# ADVANCES IN DATA SCIENCE/ARCHITECTURE

## ASSIGNMENT-2

## REPORT

**Submitted By:**

**MEGHA SINGH**

**SNEHA MALSHETTI**

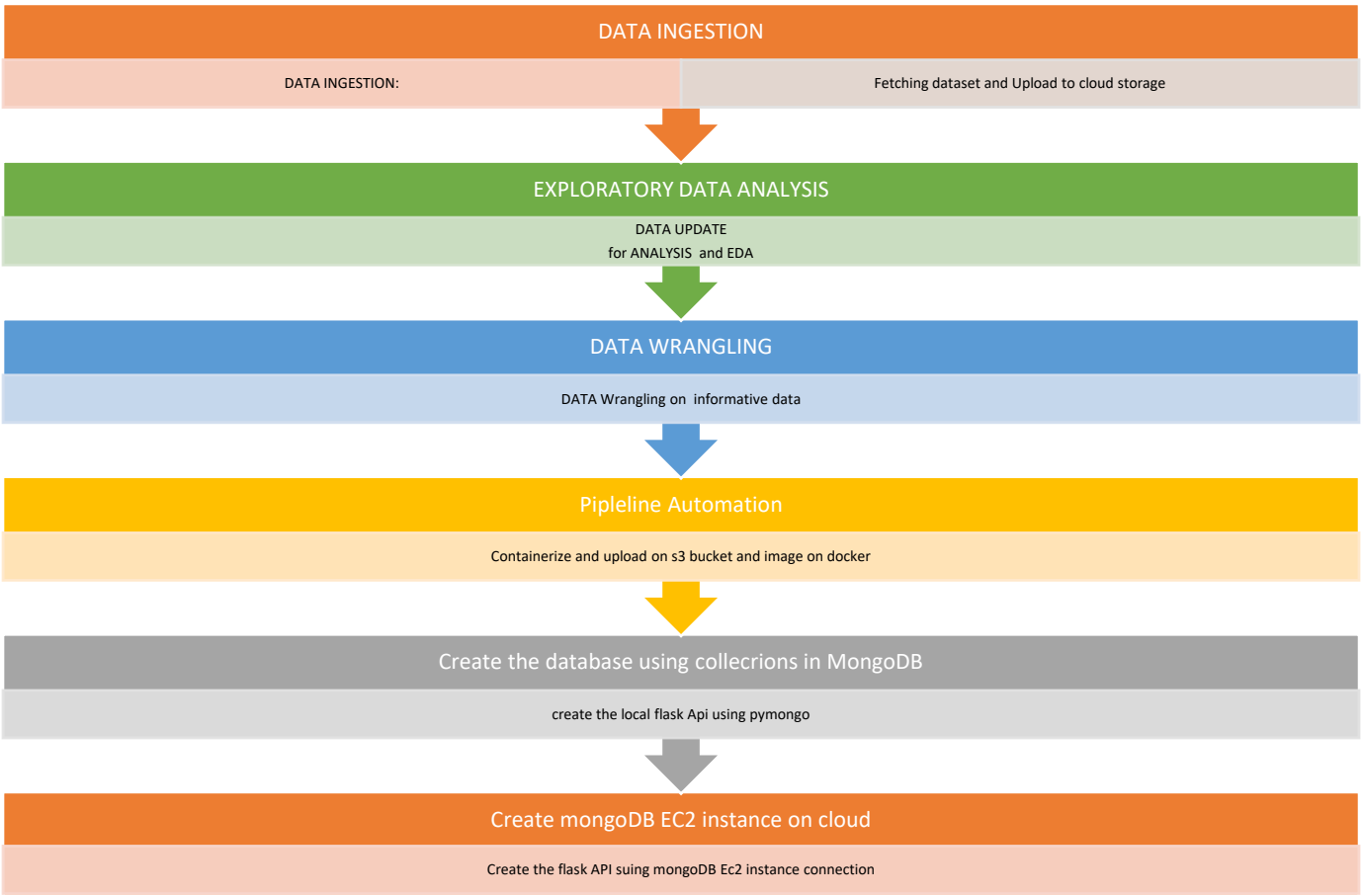
Table of Contents

## Overview

# 8. Objective Flow of the Project

1. Language Used: Python
2. Data Ingestion, EDA, Data Wrangling: Jupyter NoteBook, Boto3 connection, Amazon S3 bucket
3. Cloud: Amazon EC3 instance with MongoDB.
4. REST API for Daas: Flask with MongoDB
5. Database as service used: MongoDB (NOSQL)

## 2.1 Flowchart of Processing of Data Analysis



# 2. DATA INGESTION, EDA and Data Wrangling

## DATA INGESTION

**Descriptive Steps and script for DATA INGESTION**

1. Download the data from Zillow. (https://www.kaggle.com/c/zillow-prize-1)

Use the Jupyter Notebook for using the config

{ "AWSAccess":"access key",

"AWSSecret":"secret key"} to upload the cleaned data on S3 bucket.

**2. Create an IPYB notebook and Conduct an in-depth EDA**

## Exploratory Data Analysis and DATA WRANGLING

### Filling the Missing data

1. First merge the train Csv and properties csv to get all the available data and covert it into a dataframe using pandas

In [5]:
```
train_df = pd.read_csv("train_2016_v2.csv", parse_dates=["transactiondate"])
train_df.shape

# merge train data with properties CSV
train_df =pd.merge(train_df, rawdataspecificrows, on ='parcelid', how = 'outer')
train_df.shape


# train_df.to_csv('mergeddata.csv', index = False)

print ("successfully saved as a CSV file")
```

```
successfully saved as a CSV file
```

2. From the data Set we have made the analysis
   Column : finishedsquarefeet50 and finishedfloor1squarefeet have all the same values so we have merged column into one to get the single column with complete values of the finished squarefeet

Similarly,
Columns finishedsquarefeet6","finishedsquarefeet12","finishedsquarefeet13", "finishedsquarefeet15", "finishedsquarefeet50", "finishedfloor1squarefeet","calculatedfinishedsquarefeet" All have the same values after merging them we get the one column value.

In [7]:
```
rawdata.finishedsquarefeet50.fillna(rawdata.finishedfloor1squarefeet, inplace=True)



rawdata[["finishedsquarefeet6", "finishedsquarefeet12","finishedsquarefeet13","finishedsquarefeet15",
                 "finishedsquarefeet50", "finishedfloor1squarefeet","calculatedfinishedsquarefeet"]]

count = rawdata[["finishedsquarefeet6", "finishedsquarefeet12","finishedsquarefeet13","finishedsquarefeet15",
                 "finishedsquarefeet50", "finishedfloor1squarefeet","calculatedfinishedsquarefeet"]].count(axis=0)
print ("Count after merging finishedsquarefeet50 and finishedfloor1squarefeet :",count)
# print ("The number of values in the column are : ",)
```

```
Count after merging finishedsquarefeet50 and finishedfloor1squarefeet : finishedsquarefeet6              22003
finishedsquarefeet12            2709295
finishedsquarefeet13               7672
finishedsquarefeet15             190807
finishedsquarefeet50             202723
finishedfloor1squarefeet         202723
calculatedfinishedsquarefeet    2929774
dtype: int64
```

3. Replacing "**airconditioningtypeid"** to 5 default type for none
As the data dictionary has explained the apartments with no AC or other: having column valus as 5. So we replace all the missing values with 5.

Out[83]:

| | parcelid | airconditioningtypeid | archit |
|---|---|---|---|
| **0** | 11016594 | 1 | |
| **1** | 14366692 | 5 | |
| **2** | 12098116 | 1 | |
| **3** | 12643413 | 1 | |
| **4** | 14432541 | 5 | |

5 rows × 48 columns

4. Similarly Replacing "**architecturalstyletypeid**" to 19 default type for other for filling the missing or blank data.

Out[84]:

| | parcelid | airconditioningtypeid | architecturalstyletypeid | bas |
|---|---|---|---|---|
| **0** | 11016594 | 1 | 19 | |
| **1** | 14366692 | 5 | 19 | |
| **2** | 12098116 | 1 | 19 | |
| **3** | 12643413 | 1 | 19 | |
| **4** | 14432541 | 5 | 19 | |

5 rows × 48 columns

5. Replacing basementsqft to 0 default square feet
   Which shows the apartment is not having the basement and converting the column value to integer type

Out[85]:

| | parcelid | airconditioningtypeid | architecturalstyletypeid | basementsqft |
|---|---|---|---|---|
| **0** | 11016594 | 1 | 19 | 0 |
| **1** | 14366692 | 5 | 19 | 0 |
| **2** | 12098116 | 1 | 19 | 0 |
| **3** | 12643413 | 1 | 19 | 0 |
| **4** | 14432541 | 5 | 19 | 0 |

5 rows x 48 columns

6. Replacing the bedroom missing count with 0 as no data is available for the same.

]:
```
# replacing bedroom to 0 default square feet
rawdata.bedroomcnt.fillna(0, inplace=True)
rawdata.bedroomcnt=rawdata.bedroomcnt.astype(int)
rawdata.bedroomcnt.head(5)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\gener
A value is trying to be set on a copy of a slice from a Data

See the caveats in the documentation: http://pandas.pydata.o
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\gener
A value is trying to be set on a copy of a slice from a Data
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.o
  self[name] = value
```

]:
```
0    3
1    4
2    2
3    2
4    4
```

7. Replacing "**buildingclasstypeid"** missing values to 5: default other
   The building default type is the structure type of the building so if the data is missing we replace it with other value.

8. Now to check id the **decktypeid** is having any unique values of count for the same
   Listing unique values in col

```
In [88]:
#listing unique values in col
rawdata.decktypeid.unique()

Out[88]: array([ nan,  66.])
```

   The column is giving the information about the deck availibilty, so we can only put a flag as TRUE or FALSE in the column.
   So , We can remove the integer value as True and rest black or 0 as False

```
In [15]:
rawdata.groupby('decktypeid')['parcelid'].count()

Out[15]: decktypeid
         66.0    17096
         Name: parcelid, dtype: int64
```

9. We want to calculate the available square feet of the property
   But for the same we have still two columns So We compare the higher value from both column as that will be the final or finished available square feet including the utilities such as deck, pool and other facilities:
   So after comparing Finishedsquarefet50 is having the higher value as it inclused the **deckquarefeet** and **poolsizesquarefeet** as well.

   So further analysis we will use this column for the and to fill the missing column value we can merge the finished floorsquarefeet column.
   With this we will get the square feet area of properties. And than delete the column

```
In [18]:
# check which has the higher value
print ("finishedsquarefeet50 sum = ",rawdata["finishedsquarefeet50"].sum())

print ("finishedfloor1squarefeet sum = ",rawdata["finishedfloor1squarefeet"].sum())

finishedsquarefeet50 sum =  281572505.0
finishedfloor1squarefeet sum =  279887079.0

In [19]:
rawdata.shape

Out[19]: (2985342, 48)

In [20]:
# dropping column finishedfloor1squarefeet

del rawdata['finishedfloor1squarefeet']

rawdata.shape

Out[20]: (2985342, 47)
```

10. Converting the  Squarefeetvalue of the properties into integer type

```
rawdata.finishedsquarefeet50.fillna(0, inplace=True)
rawdata.finishedsquarefeet50=rawdata.finishedsquarefeet50.astype(int)
rawdata["finishedsquarefeet50"].head(5)
rawdata.groupby('finishedsquarefeet50')['parcelid'].count()

```

11.

In [93]:

```
rawdata.fips.fillna(0, inplace=True)
rawdata.fips=rawdata.fips.astype(int)
rawdata["fips"].head(5)
# print (rawdata.groupby('fips')['parcelid'].count())
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ge
A value is trying to be set on a copy of a slice from a D

See the caveats in the documentation: http://pandas.pydat
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ge
A value is trying to be set on a copy of a slice from a D
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydat
  self[name] = value
```

Out[93]:

```
0    6037
1    6059
2    6037
3    6037
4    6059
Name: fips, dtype: int32
```

12. Now we can calculate if the column is having relevant number of values else we can remove it from data set

In [94]:

```
rawdata.groupby('fips')['parcelid'].count()
```

Out[94]:

```
fips
0          11437
6037     2009443
6059      741603
6111      222859
Name: parcelid, dtype: int64
```

13. Check the number of properties with Fireplace availibity

In [27]:
```
rawdata.fireplacecnt.fillna(0, inplace=True)
rawdata.fireplacecnt=rawdata.fireplacecnt.astype(int)
rawdata["fireplacecnt"].head(5)
rawdata.groupby('fireplacecnt')['parcelid'].count()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:3660:
```

Out[27]:
```
fireplacecnt
0    2672695
1     269651
2      34409
3       7696
4        710
5        126
6         32
7         15
8          2
9          6
Name: parcelid, dtype: int64
```

14. Check the number of ¾ baths and filling the missing values

```
rawdata.fullbathcnt.fillna(0, inplace=True)
rawdata.fullbathcnt=rawdata.fullbathcnt.astype(int)
rawdata["fullbathcnt"].head(5)
rawdata.groupby('fullbathcnt')['parcelid'].count()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generi
A value is trying to be set on a copy of a slice from a DataF

See the caveats in the documentation: http://pandas.pydata.or
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generi
A value is trying to be set on a copy of a slice from a DataF
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.or
  self[name] = value
```

]:
```
fullbathcnt
0      128918
1      544951
2     1425717
3      660167
4      151012
```

15. Getting the number of properties with Garage Count

```
rawdata.garagecarcnt.fillna(0, inplace=True)
rawdata.garagecarcnt=rawdata.garagecarcnt.astype(int)
rawdata["garagecarcnt"].head(5)
rawdata.groupby('garagecarcnt')['parcelid'].count()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\gen
A value is trying to be set on a copy of a slice from a Da

See the caveats in the documentation: http://pandas.pydata
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\gen
A value is trying to be set on a copy of a slice from a Da
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata
  self[name] = value
```

[31]:
```
garagecarcnt
0     2116041
1      177587
2      660492
3       19635
4        8495
5        1705
6         575
7         266
8         181
```

16. Number of hot tub and spa And for better analysis changing it to flag value as True or false.. ie number of total properties with or without

# getting the hot tub and spa in the building

[33]:
```

rawdata.hashottuborspa.fillna("False", inplace=True)
rawdata["hashottuborspa"].head(5)
rawdata.groupby('hashottuborspa')['parcelid'].count()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:3660:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
  self._update_inplace(new_data)
```

t[33]:
```
hashottuborspa
True       69016
False    2916326
Name: parcelid, dtype: int64
```

17. Getting the heating systems availibilty else filling the missing value as 13(other )

```
rawdata.heatingorsystemtypeid.fillna(13, inplace=True)
rawdata.heatingorsystemtypeid=rawdata.heatingorsystemtypeid.astype(int)
rawdata["heatingorsystemtypeid"].head(5)
rawdata.groupby('heatingorsystemtypeid')['parcelid'].count()
```

```
Out[34]: heatingorsystemtypeid
1          262
2      1156876
6        27482
7       595478
10          39
11          16
12          25
```

18. Getting the numer properties with pool count and changing it to flag

```
rawdata['poolcnt']=rawdata['poolcnt'].notnull()

rawdata['poolcnt'].head(5)

rawdata.groupby('poolcnt')['parcelid'].count()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_
A value is trying to be set on a copy of a slice fro
Try using .loc[row_indexer,col_indexer] = value inst

See the caveats in the documentation: http://pandas.
  This is separate from the ipykernel package so we
```

```
Out[35]: poolcnt
False    2467783
True      517559
Name: parcelid, dtype: int64
```

19. Checking the region with no region defined

```
rawdata.regionidcounty.fillna(0, inplace=True)
rawdata.regionidcounty=rawdata.regionidcounty.astype(int)
rawdata["regionidcounty"].head(5)
rawdata.groupby('regionidcounty')['parcelid'].count()
```

```
Out[37]: regionidcounty
         0          11437
         1286      741603
         2061      222859
         3101     2009443
         Name: parcelid, dtype: int64
```

20 . Check the number of rooms per property

## check the number of rooms per property

```
In [40]: # roomcnt

         ## sum of rooms (((((check)))))

         rawdata.roomcnt.fillna(0, inplace=True)
         rawdata.roomcnt=rawdata.roomcnt.astype(int)
         rawdata["roomcnt"].head(5)
         rawdata.groupby('roomcnt')['parcelid'].count()
```

```
Out[40]: roomcnt
         0      2320450
         1           77
         2          751
         3         6289
         4        42324
         5        99587
         6       178367
         7       156400
         8       120233
         9        45242
         10       10862
         11        3021
         12        1097
         13         313
         14         156
         15          67
         16          35
         17           9
         18          22
         19           6
         20           3
```

## typeconstructiontypeid and filling the missing value

20.

In [41]:

```
# # typeconstructiontypeid
rawdata.typeconstructiontypeid.fillna(14, inplace=True)
rawdata.typeconstructiontypeid=rawdata.typeconstructiontypeid.astype(int)
rawdata["typeconstructiontypeid"].head(5)
rawdata.groupby('typeconstructiontypeid')['parcelid'].count()
```

21. Checking the year built of the house

## yearbuilt

In [45]:

```
# yearbuilt

rawdata.yearbuilt.fillna(0, inplace=True)
rawdata.yearbuilt=rawdata.yearbuilt.astype(int)
rawdata["yearbuilt"].head(5)
rawdata.groupby('yearbuilt')['parcelid'].count()

```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\cor
A value is trying to be set on a copy of a slice from

See the caveats in the documentation: http://pandas.p
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\cor
A value is trying to be set on a copy of a slice from
Try using .loc[row_indexer,col_indexer] = value inste

See the caveats in the documentation: http://pandas.p
  self[name] = value
```

Out[45]:

```
yearbuilt
0       59931
1801        3
1805        1
```

22. Checking the number of stories in properties and filling the missing values

In [46]:
```
# numberofstories

rawdata.numberofstories.fillna(0, inplace=True)
rawdata.numberofstories=rawdata.numberofstories.astype(int)
rawdata["numberofstories"].head(5)
rawdata.groupby('numberofstories')['parcelid'].count()

```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pa
  self._update_inplace(new_data)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py
A value is trying to be set on a copy of a slice from a DataFrame
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pa
  self[name] = value
```

Out[46]:
```
numberofstories
0     2303243
1      424205
2      242135
3       15679
4          58
5          11
```

23. Checking the values with zero values of latitude and ongitude and removing them

```
# rawdata = rawdata[np.isfinite(rawdata['EPS'])]
print (rawdata.shape)


rawdata = rawdata[np.isfinite(rawdata['latitude'])]
rawdata = rawdata[np.isfinite(rawdata['longitude'])]
print (rawdata.shape)
```

```
(2973905, 47)
(2973905, 47)
```

24. Save the file using pandas into CSV format on local as well on S3 bucket

```
c = boto.connect_s3(AWSAccess1, AWSSecret1)
conn = S3Connection(AWSAccess1, AWSSecret1)
bucket = c.get_bucket('team8njassignment2')
b = c.get_bucket(bucket, validate=False)

    k = Key(b)
    k.key = fname
    k.content_type = r.headers['content-type']
    k.set_contents_from_string(r.content)
    print('successfully uploaded to s3')


    rawdata.to_csv('wrangleddata.csv', index=False)
    print ("successfully saved as a CSV file")
```

```
successfully saved as a CSV file
```

## 4. MongoDB on EC2 instance

### Working with mongoDB and creating the collection of dataset

- Install Mongodb, Then using the CMD in the mongo bin directory
- Run the command:
  mongod
- Import the CSV file in mongo
- Run the command to open CSV file using CSV

mongoimport --db propertiesDB --collection users --type csv --headerline --file C:\Users\singh\Desktop\prop.csv

```
C:\Users\singh\Desktop\mongo\bin>mongoimport --db propertiesDB --collection users --type csv --headerline --file C:\Us
s\singh\Desktop\prop.csv
2017-07-05T00:47:24.765-0400    connected to: localhost
2017-07-05T00:47:27.759-0400    [........................] propertiesDB.users   6.52MB/619MB (1.1%)
2017-07-05T00:47:30.759-0400    [........................] propertiesDB.users   13.5MB/619MB (2.2%)
2017-07-05T00:47:33.760-0400    [........................] propertiesDB.users   20.6MB/619MB (3.3%)
2017-07-05T00:47:36.758-0400    [#.......................] propertiesDB.users   27.6MB/619MB (4.5%)
2017-07-05T00:47:39.759-0400    [#.......................] propertiesDB.users   34.1MB/619MB (5.5%)
2017-07-05T00:47:42.759-0400    [#.......................] propertiesDB.users   40.3MB/619MB (6.5%)
2017-07-05T00:47:45.758-0400    [#.......................] propertiesDB.users   47.2MB/619MB (7.6%)
2017-07-05T00:47:48.758-0400    [##......................] propertiesDB.users   54.1MB/619MB (8.7%)
2017-07-05T00:47:51.759-0400    [##......................] propertiesDB.users   59.4MB/619MB (9.6%)
2017-07-05T00:47:54.759-0400    [##......................] propertiesDB.users   66.3MB/619MB (10.7%)
2017-07-05T00:47:57.758-0400    [##......................] propertiesDB.users   73.3MB/619MB (11.8%)
2017-07-05T00:48:00.758-0400    [###.....................] propertiesDB.users   80.0MB/619MB (12.9%)
2017-07-05T00:48:03.764-0400    [###.....................] propertiesDB.users   87.1MB/619MB (14.1%)
2017-07-05T00:48:06.759-0400    [###.....................] propertiesDB.users   94.2MB/619MB (15.2%)
2017-07-05T00:48:09.773-0400    [###.....................] propertiesDB.users   101MB/619MB (16.4%)
2017-07-05T00:48:12.760-0400    [####....................] propertiesDB.users   108MB/619MB (17.5%)
2017-07-05T00:48:15.758-0400    [####....................] propertiesDB.users   115MB/619MB (18.6%)
2017-07-05T00:48:18.758-0400    [####....................] propertiesDB.users   122MB/619MB (19.7%)
2017-07-05T00:48:21.758-0400    [####....................] propertiesDB.users   129MB/619MB (20.8%)
2017-07-05T00:48:24.759-0400    [#####...................] propertiesDB.users   136MB/619MB (22.0%)
```

# 5. Create the REST API to serve the Data



## Using Flask to create REST API on Cloud

Step1. Import all the Flask dependencies

We use Mongo_client and Flask , request to connect with the Ec2 Server

```
client3 = MongoClient("mongodb://54.166.125.102")
print(client3)
db = client3['zillowdb']
```

Step2. Now connect the Ec2 inctance usnting the username and link for the MongDB Instance

```
app = Flask(__name__)

app.config['MONGO_DBNAME'] = 'zillowdb'
app.config['MONGO_URI'] = 'mongodb://54.166.125.102:27017/zillowdb'
app.config['MONGO3_HOST'] = 'ec2-54-166-125-102.compute-1.amazonaws.com'
app.config['MONGO3_PORT'] = 27017

mongo = PyMongo(app)
# client = MongoClient("mongodb://52.87.172.158")
# print(client)
```

Step3. This routes to the index.html page we have created for the local host to run and show output:

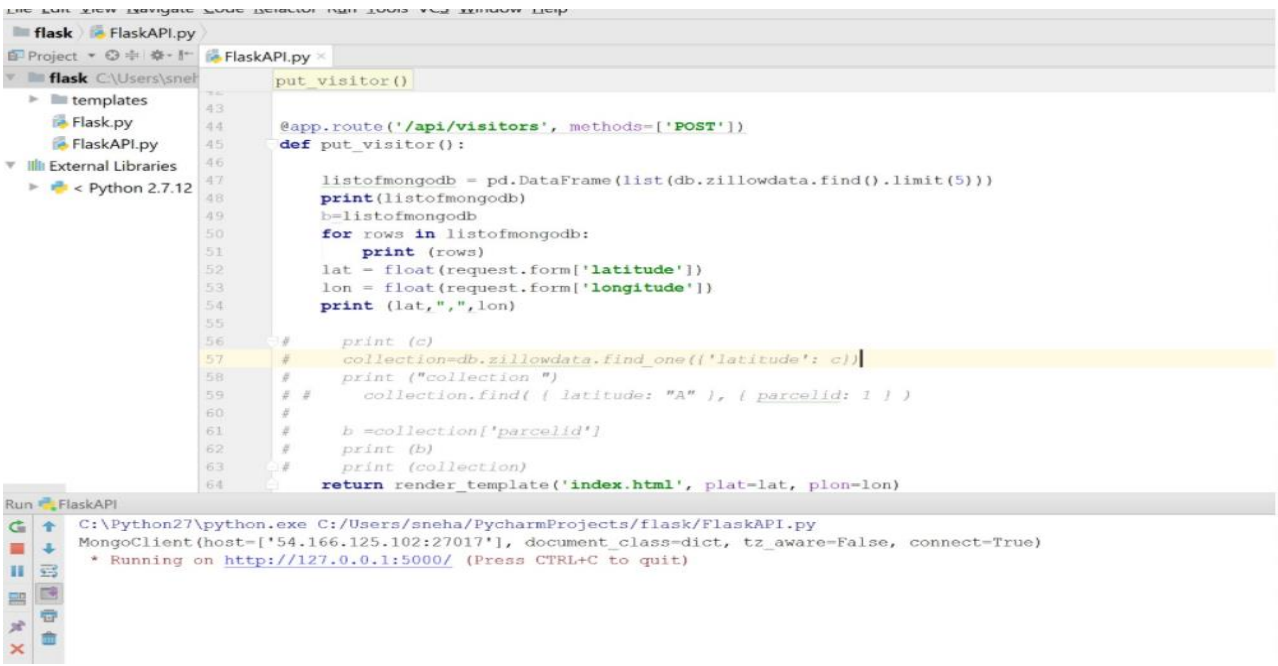
```
def home():
    print ("index")
    return render_template('index.html')



@app.route('/api/visitors', methods=['POST'])
def put_visitor():

    listofmongodb = pd.DataFrame(list(db.zillowdata.find().limit(5)))
    print(listofmongodb)
    b=listofmongodb
    for rows in listofmongodb:
        print (rows)
    lat = float(request.form['latitude'])
    lon = float(request.form['longitude'])
    print (lat,",",lon)
```

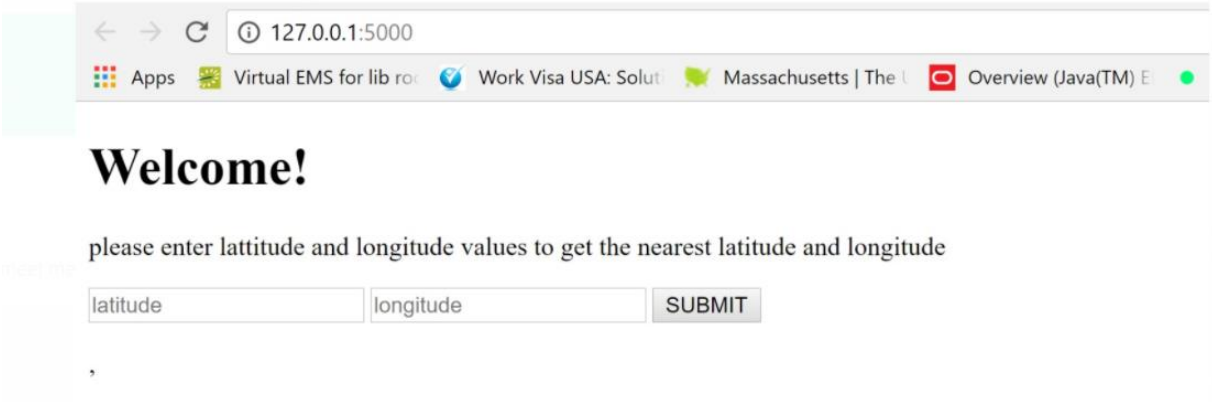
Step 5. This will return the data on the local host on the defines port:

```
    return render_template('index.html', plat=lat, plon=lon)



f __name__ == '__main__':

    port = int(os.environ.get('PORT', 5000))
      app.run(host='0.0.0.0', port=port, debug=true)
    app.run()
```

Step 6 :

The output on running the code wil be like :






This Application is fetching rthe EC2 cloud instance of MongoDb for 10 nearest values

For the same we used the MongoDB NO SQL Querries:

**Using mongodb e to search nearest values for lattitude and longitude**

For same we No SQL query to get the result:

For exact value of longitude and latitude :

Db.wrangleddata.find({ $or:[ {“latitude”:” ” }, {longitude:””}] })

For 10 nearby properties based on  lat and long:

Db.wrangleddata.find({“lat”: {$gte: 20} } # this will give greater than or equal}).limit(5)

Or

Db.wrangleddata.find(

{“lat”: {$lte: 20} } # this will give less than or equal}).limit(5)

Or just want only few columns

Db.wrangleddata.find({“lat”: {$lte: 20} }{“parcelid”:1}}).limit(5)

db.properties.find({"latitude":34280990} ,{"parcelid":1,latitude:1,longitude:1,_id:0, bedroomcnt:1}).limit(1)