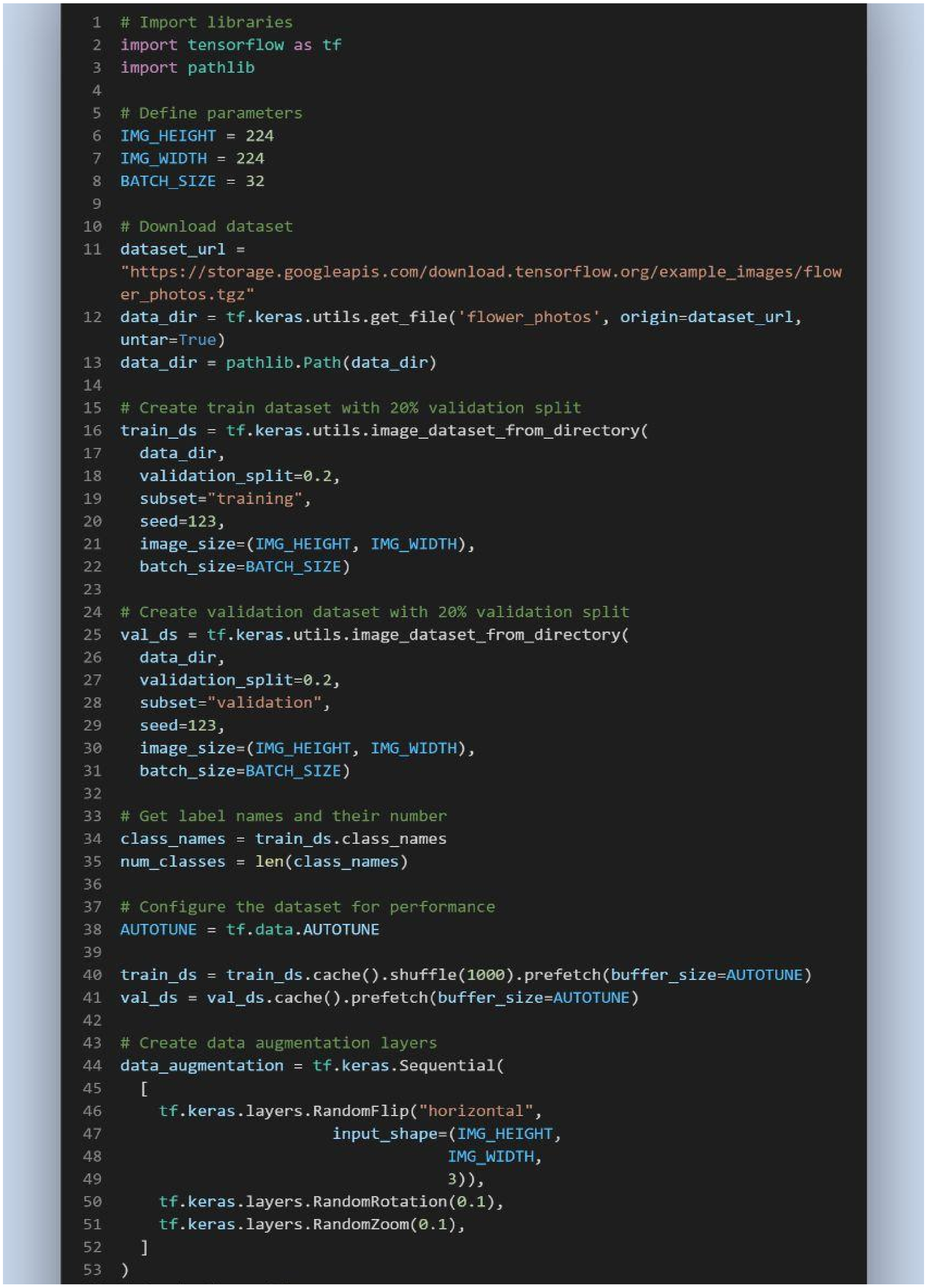
Model Preparation



```python
# Import libraries
import tensorflow as tf
import pathlib

# Define parameters
IMG_HEIGHT = 224
IMG_WIDTH = 224
BATCH_SIZE = 32

# Download dataset
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url, untar=True)
data_dir = pathlib.Path(data_dir)

# Create train dataset with 20% validation split
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(IMG_HEIGHT, IMG_WIDTH),
  batch_size=BATCH_SIZE)

# Create validation dataset with 20% validation split
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(IMG_HEIGHT, IMG_WIDTH),
  batch_size=BATCH_SIZE)

# Get label names and their number
class_names = train_ds.class_names
num_classes = len(class_names)

# Configure the dataset for performance
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

# Create data augmentation layers
data_augmentation = tf.keras.Sequential(
  [
    tf.keras.layers.RandomFlip("horizontal",
                      input_shape=(IMG_HEIGHT,
                                  IMG_WIDTH,
                                  3)),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
  ]
)
```

```python
# Create the model
model = tf.keras.Sequential([
  data_augmentation,
  tf.keras.layers.Rescaling(1./255),
  tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes)
])

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
# Train the model
epochs = 15
history = model.fit(  train_ds,
                      validation_data=val_ds,
                      epochs=epochs
)
# Save the model
model.save('model/model.h5')
```

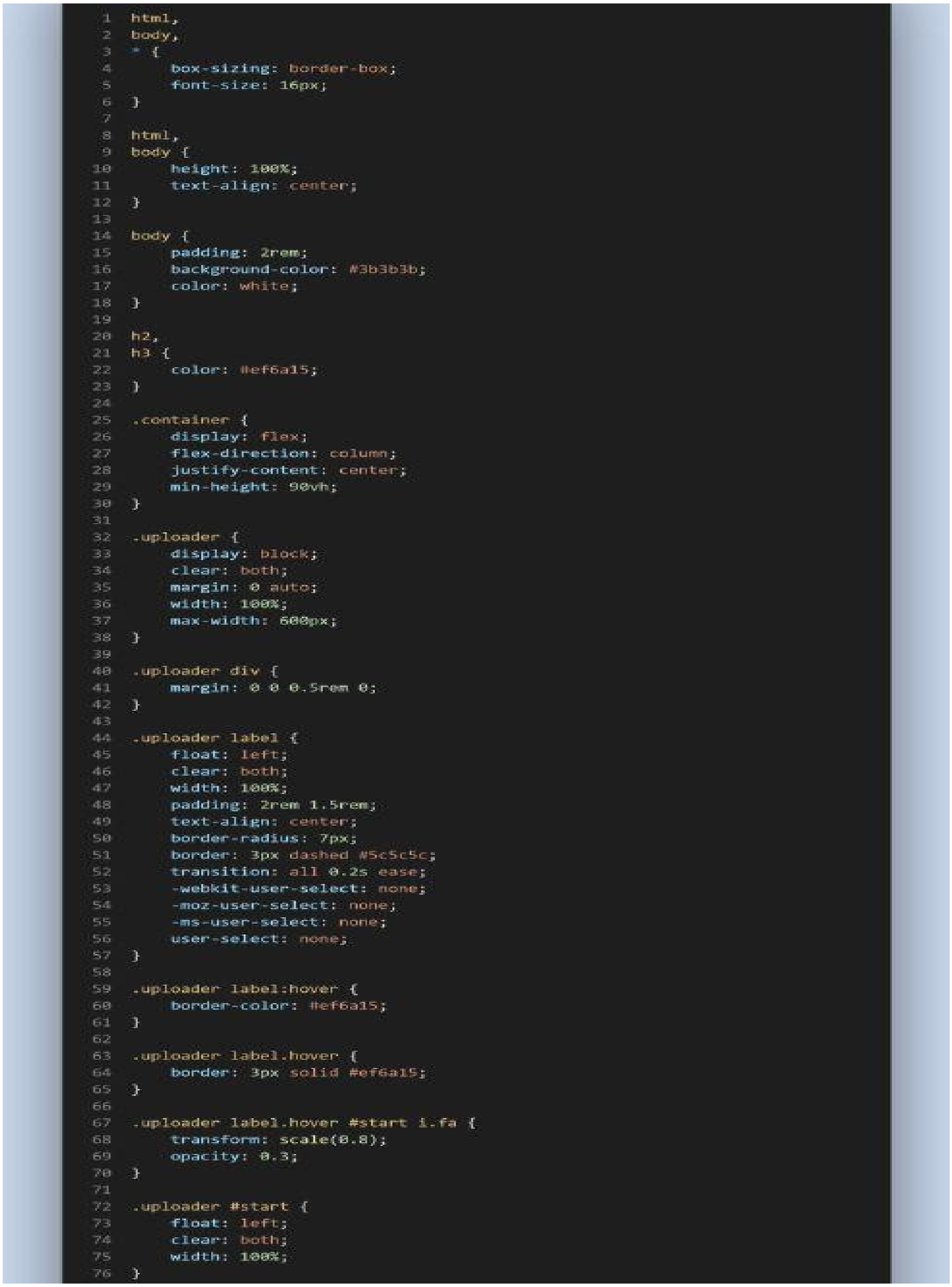
HTML

```html
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Flask Deployment</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
</head>
```

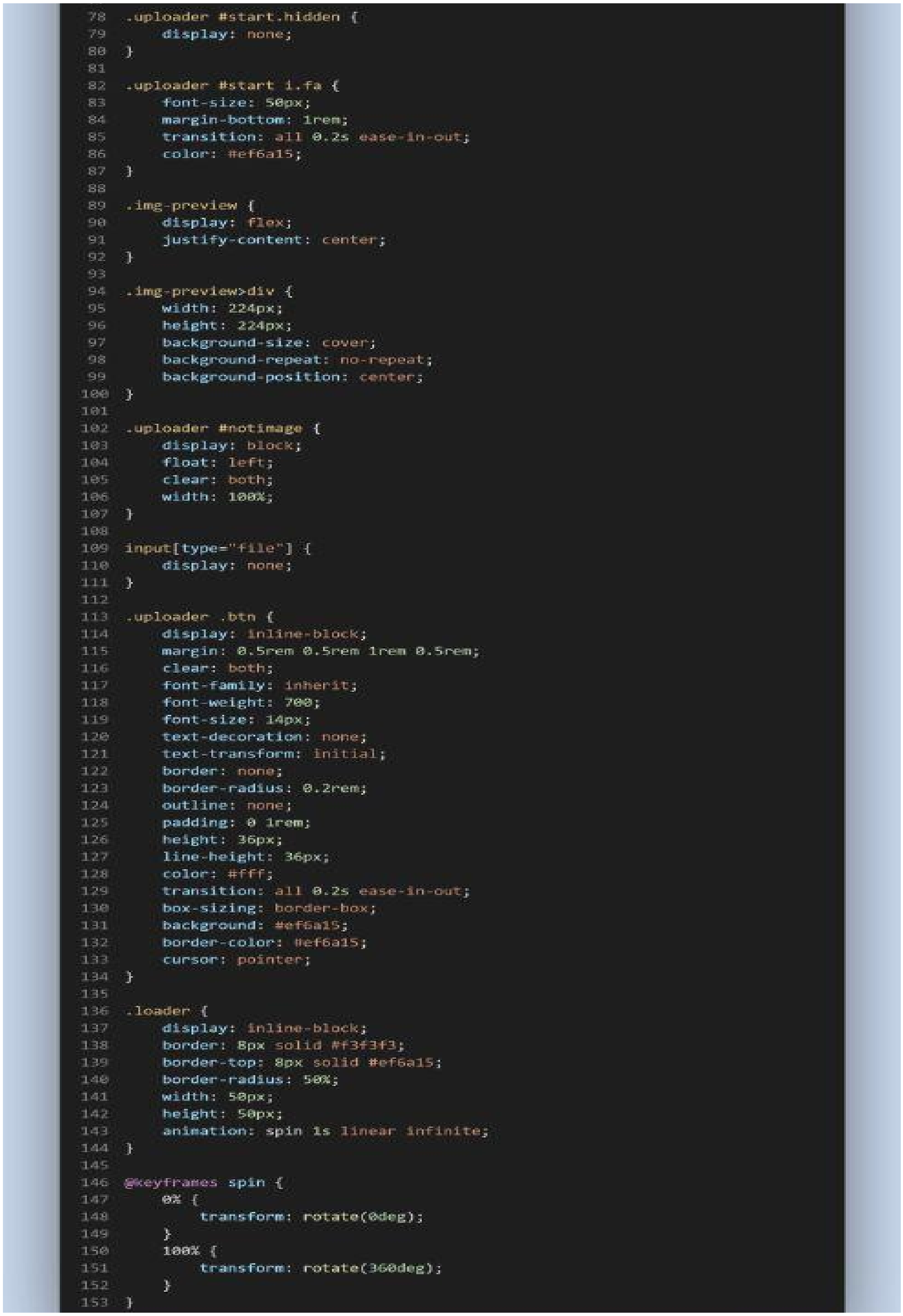
```html

<body>
    <div class="container">
        <div id="content">{% block content %}{% endblock %}</div>
    </div>
</body>

<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
</footer>

</html>
```

```html
{% extends "base.html" %} {% block content %}

<h2>Flower Image Classifier</h2>
<div>
    <form id="upload-file" method="post" enctype="multipart/form-data" class=uploader>
        <input type="file" name="file" id="imageUpload" accept="image/*">
        <label for="imageUpload" id="file-drag">
            <div class="image-section" style="display:none;">
                <div class="img-preview">
                    <div id="imagePreview">
                    </div>
                </div>
            </div>
            <div id="start">
                <i class="fa fa-cloud-upload"></i>
                <div>
                    <h5>Select a file or drag here</h5>
                </div>
                <span id="file-upload-btn" class="btn btn-primary">Select a file</span>
            </div>
        </label>
        <div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict" style="display:none;">Predict</button>
            <div class="loader" style="display:none;"></div>
            <h3 id="result">
                <span> </span>
            </h3>
        </div>
    </form>
</div>

{% endblock %}
```

CSS

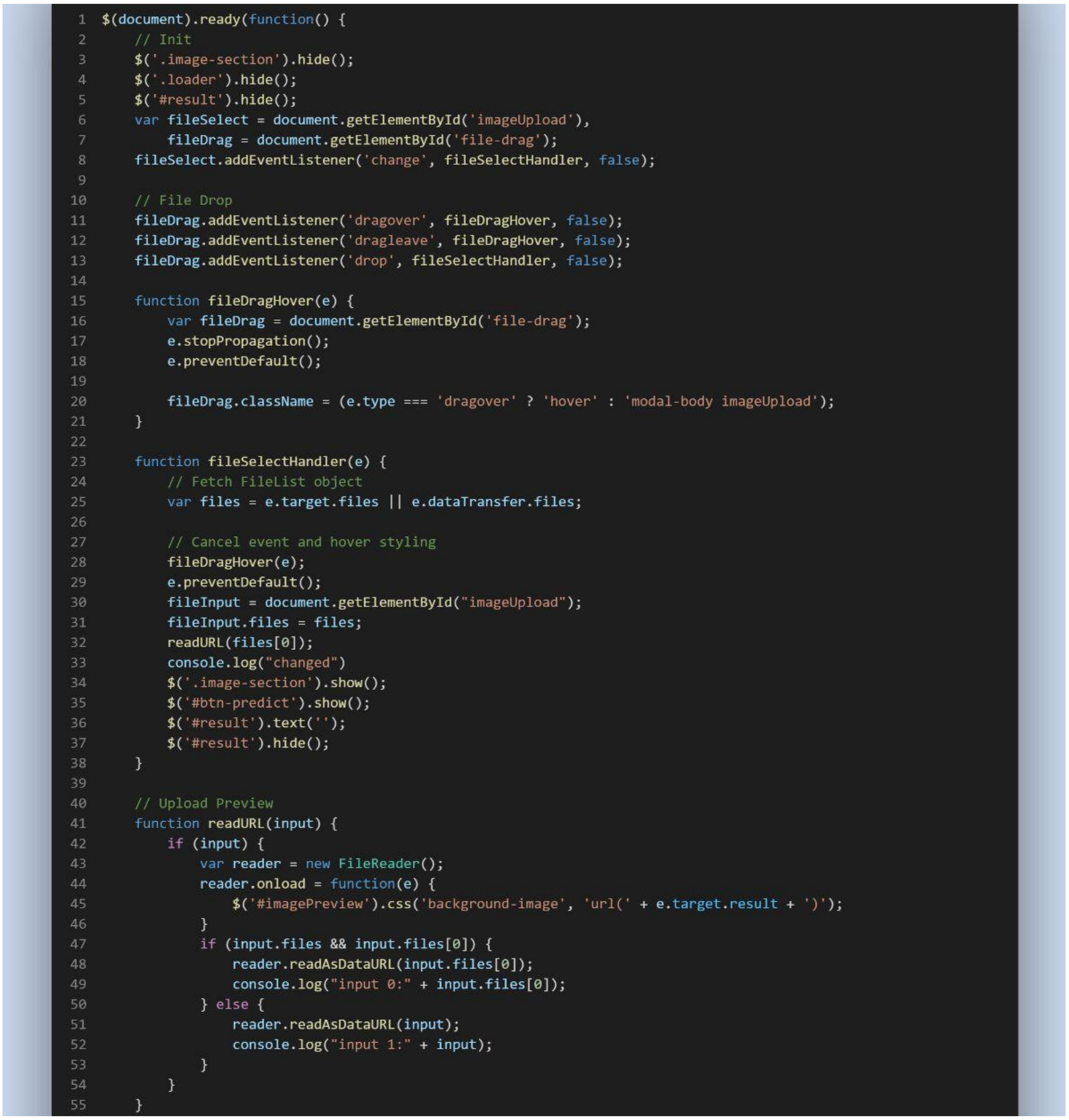


```css
html,
body,
* {
    box-sizing: border-box;
    font-size: 16px;
}

html,
body {
    height: 100%;
    text-align: center;
}

body {
    padding: 2rem;
    background-color: #3b3b3b;
    color: white;
}

h2,
h3 {
    color: #ef6a15;
}

.container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    min-height: 90vh;
}

.uploader {
    display: block;
    clear: both;
    margin: 0 auto;
    width: 100%;
    max-width: 600px;
}

.uploader div {
    margin: 0 0 0.5rem 0;
}

.uploader label {
    float: left;
    clear: both;
    width: 100%;
    padding: 2rem 1.5rem;
    text-align: center;
    border-radius: 7px;
    border: 3px dashed #5c5c5c;
    transition: all 0.2s ease;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

.uploader label:hover {
    border-color: #ef6a15;
}

.uploader label.hover {
    border: 3px solid #ef6a15;
}

.uploader label.hover #start i.fa {
    transform: scale(0.8);
    opacity: 0.3;
}

.uploader #start {
    float: left;
    clear: both;
    width: 100%;
}
```

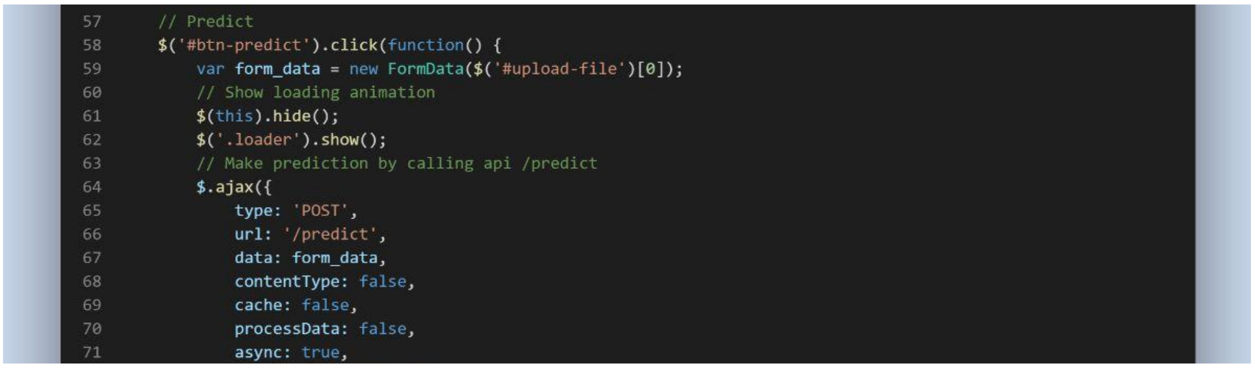


```css
.uploader #start.hidden {
    display: none;
}

.uploader #start i.fa {
    font-size: 50px;
    margin-bottom: 1rem;
    transition: all 0.2s ease-in-out;
    color: #ef6a15;
}

.img-preview {
    display: flex;
    justify-content: center;
}

.img-preview>div {
    width: 224px;
    height: 224px;
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center;
}

.uploader #notimage {
    display: block;
    float: left;
    clear: both;
    width: 100%;
}

input[type="file"] {
    display: none;
}

.uploader .btn {
    display: inline-block;
    margin: 0.5rem 0.5rem 1rem 0.5rem;
    clear: both;
    font-family: inherit;
    font-weight: 700;
    font-size: 14px;
    text-decoration: none;
    text-transform: initial;
    border: none;
    border-radius: 0.2rem;
    outline: none;
    padding: 0 1rem;
    height: 36px;
    line-height: 36px;
    color: #fff;
    transition: all 0.2s ease-in-out;
    box-sizing: border-box;
    background: #ef6a15;
    border-color: #ef6a15;
    cursor: pointer;
}

.loader {
    display: inline-block;
    border: 8px solid #f3f3f3;
    border-top: 8px solid #ef6a15;
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}
```

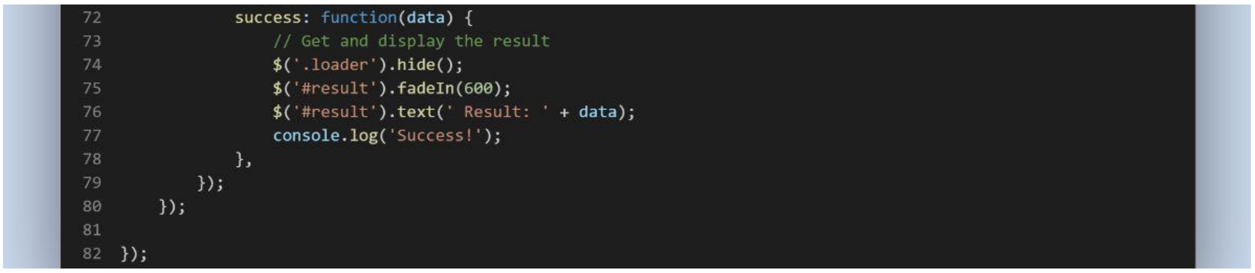
JavaScript



```javascript
$(document).ready(function() {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();
    var fileSelect = document.getElementById('imageUpload'),
        fileDrag = document.getElementById('file-drag');
    fileSelect.addEventListener('change', fileSelectHandler, false);

    // File Drop
    fileDrag.addEventListener('dragover', fileDragHover, false);
    fileDrag.addEventListener('dragleave', fileDragHover, false);
    fileDrag.addEventListener('drop', fileSelectHandler, false);

    function fileDragHover(e) {
        var fileDrag = document.getElementById('file-drag');
        e.stopPropagation();
        e.preventDefault();

        fileDrag.className = (e.type === 'dragover' ? 'hover' : 'modal-body imageUpload');
    }

    function fileSelectHandler(e) {
        // Fetch FileList object
        var files = e.target.files || e.dataTransfer.files;

        // Cancel event and hover styling
        fileDragHover(e);
        e.preventDefault();
        fileInput = document.getElementById("imageUpload");
        fileInput.files = files;
        readURL(files[0]);
        console.log("changed")
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
    }

    // Upload Preview
    function readURL(input) {
        if (input) {
            var reader = new FileReader();
            reader.onload = function(e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
            }
            if (input.files && input.files[0]) {
                reader.readAsDataURL(input.files[0]);
                console.log("input 0:" + input.files[0]);
            } else {
                reader.readAsDataURL(input);
                console.log("input 1:" + input);
            }
        }
    }
```

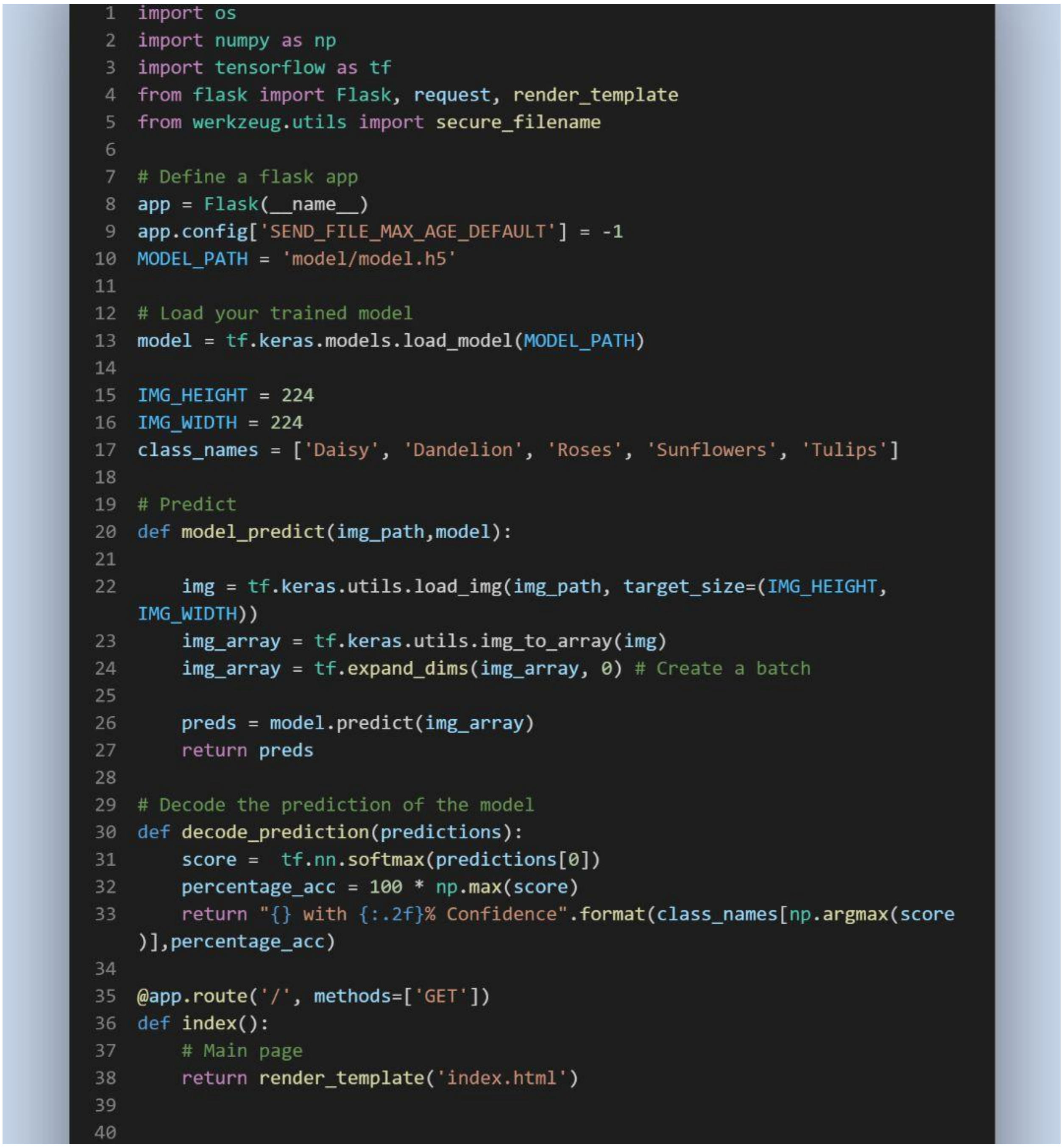


```javascript
    // Predict
    $('#btn-predict').click(function() {
        var form_data = new FormData($('#upload-file')[0]);
        // Show loading animation
        $(this).hide();
        $('.loader').show();
        // Make prediction by calling api /predict
        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
```

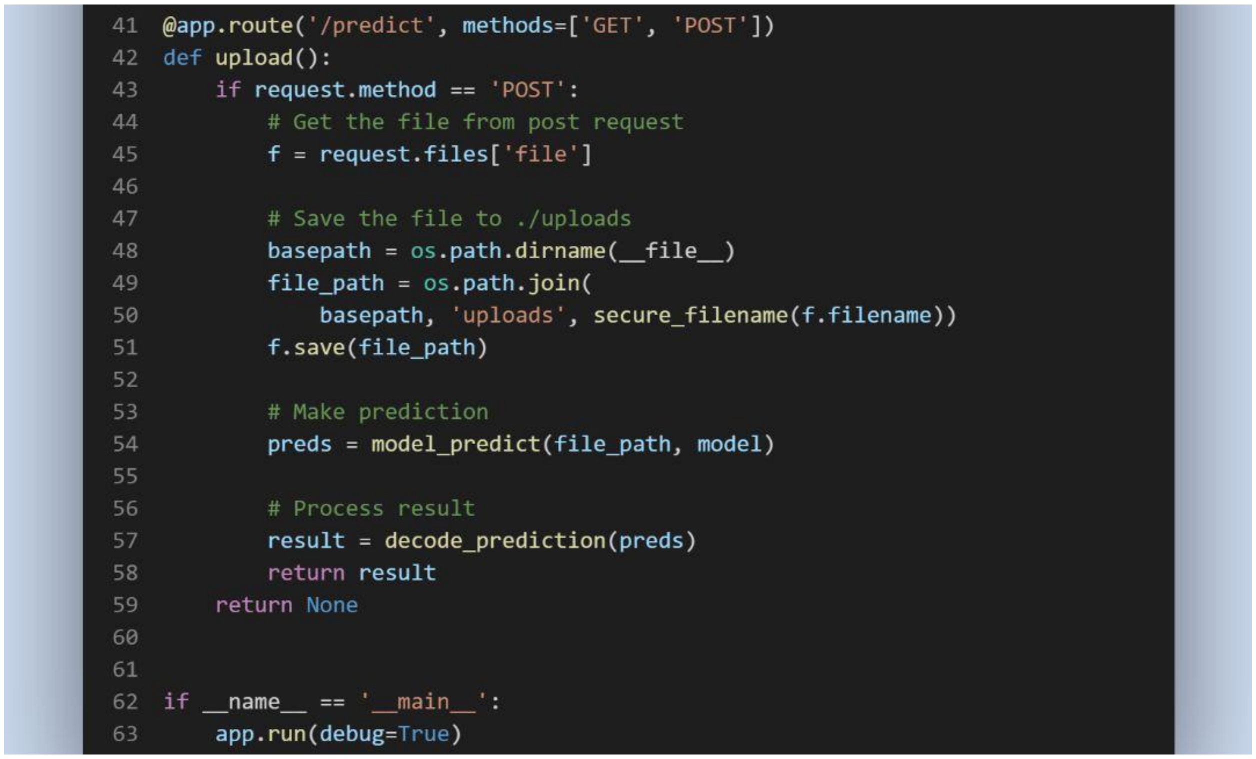


```
            success: function(data) {
                // Get and display the result
                $('.loader').hide();
                $('#result').fadeIn(600);
                $('#result').text(' Result:  ' + data);
                console.log('Success!');
            },
        });
    });

});
```

Main App

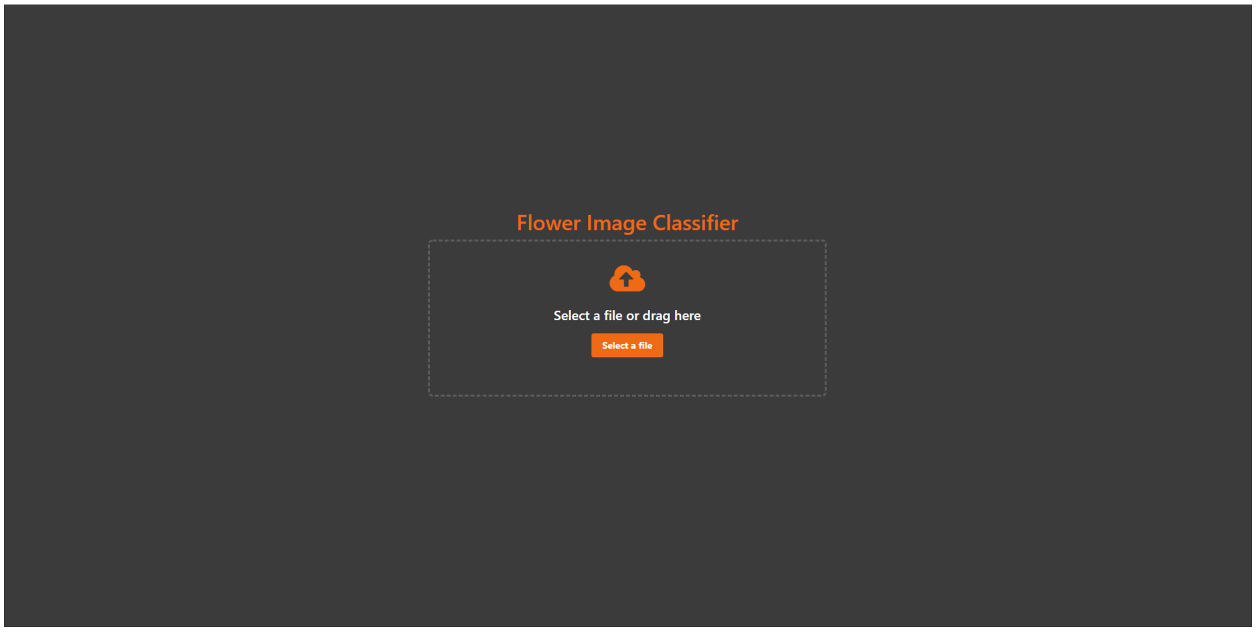


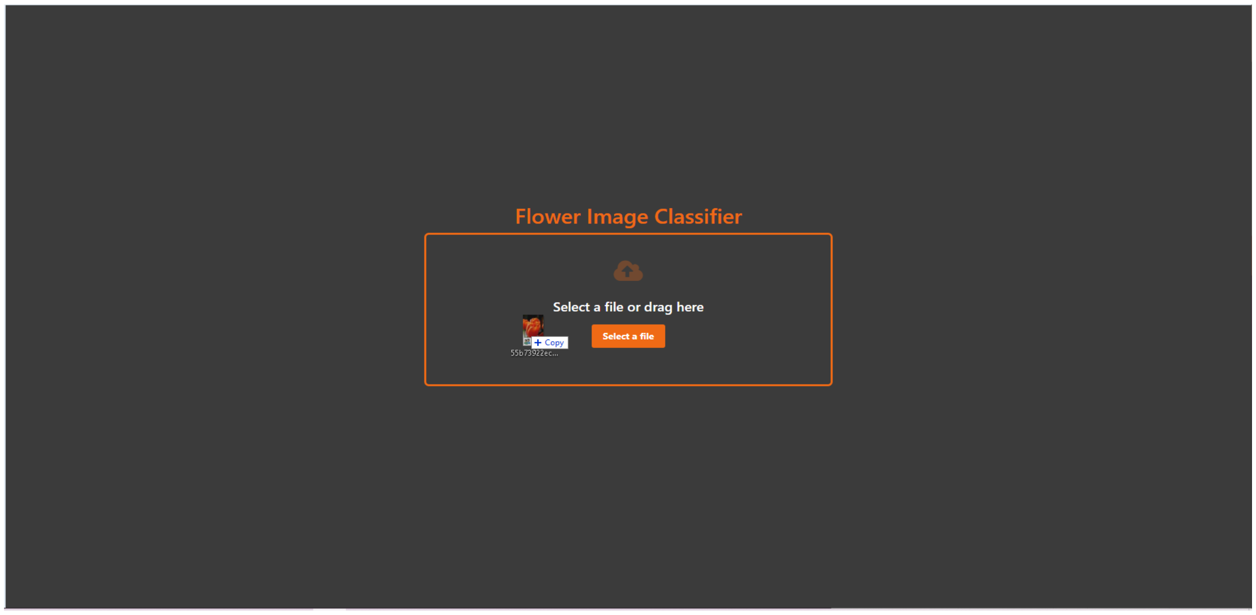
```python
import os
import numpy as np
import tensorflow as tf
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename

# Define a flask app
app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = -1
MODEL_PATH = 'model/model.h5'

# Load your trained model
model = tf.keras.models.load_model(MODEL_PATH)

IMG_HEIGHT = 224
IMG_WIDTH = 224
class_names = ['Daisy', 'Dandelion', 'Roses', 'Sunflowers', 'Tulips']

# Predict
def model_predict(img_path,model):

    img = tf.keras.utils.load_img(img_path, target_size=(IMG_HEIGHT, IMG_WIDTH))
    img_array = tf.keras.utils.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0) # Create a batch

    preds = model.predict(img_array)
    return preds

# Decode the prediction of the model
def decode_prediction(predictions):
    score =  tf.nn.softmax(predictions[0])
    percentage_acc = 100 * np.max(score)
    return "{} with {:.2f}% Confidence".format(class_names[np.argmax(score)],percentage_acc)

@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')


```

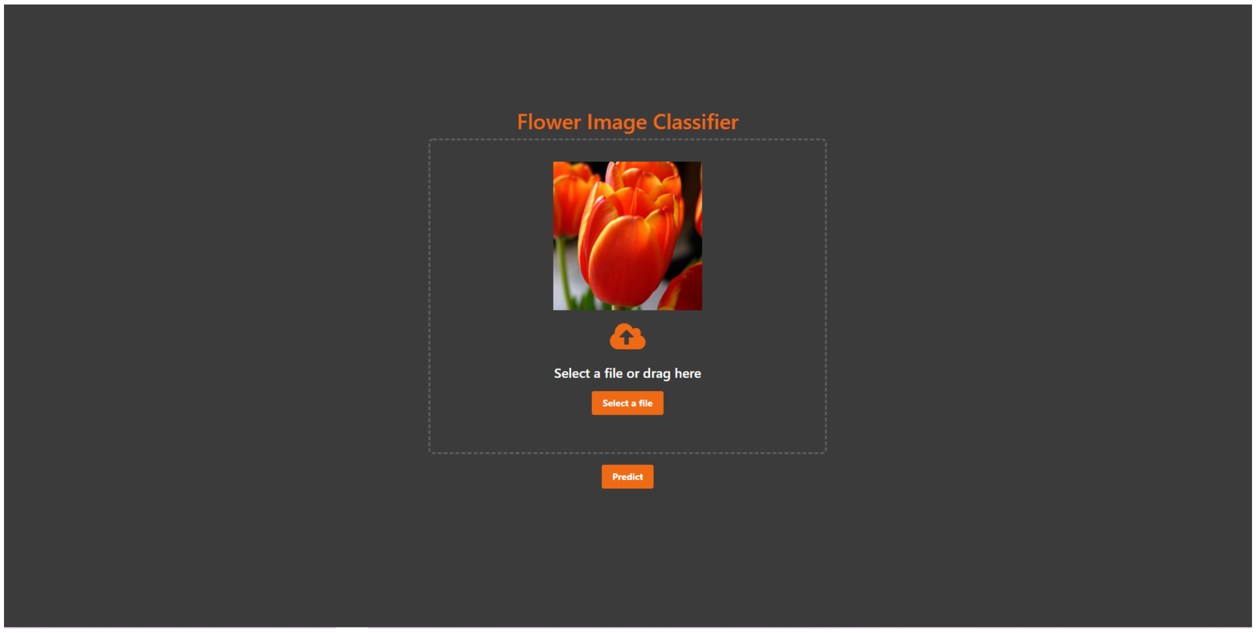


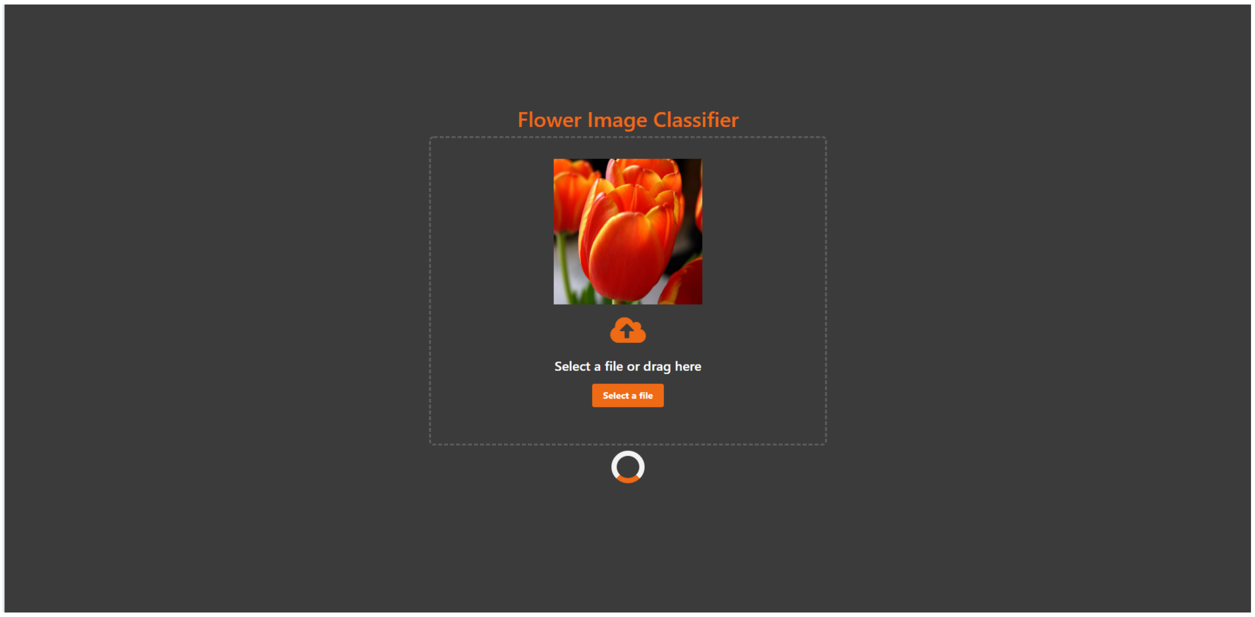
```
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        preds = model_predict(file_path, model)

        # Process result
        result = decode_prediction(preds)
        return result
    return None


if __name__ == '__main__':
    app.run(debug=True)
```

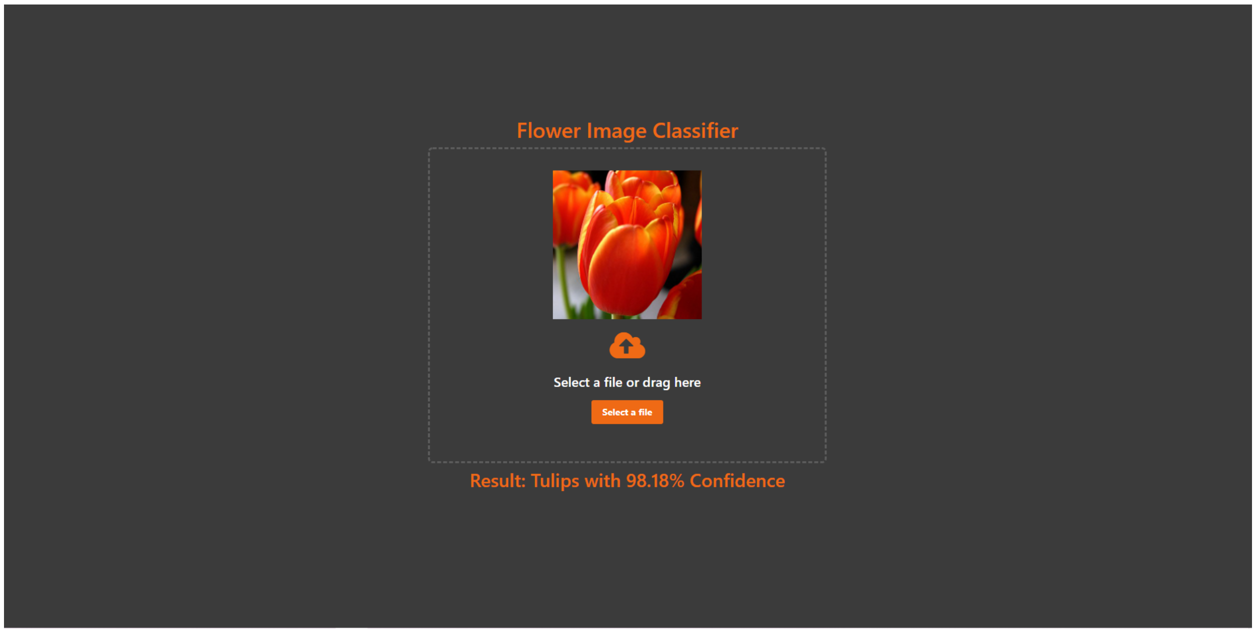
Demo




Flower Image Classifier
Select a file or drag here
+ Copy
55b73922ec...
Select a file


Flower Image Classifier
Select a file or drag here
Select a file
Predict


Flower Image Classifier
Select a file or drag here
Select a file



Flower Image Classifier

Select a file or drag here

Result: Tulips with 98.18% Confidence