Data (Loan status data – same as week 4)

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0

Model (same as week 4)

```
df_new['Gender_dummy'] = 0
df_new['Married_dummy'] = 0
df_new['Education_dummy'] = 0
df_new['Self_Employed_dummy'] = 0
df_new['Property_Area_dummy'] = 0
df_new['Dependents_dummy'] = 0

df_new['Gender_dummy'] = df_new['Gender'].apply(lambda x: 1.0 if x == 'Male' else 0.0)

df_new['Married_dummy'] = df_new['Married'].apply(lambda x: 1.0 if x == 'Yes' else 0.0)

df_new['Education_dummy'] = df_new['Education'].apply(lambda x: 1.0 if x == 'Graduate' else 0.0)
df_new['Self_Employed_dummy'] = df_new['Self_Employed'].apply(lambda x: 1.0 if x == 'Yes' else 0.0)

property_area_mapping = {
    'Urban': 1.0,
    'Rural': 0.0,
    'Semiurban': 0.5
}
df_new['Property_Area_dummy'] = df_new['Property_Area'].apply(lambda x: property_area_mapping.get(x, 0.0))

Dependents_mapping = {
    '0': 0.0,
    '1': 1.0,
    '2': 2.0,
    '3+': 3.0
}



df_new['Dependents_dummy'] = df_new['Dependents'].apply(lambda x: Dependents_mapping.get(x, 0.0))
```

```
from sklearn.linear_model import LogisticRegression
import pickle

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(x_train, y_train)

y_pred = logreg.predict(x_test)
# Output the model
pickle.dump(model, open('logistic_model.pkl', 'wb'))
```

Web Application and interface

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 22 14:42:26 2023

@author: terry
"""

import numpy as np
from flask import Flask, request, render_template
import pickle

#Create the application
app = Flask(__name__)

#Loading the model
model = pickle.load(open('logistic_model.pkl','rb'))

#Display the html interface(codes in another page)
@app.route('/')
def home():
    return render_template('index.html')
```

```
dropdown_mappings = {
    "Credit_History": {
        "1.0": 1,
        "0.0": 0,

    },

    "Property_Area": {
        "1.0": 1,
        "0.5": 0.5,
        "0.0": 0
    },

    "Married": {
        "1.0": 1,
        "0.0": 0,
    },

    "Gender": {
        "1.0": 1,
        "0.0": 0,
    },

    "Dependents": {
        "0.0": 0,
        "1.0": 1,
        "2.0": 2,
        "3.0": 3
    },

    "Education": {
        "0.0": 0,
        "1.0": 1,

    },
```

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

    form_values = request.form.to_dict()

    for dropdown_name, dropdown_mapping in dropdown_mappings.items():
        selected_value = form_values.get(dropdown_name, "")
        form_values[dropdown_name] = dropdown_mapping.get(selected_value, 0)



    #int_features = [int(x) for x in request.form.values()]
    int_features = [int(x) for x in form_values.values()]

    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

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

    return render_template('index.html', prediction_text = 'Loan Status is: {}'.format(output))


if __name__ == "__main__":
    app.run(port=5000, debug = True)
```

```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') }}">
```

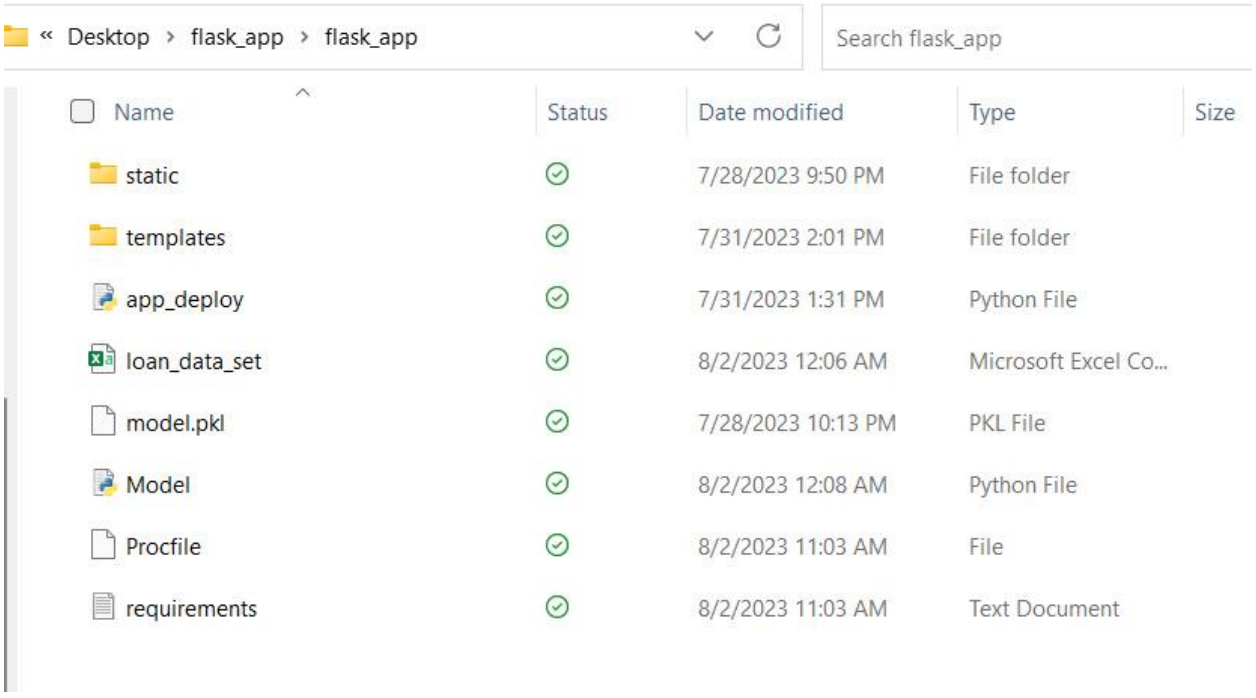
```
</head>

<body>
 <div class="login">
    <h1>Predict House Price</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">

    <!-- Dropdown list for gender -->
      <select name="Gender" required="required">
        <option value="">Select Gender</option>
        <option value="1.0">Male</option>
        <option value="0.0">Female</option>
      </select>
    <!-- Dropdown list for Married -->
      <select name="Married" required="required">
         <option value="">Marritial Status</option>
         <option value="1.0">Yes</option>
         <option value="0.0">No</option>

      </select>

      <select name="Dependents" required="required">
        <option value="">Select Number of Dependents</option>
        <option value="0.0">0</option>
        <option value="1.0">1</option>
        <option value="2.0">2</option>
        <option value="3.0">3+</option>

      </select>

        <!-- Dropdown list for Education -->
      <select name="Education" required="required">
        <option value="">Select Education</option>
        <option value="1.0">Graduate</option>
        <option value="0.0">Not Graduate</option>

      </select>
```

```
    <select name="Education" required="required">
      <option value="">Select Education</option>
      <option value="1.0">Graduate</option>
      <option value="0.0">Not Graduate</option>

    </select>


    <select name="Self_Employed" required="required">
      <option value="">self employed Status</option>
      <option value="1.0">Yes</option>
      <option value="0.0">No</option>

    </select>



      <input type="number" name="ApplicantIncome" placeholder="Applicant Income" required="required" />
      <input type="number" name="CoapplicantIncome" placeholder="Coapplicant Income" required="required" />
      <input type="number" name="LoanAmount" placeholder="Loan Amount" required="required" />
      <input type="number" name="Loan_Amount_Term" placeholder="Loan Amount Term" required="required" />

  <!-- Dropdown list for Credit_History -->
   <select name="Credit_History" required="required">
      <option value="">Select Credit History</option>
      <option value="1.0">1</option>
      <option value="0.0">0</option>

      <!-- Add more options as needed -->
    </select>

  <!-- Dropdown list for Property_Area -->
   <select name="Property_Area" required="required">
      <option value="">Select Property Area</option>
      <option value="1.0">Rural</option>
      <option value="0.5">Semiurban</option>
      <option value="0.0">Urban</option>
      <!-- Add more options as needed -->
    </select>
```

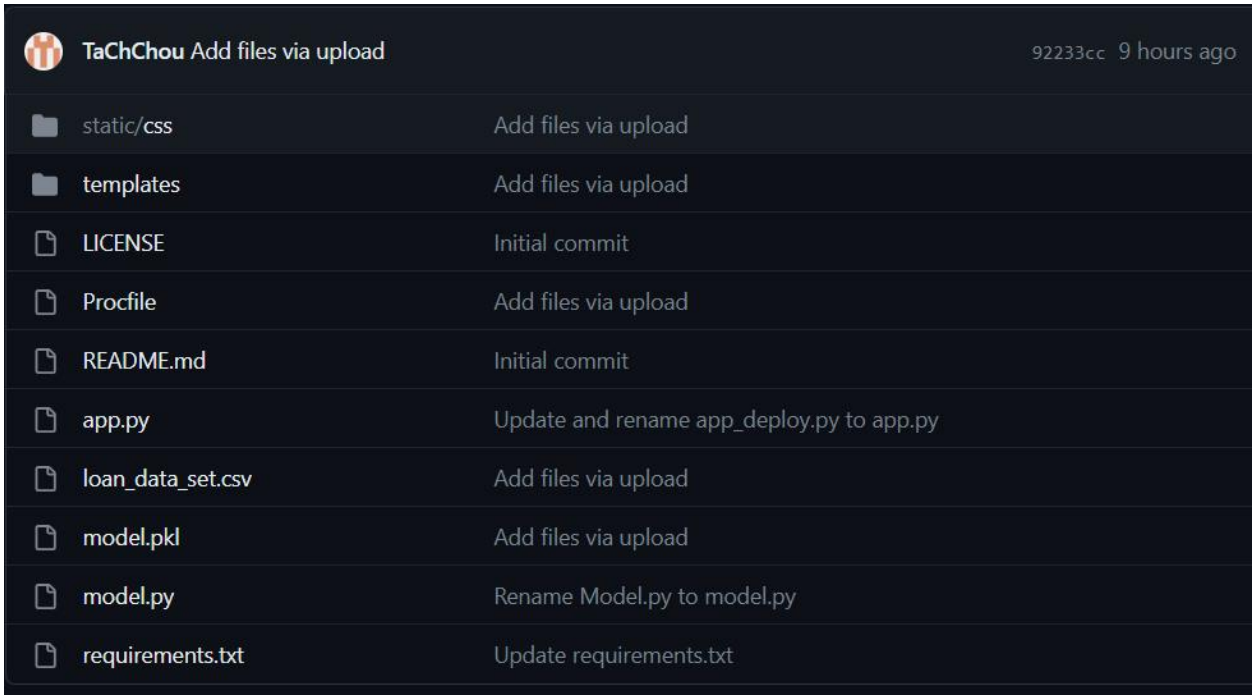
Creating requirements.txt

```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\deadl\OneDrive\文件\Heroku>pip freeze
blinker==1.6.2
click==8.1.6
colorama==0.4.6
Flask==2.3.2
itsdangerous==2.1.2
Jinja2==3.1.2
joblib==1.3.1
MarkupSafe==2.1.3
numpy==1.25.1
scikit-learn==1.3.0
scipy==1.11.1
sklearn==0.0.post7
threadpoolctl==3.2.0
Werkzeug==2.3.6

C:\Users\deadl\OneDrive\文件\Heroku>pip freeze > requirements.txt
```

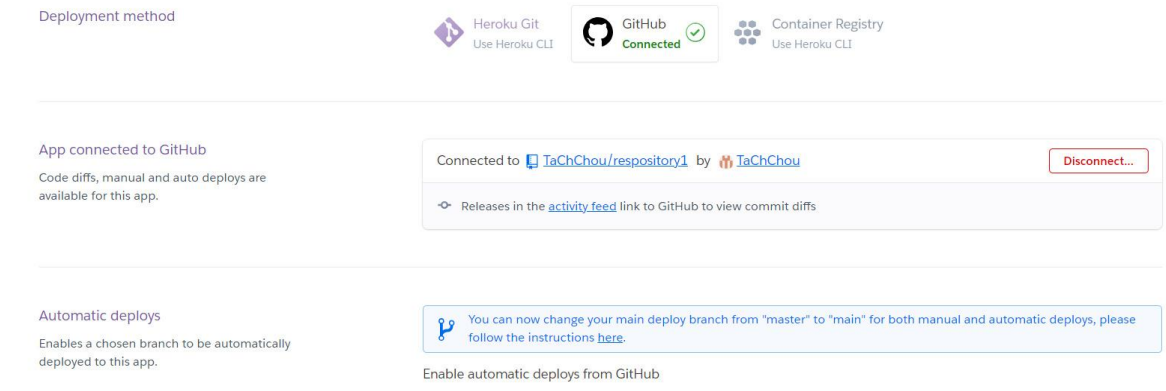
File for upload



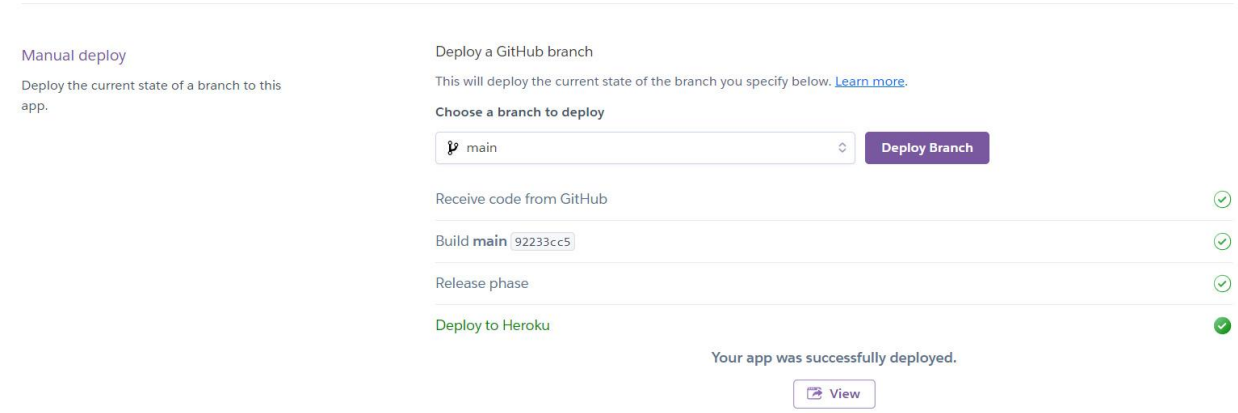
Upload to Github repository



Heroku Account creation and setting up connection with Github



Deploy Github branch



Website View