# Flask: CNN model Deployment

Step 1: Import header for flask and call libraries, as well as, create instance of flask app in script
Step 2: Load ML model
Step 3: Create a method that makes the homepage using index.html
Step 4: Add GET request to capture input file and POST request to output predictions

**Steps 1-4 pic:**

```python
from json import dump
import numpy as np
from flask import Flask, request,render_template
import re
import random
import os
from tensorflow import keras
from Bio import SeqIO
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from werkzeug.utils import secure_filename


app = Flask(__name__) # step 1: create instance of app
model = keras.models.load_model('weights.best.hdf5') # step 2: loading my CNN model


@app.route('/')
def home():
    return render_template('index.html') # step 3: create a method that makes the homepage (uses style sheet as well)
                                         #adjusted input type to file and changed enctype to 'multipart/form-data'
                                         #to allow for 'GET'

@app.route('/predict',methods=['POST', 'GET']) # Step 4: Adding GET request to capture input file
                                               #POST request to output predictions
```

Step 5: Customize index.html

- Change the amount of inputs to 1
- Change input type = ‘text’ to input type = ‘file’
- Change enctype from default to enctype = ‘multipart/form-data’

**Step 5 pic:**

```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
  <h1>Predict Microbial Community</h1>

    <!-- Main Input For Receiving Query to our ML.....*****enctype = multipart/form-data is important for input type ='file'-->
    <form action="{{ url_for('predict')}}"method="post"enctype="multipart/form-data">
        <input type="file" name="DNA_fastq" placeholder="Fastq" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

    <br>
    <br>
    {{ prediction_text }}

 </div>
 <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company Logo"/>

</body>
</html>
```

Step 6: Create method for Prediction of CNN model

**Step 6 pic:**

```
def predict(): # Step 5: Creating Method for prediciton and rendering result on html gui

    #----------------------Code Below is for processing Input file and making it into a format that can be fed into CNN -----------------------------------
    nt=80
    features = []

    file = request.files[('DNA_fastq')] # reads the fastq file, has to be named DNA_fastq because that is what I named it in index.html
    file= file.read() # reads byte object (necessary to change bytes to string)
    file = file.decode('UTF-8') # decodes bytes that were utf encoded to strings

    with open("temp.txt", "w") as temp:
        for i in file:
            temp.write(i)
    with open('temp.txt', 'r') as f:

        for record in SeqIO.parse(f, "fastq"):
            if len(str(record.seq)) >= nt:
                features.append(str(record.seq))
        features = random.sample(features, 1000)
        features = [j.rstrip()[0:nt] for j in features]
        features =  [k for k in features if
                     k.count('A') > 0 and k.count('C') > 0 and k.count('G') > 0 and k.count('T') > 0
                     and len(list(k)) == nt]
        features=list(filter(lambda v: re.match('^[AGCT]+$', v), features))

        integer_encoder = LabelEncoder()
        one_hot_encoder = OneHotEncoder()
        input_test_features = []

        for sequence in features:
            integer_encoded = integer_encoder.fit_transform(list(sequence))
            integer_encoded = np.array(integer_encoded).reshape(-1, 1)
            one_hot_encoded = one_hot_encoder.fit_transform(integer_encoded)
            input_test_features.append(one_hot_encoded.toarray())

        np.set_printoptions(threshold = 40)
        input_test_features = np.stack(input_test_features)

        prediction = model.predict(input_test_features[10,:,:][np.newaxis,:,:])
        classes=np.argmax(prediction,axis=1)

    #output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Categories \n 0=oral\n1=gut\n2=skin\n3=vagina\nMicrobial Environment:  {}'.format(classes))
```

Step 7: Create secret key and debug app in main method:

**Step 7 pic:**

```
if __name__ == "__main__": # Step 8 Debug
    app.secret_key = 'super secret key' # idk why we need this, but it wont run without it for some reason (reasearch later?)
    app.run(debug=True)
```