# ==========
# README.md
# Author : Darshan Gadkari
# Created : Jun 2019
# ==========

#### Requirements
This was built and tested using MacOS but it should work well with any flavor of Linux or Windows
Python 3.6 or later but I am sure Python 3.* would work
Docker is optional
conda or virtualenv is optional
For testing the application:
    Browser (most popular browsers except for IE should work)
    Postman chrome extension or some similar tool

#### What test you completed?
Backend

#### What you'd have changed, if you had more time?
I would have added nginx and gunicorn to make the application scalable
I would have used React as the UI rather than just Flask Jinga2 templates
I would have built UI (or even React UI) to get postcodes within a specific radius
I would have done more error/exception handling
I would have used Flask werkzeug SimpleCache or MemcachedCache for repeat requests to the same postcode to avoid making external API calls
I would have added more documentation
I would have made Blueprint implementation using classes
Possibly added a few more test cases
I would have added security (example: jwt or even oath2)

#### What bits did you find the toughest? What bit are you most proud of? In both cases, why?
The only thing tough was I could not do the above items that I would have done if I had more time. If i had more time then the toughest part would have been React components
Why? - Because I am more comfortable with Python than React/JS
I am proud of the facts that I implemented
Blueprint
One API calling another
pandas (for joining data)
Basic MVC (Model, View/Template, Controller)
Why? - Because I love pandas and writing code in smaller chunks/modules

#### How can we improve this test?
This is a good test. One suggestion I can think of is to deploy this as a microservice to AWS lambda
I like using zappa for deploying Flask microservice to lambda

#### Running the Application
dockerfile in included
requirements.txt is included to cover dependent packages
Tests are written using unittest and are part of the build process
Python version: 3.6.2
I use conda for devlopment but this should work with virtualenv as well

To run:
> In linux shell navigate to `tailssubmission` folder
> If you are not using docker then run `pip install -r requirements.txt'
> In the linux shell, run `python app.py'

To see the application:
> In browser visit http://localhost:5000 to see the html rendering
> In browser visit http://localhost:5000/data to see the raw json
> Using Postman chrome extension visit http://localhost:5000/data with GET to see the raw json
> using Postman chrom extension visit http://localhost:5000/radius with POST, headers = {'Content-type': 'application/json'} to get list of postcodes within the 20 kilometers radius (default)
> using Postman chrom extension visit http://localhost:5000/radius with POST, headers = {'Content-type': 'application/json'} and add json

```
{
        "postcode": "N11 3PW",
        "radius": 10,
        "distance_in": "mi"
}
```

to get list of postcodes within the 10 miles radius of "N11 3PW"

To see the logs:
I have implemented logging. To tail the logs
> In linux shell navigate to `tailssubmission/logs` folder
> In the linux shell, run `tail -f app.log'

To run the tests:

> In linux shell navigate to `tailssubmission` folder
> In the linux shell, run `python -m unittest'

#### Application Structure
##### controllers Folder
Contains a Controller module with Controller class to implement MVC:
Controller

##### helpers Folder
Contains a set of helper modules with Helper classes:
Logging
LogHelper
PostCodeHelper

##### blueprints Folder
Contains a blueprint module that has one blueprint that contains 3 routes

##### static Folder
Contains css folder/file, header image and stores.json

##### settings Folder
Contains a .env file of the project

##### Templates Folder
Contains the html files for rendering

#### Final Notes
I thoroughly enjoy doing this test
I ran all .py files through pep8 checker to ensure PEP8 format is adhered too :)