

## Building Python Code

### App.py

```
import os
from pymongo import MongoClient
from flask import Flask, request, render_template
import requests
client = MongoClient('localhost', 27017)
db = client.login

login = db.users
# NOTE: you must manually set API_KEY below using information retrieved from your
IBM Cloud account.
API_KEY = "FDu8w9acEuLpZiojHIEoW5Rc2uHT9889GjnPT5QZ0-LN"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
app=Flask(__name__)
@app.route('/', methods=('GET','POST'))
def home():
    if request.method=='POST':
        email=request.form['em']
        uname=request.form['unme']
        password=request.form['pswd']
        login.insert_one({"email":email,"username":uname,"password":password})
        return render_template('login.html')
    return render_template('login.html')

@app.route('/sign')
def sign():
    return render_template('signup.html')

@app.route('/dash', methods=('GET','POST'))
def dashh():
    if request.method=='POST':
        coll=login.find()
        uname=request.form['uname']
```

```

password=request.form['psw']
for i in coll:
    if(uname==i['username'] and password==i['password']):
        return render_template('dashboard.html')
if request.method=='GET':
    return render_template('dashboard.html')

return render_template('login.html')

```

```

@app.route('/form')
def formm():
    return render_template('summa.html')

```

```

@app.route('/predict',methods=['POST'])
def predict():
    """
    For rendering results on HTML GUI
    """

```

```

b=int(request.form["month"])
c=request.form["daym"]
d=request.form["dayw"]
e=request.form["fnum"]
f=int(request.form["airport"])
g=int(request.form["airportd"])
h=request.form["dtime"]
i=request.form["atime"]
j=request.form["ttime"]
if b==1 or b==2 or b==3:
    a=1
    l=2
elif b==4 or b==5:
    a=2
    l=3
elif b==6:
    a=2
    l=0
elif b==7 or b==8:
    a=3
    l=0

```

```
elif b==9:
    a=3
    l=1
elif b==10 or b==11:
    l=1
    a=4
elif b==12:
    a=4
    l=2
ff=f
gg=g
if ff==gg:
    return render_template('summa.html', prediction_text='No delay(same airport!)')
if ff<gg:
    ff,gg=gg,ff
if gg==1 and ff==2:
    k=594
elif gg==1 and ff==3:
    k=760

elif gg==1 and ff==4:
    k=907

elif gg==1 and ff==5:
    k=2182

elif gg==2 and ff==3:
    k=509

elif gg==2 and ff==4:
    k=528

elif gg==2 and ff==5:
    k=1927

elif gg==3 and ff==4:
    k=1029

elif gg==3 and ff==5:
    k=2422
```

```
elif gg==4 and ff==5:
```

```
    k=1399
```

```
#print (a,b,c,d,e,f,g,h,i,j,k,l)
```

```
    payload_scoring = {"input_data": [{"field":  
[["QUARTER","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","FL_NUM","ORIGIN","  
DEST","CRS_DEP_TIME","CRS_ARR_TIME","CRS_ELAPSED_TIME","DISTANCE","S  
EASON"]], "values": [[a,b,c,d,e,f,g,h,i,j,k,l]]}]}
```

```
    response_scoring =
```

```
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/7d6c3b49-ec70-4cf  
e-ab88-4f6dbe6a7997/predictions?version=2022-11-16', json=payload_scoring,
```

```
    headers={'Authorization': 'Bearer ' + mltoken})
```

```
#print("Scoring response")
```

```
predictions=response_scoring.json()
```

```
m=predictions['predictions'][0]['values'][0][0]
```

```
if m==0:
```

```
    return render_template('pred.html', prediction_text='No delay is predicted to  
happen. HAVE A NICE FLIGHT!!')
```

```
elif m==1:
```

```
    return render_template('pred.html', prediction_text='Delay in flight departure is  
predicted to happen')
```

```
elif m==2:
```

```
    return render_template('pred.html', prediction_text='Delay in both flight departure  
and arrival is predicted to happen')
```

```
elif m==3:
```

```
    return render_template('pred.html', prediction_text='Flight is predicted to get  
Diverted')
```

```
elif m==4:
```

```
    return render_template('pred.html', prediction_text='Flight is predicted to get  
Cancelled!')
```

```
else:
```

```
    return render_template('pred.html', prediction_text='output {}'.format(m))
```

```
if __name__ == "__main__":
```

```
os.environ.setdefault('FLASK_ENV', 'development')
app.run(debug=False)
```

Working:

```
In [4]: runfile('C:/Users/manis/OneDrive/Desktop/ibm working/app.py', wdir='C:/Users/manis/OneDrive/
Desktop/ibm working')
* Serving Flask app "app" (lazy loading)
* Environment: development
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [18/Nov/2022 19:22:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 19:22:59] "GET /static/flight.jpg HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 19:23:05] "POST /dash HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 19:23:05] "GET /static/relation.jpg HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 19:23:05] "GET /static/corr.jpg HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 19:23:05] "GET /static/reason.jpg HTTP/1.1" 304 -
```