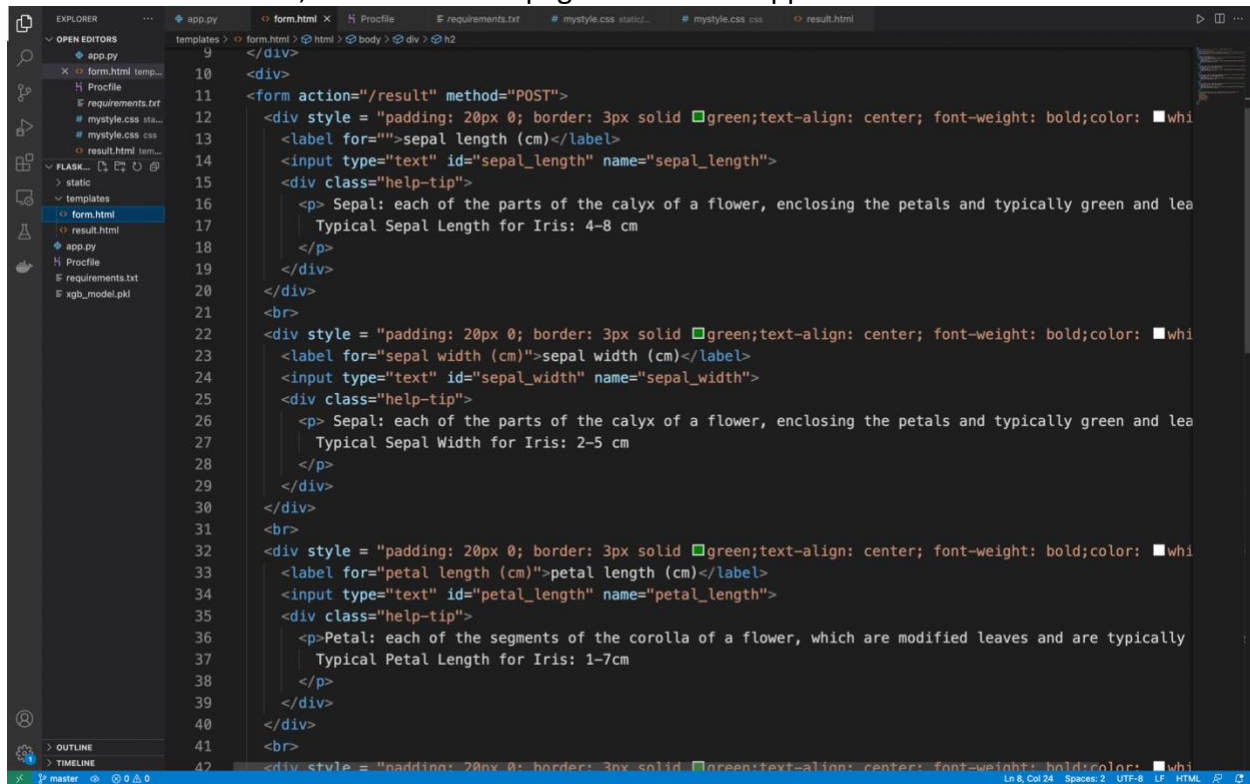
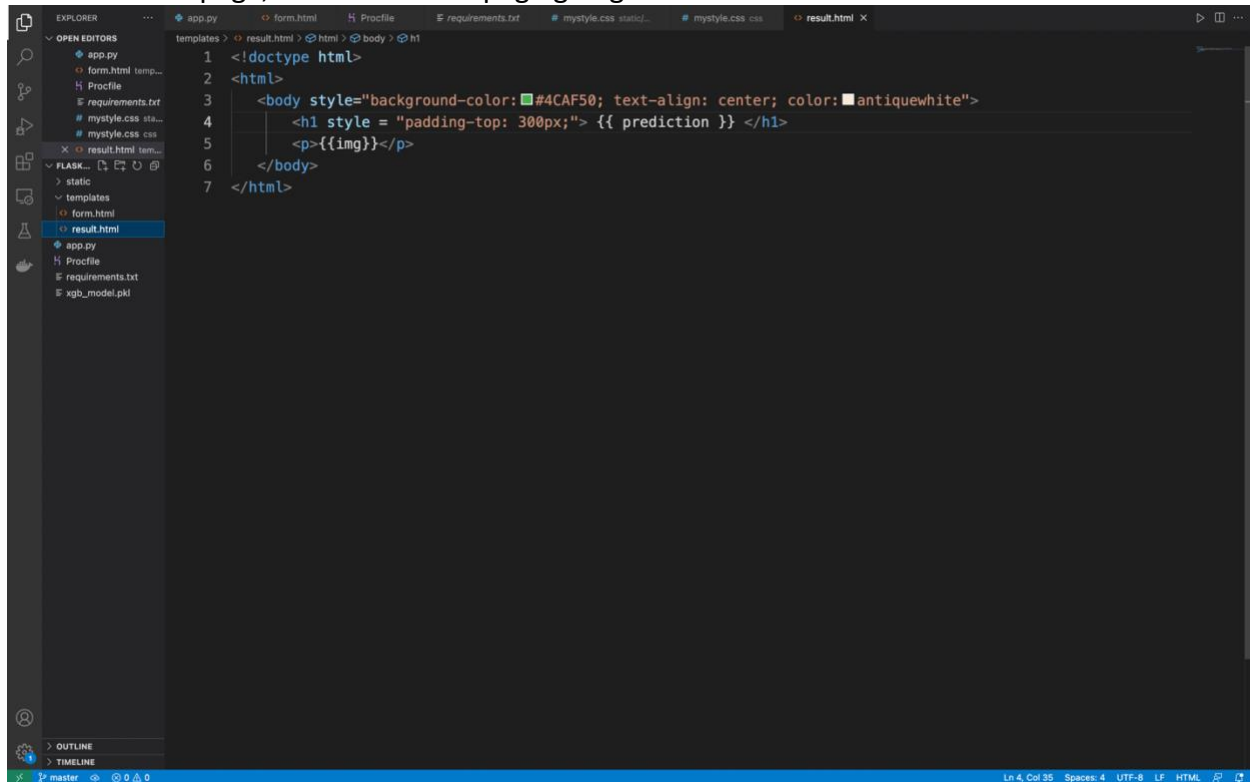


Created the form.html, this is the index page for our web application

A screenshot of the Visual Studio Code editor with the file 'form.html' open. The Explorer sidebar on the left shows a project structure with files like 'app.py', 'form.html', 'requirements.txt', 'mystyle.css', and 'result.html'. The main editor area displays the HTML code for 'form.html'. The code includes a form with three input fields: 'sepal length (cm)', 'sepal width (cm)', and 'petal length (cm)'. Each input field is followed by a 'help-tip' div containing descriptive text about the part of the flower. The form has an action of '/result' and a method of 'POST'. The code is styled with a dark theme and includes inline CSS for styling the form elements.

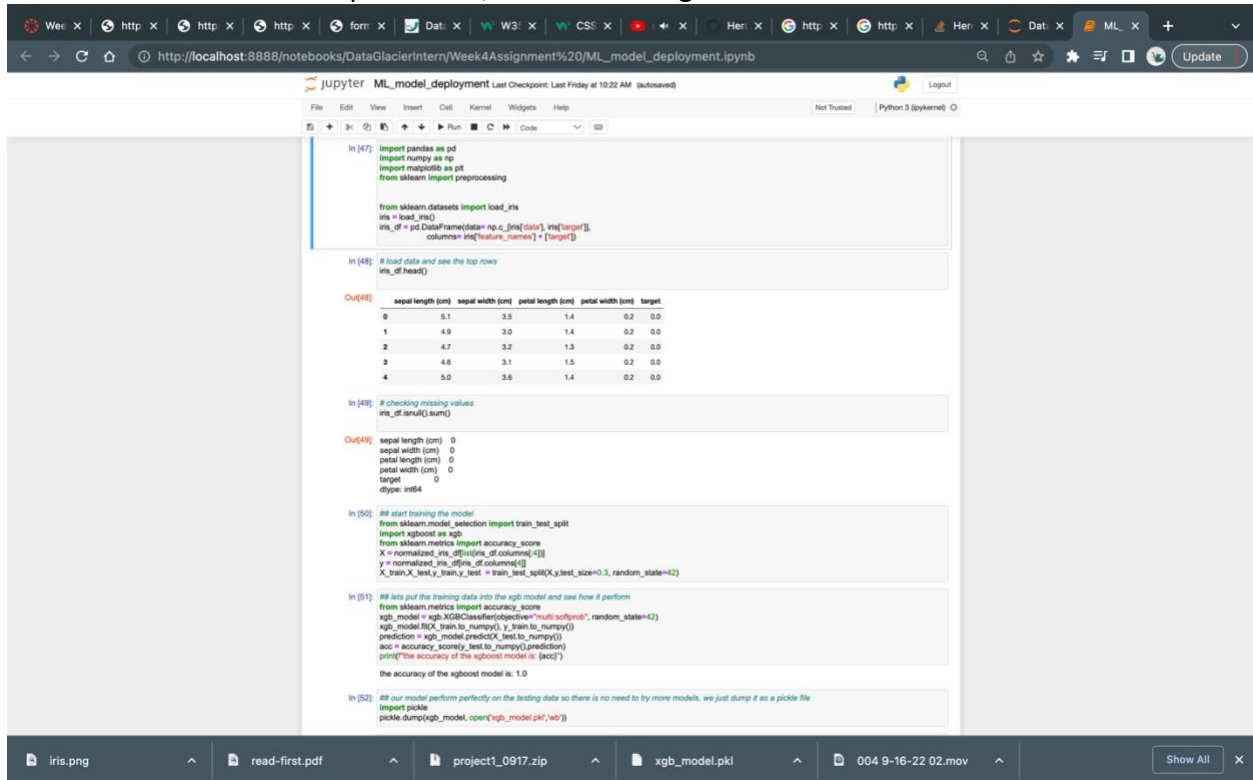
```
10 </div>
11 <form action="/result" method="POST">
12   <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white" >
13     <label for="">sepal length (cm)</label>
14     <input type="text" id="sepal_length" name="sepal_length">
15     <div class="help-tip">
16       <p> Sepal: each of the parts of the calyx of a flower, enclosing the petals and typically green and lea
17         Typical Sepal Length for Iris: 4-8 cm
18     </p>
19   </div>
20 </div>
21 <br>
22 <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white" >
23   <label for="sepal width (cm)">sepal width (cm)</label>
24   <input type="text" id="sepal_width" name="sepal_width">
25   <div class="help-tip">
26     <p> Sepal: each of the parts of the calyx of a flower, enclosing the petals and typically green and lea
27       Typical Sepal Width for Iris: 2-5 cm
28   </p>
29 </div>
30 </div>
31 <br>
32 <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white" >
33   <label for="petal length (cm)">petal length (cm)</label>
34   <input type="text" id="petal_length" name="petal_length">
35   <div class="help-tip">
36     <p>Petal: each of the segments of the corolla of a flower, which are modified leaves and are typically
37       Typical Petal Length for Iris: 1-7cm
38   </p>
39 </div>
40 </div>
41 <br>
42 <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white" >
```

Created result page, this is what the page going to take us after we click submit

A screenshot of the Visual Studio Code editor with the file 'result.html' open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the HTML code for 'result.html'. The code is a simple HTML document with a doctype, html, and body tags. The body contains a h1 tag with a style attribute for padding-top and a p tag with a placeholder for an image. The code is styled with a dark theme and includes inline CSS for styling the page elements.

```
1 <!doctype html>
2 <html>
3   <body style="background-color: #4CAF50; text-align: center; color: antiquewhite">
4     <h1 style = "padding-top: 300px;"> {{ prediction }} </h1>
5     <p>{{img}}</p>
6   </body>
7 </html>
```

Created the model for Iris prediction, the model is xgboost.



The screenshot shows a Jupyter Notebook with the following code and output:

```
In [47]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing

from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
                      columns=iris['feature_names'] + ['target'])

In [48]: # load data and see the top rows
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

```
In [49]: # checking missing values
iris_df.isnull().sum()

Out[49]: sepal length (cm)  0
sepal width (cm)  0
petal length (cm)  0
petal width (cm)  0
target  0
dtype: int64

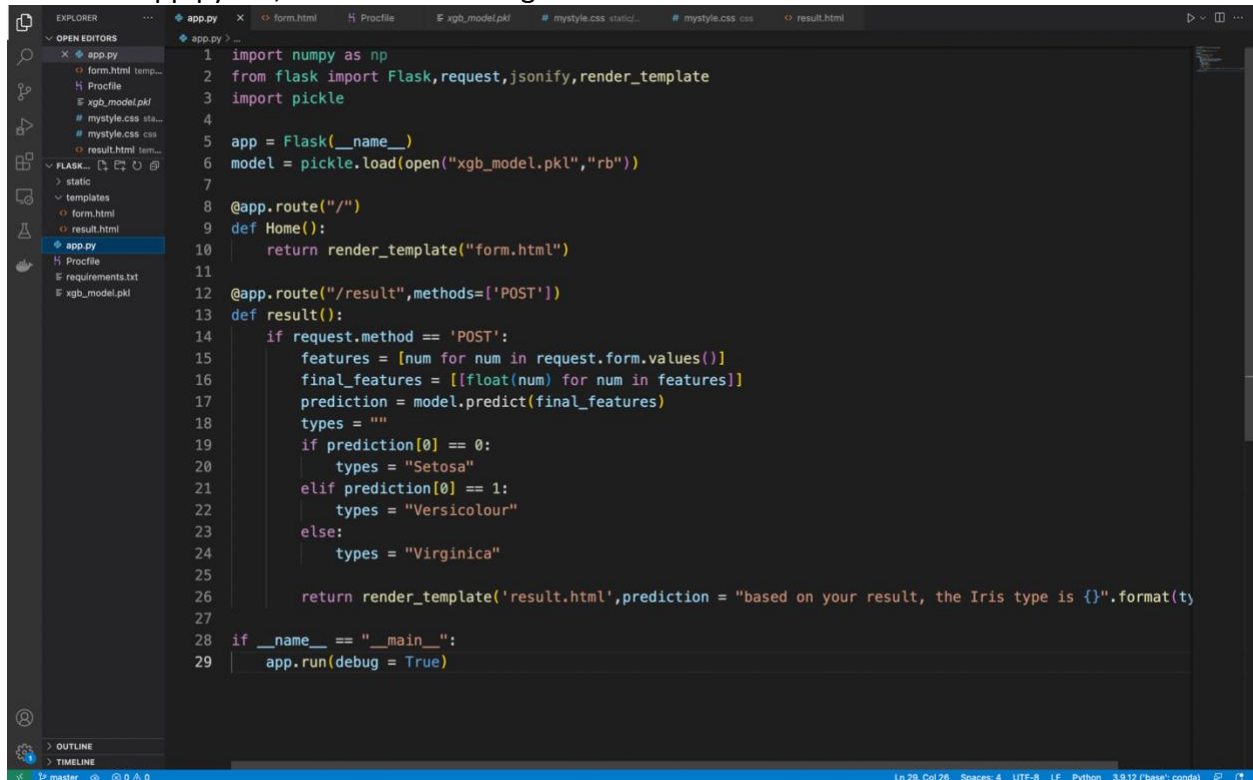
In [50]: # start training the model
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import accuracy_score
X = normalized_iris_df[iris_df.columns[0:4]]
y = normalized_iris_df[iris_df.columns[5]]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [51]: # lets put the training data into the xgb model and see how it perform
from sklearn.metrics import accuracy_score
xgb_model = xgb.XGBClassifier(objective='multi:softmax', random_state=42)
xgb_model.fit(X_train.to_numpy(), y_train.to_numpy())
prediction = xgb_model.predict(X_test.to_numpy())
acc = accuracy_score(y_test.to_numpy(), prediction)
print("the accuracy of the xgboost model is: {}".format(acc))

the accuracy of the xgboost model is: 1.0

In [52]: # our model perform perfectly on the testing data so there is no need to try more models, we just dump it as a pickle file
import pickle
pickle.dump(xgb_model, open("xgb_model.pkl", "wb"))
```

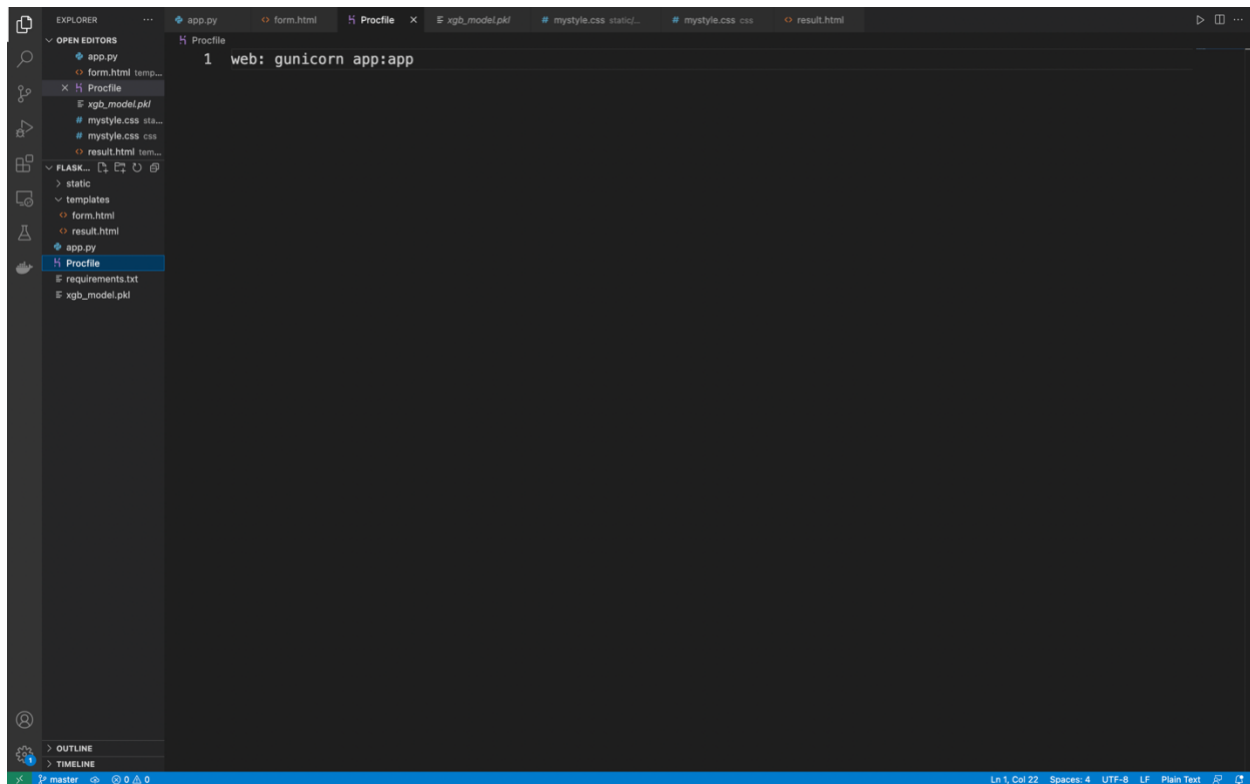
Write the app.py file, it is a written using flask



The screenshot shows the following code in the app.py file:

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open("xgb_model.pkl", "rb"))
7
8 @app.route("/")
9 def Home():
10     return render_template("form.html")
11
12 @app.route("/result", methods=['POST'])
13 def result():
14     if request.method == 'POST':
15         features = [num for num in request.form.values()]
16         final_features = [[float(num) for num in features]]
17         prediction = model.predict(final_features)
18         types = ""
19         if prediction[0] == 0:
20             types = "Setosa"
21         elif prediction[0] == 1:
22             types = "Versicolour"
23         else:
24             types = "Virginica"
25
26         return render_template('result.html', prediction = "based on your result, the Iris type is {}".format(types))
27
28 if __name__ == "__main__":
29     app.run(debug = True)
```

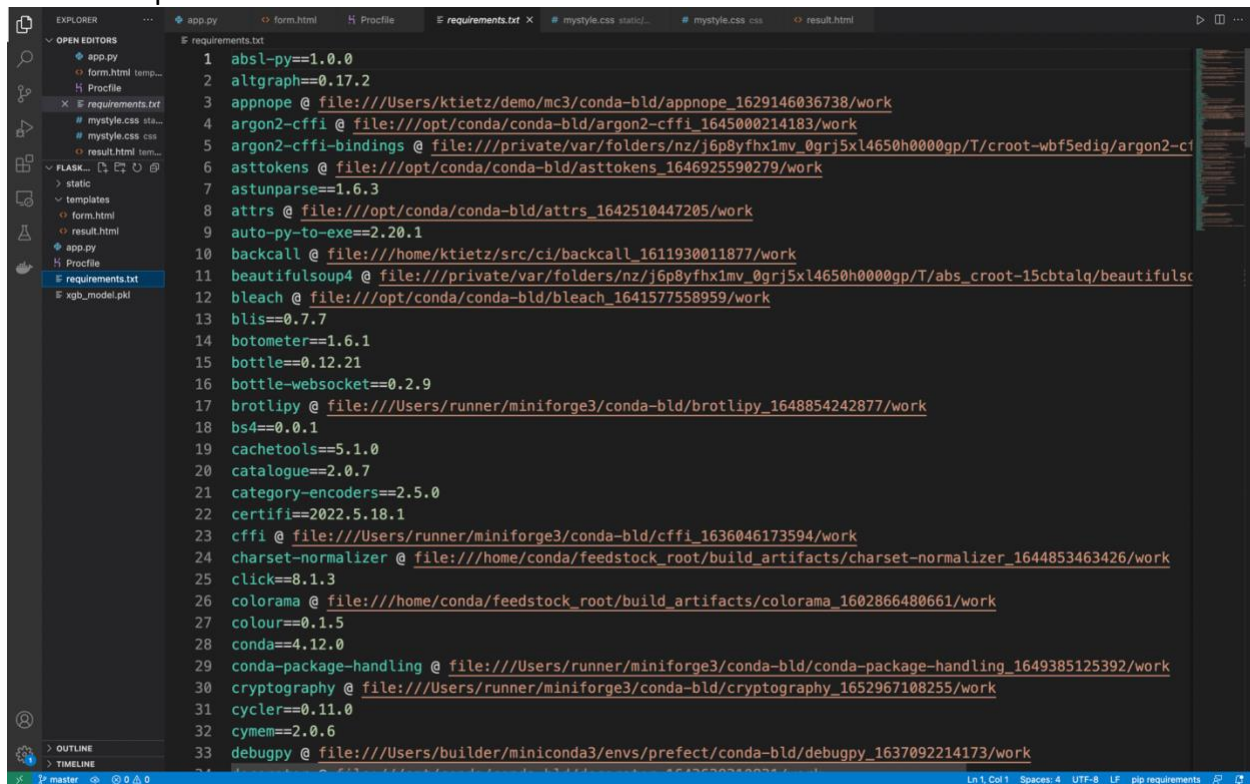
Created Procfile for heroku



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with files like `app.py`, `form.html`, `Procfile`, `xgb_model.pkl`, `mystyle.css`, `static`, `templates`, `form.html`, `result.html`, and `app.py`. The code editor displays the content of the `Procfile` file, which contains a single line: `1 web: gunicorn app:app`. The status bar at the bottom indicates the file is at line 1, column 22, with 4 spaces, in UTF-8 encoding, using LF line endings, and is a plain text file.

```
1 web: gunicorn app:app
```

Created requirements.txt






The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with files like `app.py`, `form.html`, `Procfile`, `xgb_model.pkl`, `mystyle.css`, `static`, `templates`, `form.html`, `result.html`, and `app.py`. The code editor displays the content of the `requirements.txt` file, which lists various Python packages and their versions, along with their source URLs. The status bar at the bottom indicates the file is at line 1, column 1, with 4 spaces, in UTF-8 encoding, using LF line endings, and is a pip requirements file.

```
1 absl-py==1.0.0
2 altgraph==0.17.2
3 appnope @ file:///Users/ktietz/demo/mc3/conda-bld/appnope_1629146036738/work
4 argon2-cffi @ file:///opt/conda/conda-bld/argon2-cffi_164500214183/work
5 argon2-cffi-bindings @ file:///private/var/folders/nz/j6p8yfhx1mv_0grj5x14650h0000gp/T/croot-wbf5edig/argon2-cffi_164500214183/work
6 asttokens @ file:///opt/conda/conda-bld/asttokens_1646925590279/work
7 astunparse==1.6.3
8 attrs @ file:///opt/conda/conda-bld/attrs_1642510447205/work
9 auto-py-to-exe==2.20.1
10 backcall @ file:///home/ktietz/src/ci/backcall_1611930011877/work
11 beautifulsoup4 @ file:///private/var/folders/nz/j6p8yfhx1mv_0grj5x14650h0000gp/T/abs_croot-15cbtalq/beautifulsoup4_164500214183/work
12 bleach @ file:///opt/conda/conda-bld/bleach_1641577558959/work
13 blis==0.7.7
14 botometer==1.6.1
15 bottle==0.12.21
16 bottle-websocket==0.2.9
17 brotli @ file:///Users/runner/miniforge3/conda-bld/brotli_1648854242877/work
18 bs4==0.0.1
19 cachetools==5.1.0
20 catalogue==2.0.7
21 category-encoders==2.5.0
22 certifi==2022.5.18.1
23 cffi @ file:///Users/runner/miniforge3/conda-bld/cffi_1636046173594/work
24 charset-normalizer @ file:///home/conda/feedstock_root/build_artifacts/charset-normalizer_1644853463426/work
25 click==8.1.3
26 colorama @ file:///home/conda/feedstock_root/build_artifacts/colorama_1602866480661/work
27 colour==0.1.5
28 conda==4.12.0
29 conda-package-handling @ file:///Users/runner/miniforge3/conda-bld/conda-package-handling_1649385125392/work
30 cryptography @ file:///Users/runner/miniforge3/conda-bld/cryptography_1652967108255/work
31 cycycler==0.11.0
32 cymem==2.0.6
33 debugpy @ file:///Users/builder/miniconda3/envs/prefect/conda-bld/debugpy_1637092214173/work
```

Make sure our webapp run perfectly

Week 5: x Data Sc: x W3Sch: x CSS for: x (1) 1 x Heroku: x https:// x https:// x Heroku: x DataG: x iris-pre: x https:// x +

← → ↻ 🏠 https://iris-predictor-01.herokuapp.com ⚙️ ☆ ⚙️ 📄 🔄 Update



Iris Versicolor Iris Setosa Iris Virginica

Iris Classifier

sepal length (cm) ?

sepal width (cm) ?

petal length (cm) ?

petal width (cm) ?

Week 5: x Data Sc: x W3Sch: x CSS for: x (1) 1 x Heroku: x https:// x https:// x Heroku: x DataG: x iris-pre: x https:// x +

← → ↻ 🏠 https://iris-predictor-01.herokuapp.com/result ⚙️ ☆ ⚙️ 📄 🔄 Update

based on your result, the Iris type is Virginica