**Phase 3: Development Part 1**

Introduction:

This document describes the work we did and the steps we took in the first phases of developing our project with IBM Cloud Foundry. The first section of development is devoted to project setup, feature development, and component configuration.

PYTHON PROGRAM:

import cv2

from deepface import DeepFace

from flask import Flask, render\_template, request, redirect, url\_for

import os

app = Flask(\_\_name\_\_)

def is\_human(img):

    img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

    faces = face\_cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=2, minSize=(30, 30))

    return bool(len(faces) > 0)

def get\_emotion(img):

    img = cv2.imread(img)

    if is\_human(img):

        result = DeepFace.analyze(img, actions=['emotion'], enforce\_detection=False)

        emotions\_score = result[0]['emotion']

        emotion = max(emotions\_score, key=emotions\_score.get)

        return emotion

    else:

        return "No human face detected"

UPLOAD\_FOLDER = 'static/uploads'

@app.route('/')

def index():

    # os.remove("static/uploads/uploaded\_image.jpg")

    return render\_template('index.html', img\_path='', label="Select Image", preview\_style="")

@app.route('/upload', methods=['POST'])

def upload\_image():

    if 'image' not in request.files:

        return "No image part"

    image = request.files['image']

    if image.filename == '':

        return "No selected image file"

    if not os.path.exists(UPLOAD\_FOLDER):

        os.makedirs(UPLOAD\_FOLDER)

    app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

    image.save(os.path.join(app.config['UPLOAD\_FOLDER'], 'uploaded\_image.jpg'))

    emotion = "Emotion: " + get\_emotion('static/uploads/uploaded\_image.jpg').capitalize()

    return render\_template('index.html', emotion=emotion, label="Change Image", preview\_style="display: inline;")

@app.errorhandler(405)

def method\_not\_allowed(error):

    return redirect(url\_for('index'))

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Application Development:

HTML:

<!DOCTYPE html>

<html>

<head>

    <title>Emotion Detection</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <div id="main-container">

        <h1>Emotion Detection</h1>

        <form action="/upload" method="POST" enctype="multipart/form-data">

            <label for="image-input">{{ label }}</label>

            <input type="file" accept="image/\*" id="image-input" name="image" onchange="previewImage(event)" required>

            <input type="submit" id="show-emotion-button" onclick="showEmotion()" value="Show Emotion"></input>

        </form>

        <div id="image-preview">

            <img id="preview-image" src={{ url\_for('static', filename='uploads/uploaded\_image.jpg') }} alt="Image Preview" style='{{ preview\_style }}'>

        </div>

        <div id="emotion-result">

            <span id="emotion-text">{{ emotion }}</span>

        </div>

    </div>

    <script src="{{ url\_for('static', filename='main.js') }}"></script>

</body>

</html>

Style.css:

body {

    font-family: Arial, sans-serif;

    background: rgb(212, 212, 212);

    background-position: center;

    background-size: cover;

    background-position: fixed;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}

#main-container {

    background-color: #ffffff59;

    border-radius: 10px;

    box-shadow: 0 0 50px 20px rgba(0, 0, 0, 0.2);

    padding: 20px;

    text-align: center;

    transition: transform 0.3s;

}

#main-container:hover {

    transform: scale(1.05);

}

h1 {

    color: #333;

}

input[type="file"] {

    display: none;

}

label {

    background-color: #3498db;

    color: #fff;

    padding: 10px 20px;

    cursor: pointer;

}

label:hover {

    background-color: #2c82c4;

}

#image-preview {

    margin-top: 20px;

}

#image-preview img {

    max-width: 100%;

    max-height: 300px;

    display: none;

}

#emotion-result {

    margin-top: 20px;

    font-size: 20px;

    font-weight: 700;

    color: black;

}

#show-emotion-button {

    background-color: #3498db;

    color: #fff;

    padding: 10px 20px;

    cursor: pointer;

    margin-top: 20px;

    display: none;

}

#show-emotion-button:hover {

    background-color: #2c82c4;

}

Main.js

function previewImage(event) {

    document.getElementById("emotion-result").style.display="none";

    document.querySelector('label[for="image-input"]').textContent = "Change Image";

    const input = event.target;

    const preview = document.getElementById("preview-image");

    document.getElementById("preview-image").style.display = "inline";

    if (input.files.length > 0) {

        const reader = new FileReader();

        reader.onload = function() {

            preview.src = reader.result;

        };

        reader.readAsDataURL(input.files[0]);

    } else {

        preview.src = "";

    }

    document.getElementById("show-emotion-button").style.display = "inline";

}

function showEmotion(event) {

    // document.getElementById("preview-image").src="../uploads/uploaded\_image.jpg";

}

Conclusion :

**The work completed and results attained during "Development Part 1" of our IBM Cloud Foundry project are compiled in this document. The foundation for additional development, configuration, and testing in the following stages has been laid by these preliminary actions.**