the application. For production purpose SQL database is recommended. JSON store is used solely for demonstration and convenience for this app.

To get weather updates, the [weatherbit-python](#) library has been used which in turn uses the [weatherbit API](#). This is the API which provides free current weather and historical weather data. Other APIs like OpenWeatherMap did not provide free historical API. Historical API has been used to find the average temperature.

For Email, the [Elastic Mail](#) REST API service has been used. The email html content is customized as per the weather parameter checks suggested. Every email will have an image which would reflect the mood of the weather.

Some more notes based on the suggested criteria

Security

Server and HTML client side validations have been added to validate the data passed to the API. The overall security could be improved by adding authentication and enabling TLS.

Re-Usability

Code has been modularized into following re-usable modules:-

- File_agent - Module to handle file json data store.
- Email_agent - Module to handle email sending.
- Weather_agent - Module to retrieve weather details.

Re-Inventing the Wheel?

The application uses a third party email service [Elastic mail](#) and weather api [Weatherbit](#). This saves a big deal of effort setting up your own email SMTP server or establishing and maintaining your own weather service.

Usability

Usability of this application could be improved if it is published as a widget within other website or application.

The email sending script could be plugged in to

Current Limitations

- Elastic Email service is currently restricted to 50 emails per day.
- Weatherbit API calls are restricted to 75 requests per hour.