Laporan Project PCV
“Basal Cell Carcinoma (BCC) Detector Using Computer Vision”

Nama: Hanifa Bunayya

NIM : 2155301056

Kelas : 4 TI E

Dosen Pengampu : Ananda, S.Kom., M.T., Ph.D.

ILB : Mhd.Anwar, S.Tr.Kom

PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK CALTEX RIAU

T.A 2024/2025

I. Deskripsi:

Kanker kulit adalah salah satu jenis kanker yang paling umum di dunia, dan Basal Cell Carcinoma (BCC) merupakan bentuk kanker kulit yang paling sering ditemukan. Deteksi dini BCC sangat krusial untuk keberhasilan penanganan dan prognosis pasien. Namun, identifikasi BCC secara manual seringkali memerlukan keahlian dermatologis dan dapat bersifat subjektif, berpotensi menyebabkan keterlambatan diagnosis atau kesalahan.

Dalam era digital saat ini, teknologi, khususnya Computer Vision, menawarkan solusi yang efektif dan efisien untuk membantu mendeteksi indikasi BCC secara otomatis. Hal ini menjadi penting sebagai alat skrining awal dan pendukung keputusan bagi profesional medis, serta dapat meningkatkan kesadaran masyarakat akan pentingnya deteksi dini.

Proyek ini bertujuan untuk mengembangkan sistem klasifikasi otomatis untuk mendeteksi potensi Basal Cell Carcinoma berdasarkan citra digital (gambar) lesi kulit. Sistem ini menggunakan model deep learning (CNN dengan arsitektur ResNet50) untuk mengenali karakteristik visual BCC dan mengklasifikasikannya ke dalam dua kategori, yaitu:

- BCC (Positif): Gambar menunjukkan karakteristik yang sangat mungkin mengindikasikan Basal Cell Carcinoma.
- Non-BCC (Negatif): Gambar menunjukkan lesi kulit yang tidak mengindikasikan Basal Cell Carcinoma.

Gambar lesi kulit diproses dengan teknik augmentasi data dan dilatih menggunakan model klasifikasi citra untuk mencapai akurasi prediksi yang tinggi. Model yang dihasilkan kemudian akan diintegrasikan ke dalam antarmuka web, memungkinkan akses yang mudah dan cepat bagi pengguna.

Dataset: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

II. Manfaat

- Meningkatkan Akses ke Skrining Awal: Menyediakan alat skrining awal yang mudah diakses, terutama di daerah dengan keterbatasan akses dermatologis.

- Mendukung Keputusan Klinis: Memberikan panduan awal dan alat bantu bagi tenaga medis dalam mengidentifikasi kasus yang memerlukan pemeriksaan lebih lanjut oleh dermatolog.
- Mengurangi Beban Kerja Medis: Membantu memprioritaskan kasus yang dicurigai BCC, sehingga mengurangi waktu dan sumber daya yang dihabiskan untuk lesi jinak.
- Meningkatkan Kesadaran Masyarakat: Mendorong individu untuk melakukan skrining mandiri dan mencari bantuan medis jika ada indikasi yang mencurigakan.

III. Tujuan

- Mengembangkan sistem klasifikasi otomatis untuk mendeteksi potensi Basal Cell Carcinoma berdasarkan gambar lesi kulit.
- Melatih dan menguji performa model deep learning dalam membedakan antara lesi yang dicurigai BCC dan lesi non-BCC.
- Mengembangkan antarmuka web yang intuitif untuk mengunggah gambar dan menerima hasil klasifikasi.

IV. Fitur Utama

- Deteksi Potensi BCC Otomatis: Mengklasifikasikan gambar lesi kulit ke dalam kategori BCC atau Non-BCC.

V. Teknologi yang Digunakan

- Google colab digunakan untuk pembuatan model
- VSCode digunakan untuk implementasi ke website

VI. Implementasi

a. Pembuatan Model

1. Mount dan Ekstrak Dataset

```
[1] from google.colab import drive
    drive.mount('/content/drive')

    import zipfile
    zip_path = '/content/drive/MyDrive/ Hanifa Bunayya-Computer Vision Project/HAM10000.zip'
    extract_path = '/content/'

    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_path)

    print("Selesai ekstrak!")

    Mounted at /content/drive
    Selesai ekstrak!
```

- Menghubungkan Google Drive ke Colab.
- Ekstrak file ZIP dataset HAM10000 ke folder /content/.

```python
import pandas as pd
import shutil
import os

# Path metadata dan folder hasil ekstrak
metadata_path = '/content/drive/MyDrive/ Hanifa Bunayya-Computer Vision Project/HAM10000_metadata.csv'
image_dir = '/content/HAM10000'  # hasil ekstrak ZIP
output_dir = '/content/bcc_binary'

bcc_dir = os.path.join(output_dir, 'bcc')
non_bcc_dir = os.path.join(output_dir, 'non_bcc')

os.makedirs(bcc_dir, exist_ok=True)
os.makedirs(non_bcc_dir, exist_ok=True)

df = pd.read_csv(metadata_path)

# Salin gambar sesuai label
for _, row in df.iterrows():
    img_id = row['image_id']
    label = row['dx']
    src = os.path.join(image_dir, img_id + '.jpg')

    if not os.path.exists(src):
        continue

    dst_dir = bcc_dir if label == 'bcc' else non_bcc_dir
    shutil.copy(src, os.path.join(dst_dir, img_id + '.jpg'))

print("✅ Selesai memisahkan gambar ke bcc dan non_bcc.")
```

```
✅ Selesai memisahkan gambar ke bcc dan non_bcc.
```

- Baca metadata dan gambar.
- Buat dua folder: bcc/ dan non_bcc/.
- Memisahkan gambar ke dua folder berdasarkan label ‘bcc’ atau ‘non_bcc’.

```
[ ]  import random

     source_dir = '/content/bcc_only/bcc'
     split_base = '/content/bcc_binary_split'

     # Buat folder
     for split in ['train', 'val', 'test']:
         os.makedirs(os.path.join(split_base, split, 'bcc'), exist_ok=True)

     files = os.listdir(source_dir)
     random.shuffle(files)

     n = len(files)
     train_split = int(0.7 * n)
     val_split = int(0.85 * n)

     train_files = files[:train_split]
     val_files = files[train_split:val_split]
     test_files = files[val_split:]

     for f in train_files:
         shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'train', 'bcc', f))
     for f in val_files:
         shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'val', 'bcc', f))
     for f in test_files:
         shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'test', 'bcc', f))

     print("✅ Dataset BCC berhasil di-split.")

     ✅ Dataset BCC berhasil di-split.
```

- Buat struktur folder: train/bcc, train/non_bcc, dll.
- Pisahkan data untuk pelatihan, validasi, dan pengujian.

```
     import os
     import shutil
     import random

     source_base = '/content/bcc_binary'
     split_base = '/content/bcc_binary_split'

     # Buat folder struktur train/val/test untuk kedua kelas
     for split in ['train', 'val', 'test']:
         for cls in ['bcc', 'non_bcc']:
             os.makedirs(os.path.join(split_base, split, cls), exist_ok=True)

     # Fungsi untuk split dan copy file
     def split_and_copy(source_dir, class_name):
         files = os.listdir(source_dir)
         random.shuffle(files)

         n = len(files)
         train_split = int(0.7 * n)
         val_split = int(0.85 * n)

         train_files = files[:train_split]
         val_files = files[train_split:val_split]
         test_files = files[val_split:]

         for f in train_files:
             shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'train', class_name, f))
         for f in val_files:
             shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'val', class_name, f))
         for f in test_files:
             shutil.copy(os.path.join(source_dir, f), os.path.join(split_base, 'test', class_name, f))
```

```
# Lakukan split untuk kedua kelas
split_and_copy(os.path.join(source_base, 'bcc'), 'bcc')
split_and_copy(os.path.join(source_base, 'non_bcc'), 'non_bcc')

print("✅ Dataset berhasil di-split menjadi train, val, dan test untuk kedua kelas.")
```

```
✅ Dataset berhasil di-split menjadi train, val, dan test untuk kedua kelas.
```

```
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import tensorflow as tf

    datagen = ImageDataGenerator(rescale=1./255)

    train_gen = datagen.flow_from_directory(
        '/content/bcc_binary_split/train',
        target_size=(224, 224),
        batch_size=32,
        class_mode='binary'
    )

    val_gen = datagen.flow_from_directory(
        '/content/bcc_binary_split/val',
        target_size=(224, 224),
        batch_size=32,
        class_mode='binary'
    )

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(224, 224, 3)),
        tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1, activation='sigmoid')  # 1 neuron = prediksi kemiripan dengan BCC
    ])

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    model.fit(train_gen, validation_data=val_gen, epochs=10)
```

```
Found 359 images belonging to 1 classes.
Found 77 images belonging to 1 classes.
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its
  self._warn_if_super_not_called()
Epoch 1/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 47s 4s/step - accuracy: 0.7064 - loss: 0.2170 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 2/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 72s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 40s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 33s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 34s 3s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
<keras.src.callbacks.history.History at 0x7b23523d9590>
```

- Model CNN sederhana:
- 2 layer Conv2D
- 2 MaxPooling
- Dense 64
- Output sigmoid untuk binary classification
- Latih model selama 10 epoch.

```
[ ] model.save('/content/drive/MyDrive/ Hanifa Bunayya-Computer Vision Project/bcc_model.h5')

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the n
```

- Simpan model ke Google Drive.

```
[ ] test_gen = datagen.flow_from_directory(
        '/content/bcc_binary_split/test',
        target_size=(224, 224),
        batch_size=32,
        class_mode='binary'
    )

    test_loss, test_acc = model.evaluate(test_gen)
    print(f"Akurasi data test: {test_acc:.4f}, Loss: {test_loss:.4f}")

Found 78 images belonging to 1 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 767ms/step - accuracy: 1.0000 - loss: 0.0000e+00
Akurasi data test: 1.0000, Loss: 0.0000
```

- Uji model pada data test dan tampilkan akurasi.

```
[ ] !pip install flask-ngrok

Collecting flask-ngrok
  Downloading flask_ngrok-0.0.25-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.11/dist-packages (from flask-ngrok) (3.1.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from flask-ngrok) (2.32.3)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (1.9.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (8.2.1)
Requirement already satisfied: itsdangerous>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (3.0.2)
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok) (3.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngrok) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngrok) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngrok) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngrok) (2025.6.15)
Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

- Install library untuk membuat server dan akses publik via ngrok.

```
[ ] !pip install pyngrok

Collecting pyngrok
  Downloading pyngrok-7.2.11-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.11/dist-packages (from pyngrok) (6.0.2)
Downloading pyngrok-7.2.11-py3-none-any.whl (25 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.2.11
```

```
[ ] from pyngrok import ngrok
    ngrok.set_auth_token("2zKFWFvfJG19UYHhG4mRk231Rh0_4DXVtZYP3heL8hSiFbLXt")
    public_url = ngrok.connect(5000)
    print(public_url)

NgrokTunnel: "https://6958-34-75-60-137.ngrok-free.app" -> "http://localhost:5000"
```

- Buat URL publik ngrok untuk akses Flask API.

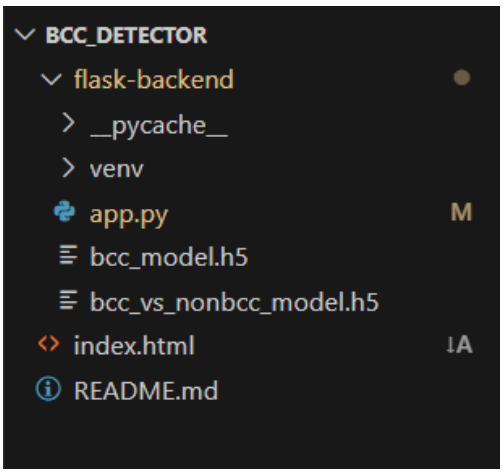
```
[ ] %%writefile app.py
    from flask import Flask, request, jsonify
    from flask_cors import CORS
    import tensorflow as tf
    import numpy as np
    from PIL import Image

    app = Flask(__name__)
    CORS(app)

    # Ganti path di bawah dengan path model kamu di lokal
    model = tf.keras.models.load_model('bcc_vs_nonbcc_model.h5')

    def preprocess(img):
        img = img.resize((224, 224))
        img = np.array(img) / 255.0
        return np.expand_dims(img, 0)

    @app.route("/predict", methods=["POST"])
    def predict():
        try:
            file = request.files['image']
            img = Image.open(file.stream).convert('RGB')  # Pastikan selalu 3 channel

            input_tensor = preprocess(img)
            pred = model.predict(input_tensor)[0][0]
            print("Prediksi mentah:", pred)  # Debug confidence

            result = "Basal Cell Carcinoma" if pred >= 0.5 else "Non-Basal Cell Carcinoma"

            return jsonify({
                "result": result,
                "confidence": float(pred)
            })
        except Exception as e:
            return jsonify({"error": str(e)}), 500

    if __name__ == "__main__":
        app.run(port=5050)

    Writing app.py
```

- Tulis file Flask app.py.
- Terima gambar dari pengguna, prediksi apakah BCC atau bukan, lalu kembalikan hasil dan confidence.

```
[ ] from google.colab import files
    files.download('app.py')
```

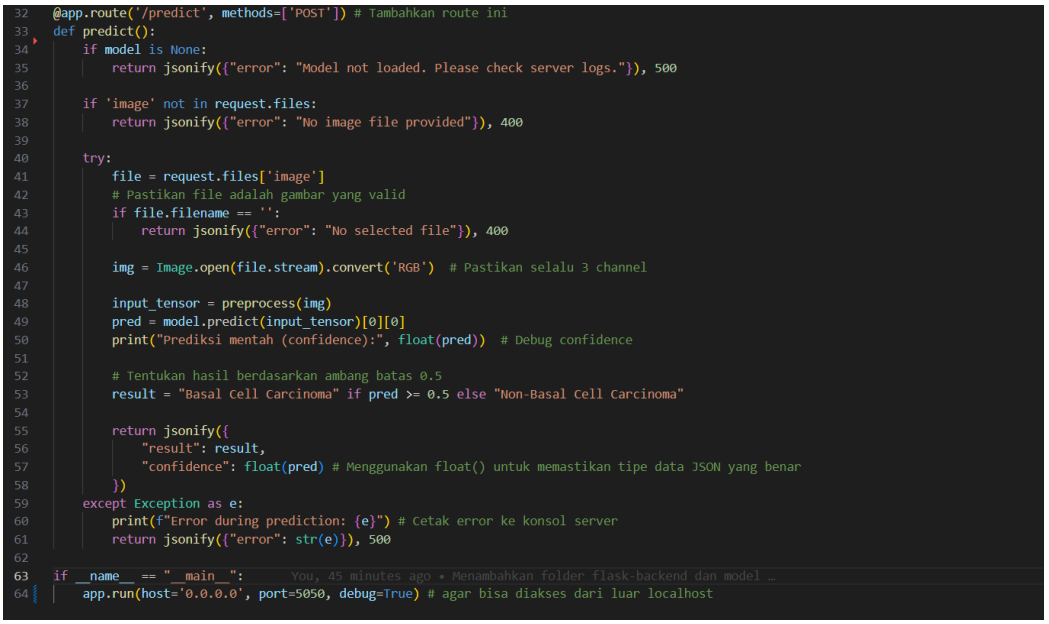
- Unduh file app.py ke lokal untuk digunakan.

b. Implementasi Model ke dalam Website



- Struktur projek dengan folder flask-backend berisi model dan app.py. Lalu file index.html berisi tampilan halaman website.

```
flask-backend > app.py > ...
You, 2 seconds ago | 1 author (You)
from flask import Flask, request, jsonify, render_template
from flask_cors import CORS
import tensorflow as tf
import numpy as np
from PIL import Image
import os # Import os module for path handling

app = Flask(__name__)
CORS(app)

MODEL_PATH = 'bcc_vs_nonbcc_model.h5'

# Periksa apakah model ada sebelum mencoba memuatnya
if not os.path.exists(MODEL_PATH):
    print(f"Error: Model file not found at {MODEL_PATH}")
try:
    model = tf.keras.models.load_model(MODEL_PATH)
    print("Model loaded successfully.")
except Exception as e:
    print(f"Error loading model: {e}")
    model = None # Set model to None if loading fails

def preprocess(img):
    img = img.resize((224, 224))
    img = np.array(img) / 255.0
    return np.expand_dims(img, 0)

@app.route('/')
def home():
    return render_template('index.html')
```



```
@app.route('/predict', methods=['POST']) # Tambahkan route ini
def predict():
    if model is None:
        return jsonify({"error": "Model not loaded. Please check server logs."}), 500

    if 'image' not in request.files:
        return jsonify({"error": "No image file provided"}), 400

    try:
        file = request.files['image']
        # Pastikan file adalah gambar yang valid
        if file.filename == '':
            return jsonify({"error": "No selected file"}), 400

        img = Image.open(file.stream).convert('RGB')  # Pastikan selalu 3 channel

        input_tensor = preprocess(img)
        pred = model.predict(input_tensor)[0][0]
        print("Prediksi mentah (confidence):", float(pred))  # Debug confidence

        # Tentukan hasil berdasarkan ambang batas 0.5
        result = "Basal Cell Carcinoma" if pred >= 0.5 else "Non-Basal Cell Carcinoma"

        return jsonify({
            "result": result,
            "confidence": float(pred) # Menggunakan float() untuk memastikan tipe data JSON yang benar
        })
    except Exception as e:
        print(f"Error during prediction: {e}") # Cetak error ke konsol server
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":      You, 45 minutes ago • Menambahkan folder flask-backend dan model …
    app.run(host='0.0.0.0', port=5050, debug=True) # agar bisa diakses dari luar localhost
```

```
You, 49 minutes ago | 1 author (You)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>BCC Skin Cancer Detection</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
    <style>
        :root {
            --primary-blue: #4a90e2;
            --dark-blue: #2c5e9a;
            --light-gray: #f5f7fa;
            --medium-gray: #e0e6ed;
            --dark-gray: #334e68;
            --red-alert: #e74c3c;
            --green-success: #2ecc71;
            --text-color: #333;
        }

        body {
            font-family: 'Roboto', sans-serif;
            background-color: var(--light-gray);
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            color: var(--text-color);
            line-height: 1.6;
        }

        h1 {
            margin-top: 2.5rem;
            margin-bottom: 1.5rem;
            color: var(--dark-blue);
```

```
            font-weight: 700;
            text-align: center;
            letter-spacing: 0.5px;
        }

        .container {
            background: white;
            padding: 2.5rem 2rem;
            border-radius: 15px;
            box-shadow: 0 8px 25px rgba(0,0,0,0.1);
            width: 90%;
            max-width: 550px;
            text-align: center;
            border: 1px solid var(--medium-gray);
        }

        .form-group {
            margin-bottom: 1.5rem;
        }

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

        .custom-file-upload {
            border: 2px dashed var(--primary-blue);
            padding: 1.5rem;
            border-radius: 8px;
            cursor: pointer;
            display: block;
            margin-bottom: 1rem;
            transition: background-color 0.3s ease, border-color 0.3s ease;
            font-size: 1rem;
```

```
        color: var(--primary-blue);
        font-weight: 500;
    }

    .custom-file-upload:hover {
        background-color: #e6f0fa;
        border-color: var(--dark-blue);
    }

    #file-name {
        margin-top: 0.5rem;
        font-size: 0.9rem;
        color: var(--dark-gray);
        min-height: 20px;
    }

    button {
        background-color: var(--primary-blue);
        color: white;
        padding: 0.9rem 1.8rem;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        width: 100%;
        font-size: 1.1rem;
        font-weight: 600;
        transition: background-color 0.3s ease, transform 0.2s ease;
        box-shadow: 0 4px 10px rgba(0, 145, 255, 0.2);
    }

    button:hover {
        background-color: var(--dark-blue);
        transform: translateY(-2px);
```

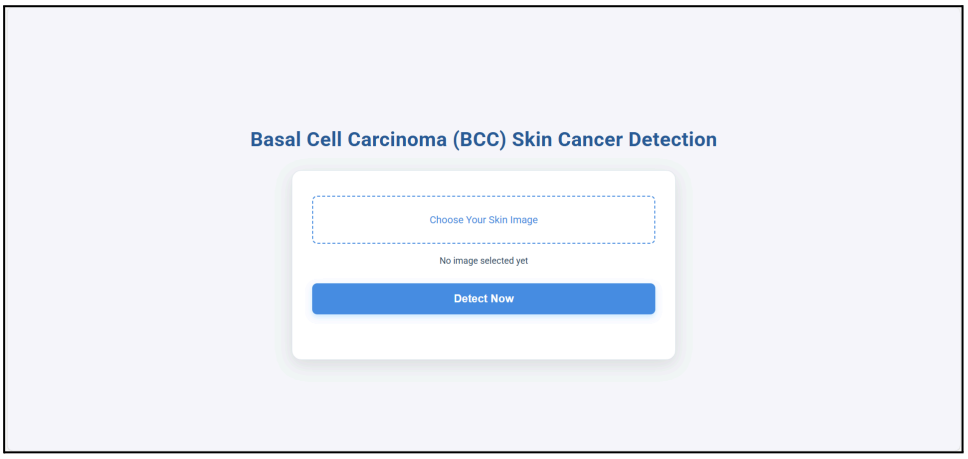
```
    }

    #loader {
        display: none;
        margin-top: 2rem;
        font-size: 1.1rem;
        color: var(--primary-blue);
        animation: pulse 1.5s infinite;
    }

    @keyframes pulse {
        0% { opacity: 0.7; }
        50% { opacity: 1; }
        100% { opacity: 0.7; }
    }

    #result {
        margin-top: 2rem;
        font-size: 1.3rem;
        text-align: center;
        opacity: 0;
        transform: translateY(10px);
        transition: opacity 0.5s ease-out, transform 0.5s ease-out;
    }

    #result.show {
        opacity: 1;
        transform: translateY(0);
    }

    .result-item {
        margin-bottom: 0.8rem;
    }
```

```
        .highlight-positive {
            font-weight: bold;
            color: var(--red-alert);
            font-size: 1.4rem;
        }

        .highlight-negative {       You, 47 minutes ago • Menambahkan folder flask-backend dan model
            font-weight: bold;
            color: var(--green-success);
            font-size: 1.4rem;
        }

        .error-message {
            color: var(--red-alert);
            font-weight: 500;
        }
    </style>
</head>
<body>
    <h1>Basal Cell Carcinoma (BCC) Skin Cancer Detection</h1>
    <div class="container">
        <form id="prediction-form" onsubmit="predictImage(event)">
            <div class="form-group">
                <label for="image" class="custom-file-upload">Choose Your Skin Image</label>
                <input type="file" id="image" accept="image/*" required />
                <div id="file-name">No image selected yet</div>
            </div>
            <button type="submit">Detect Now</button>
        </form>
        <div id="loader">Processing image... Please wait...</div>
        <div id="result"></div>
    </div>
```

```
    <script>
        document.getElementById('image').addEventListener('change', function () {
            const fileNameDisplay = document.getElementById('file-name');
            if (this.files.length > 0) {
                fileNameDisplay.textContent = `Selected image: ${this.files[0].name}`;
            } else {
                fileNameDisplay.textContent = 'No image selected yet';
            }
        });

        async function predictImage(event) {
            event.preventDefault();

            const formData = new FormData();
            const imageInput = document.getElementById('image');
            const resultText = document.getElementById('result');
            const loader = document.getElementById('loader');

            resultText.classList.remove('show');
            resultText.innerHTML = "";

            if (imageInput.files.length === 0) {
                alert("Please select an image first.");
                return;
            }

            formData.append("image", imageInput.files[0]);
            loader.style.display = "block";

            try {
                const response = await fetch("http://127.0.0.1:5050/predict", {
                    method: "POST",
                    body: formData
                });
```

```
                const data = await response.json();
                console.log("Response:", data);

                let resultClass = '';
                let resultMessage = '';

                if (data.class) {
                    if (data.class === 'malignant') {
                        resultClass = 'highlight-positive';
                        resultMessage = 'Cancer Detected (Malignant)';
                    } else {
                        resultClass = 'highlight-negative';
                        resultMessage = 'No Cancer Detected (Benign)';
                    }

                    resultText.innerHTML = `
                        <p class="result-item"><strong>Detection Result:</strong> <span class="${resultClass}">${resultMessage}</span></p>
                        <p class="result-item"><strong>Confidence:</strong> ${Math.round(data.confidence * 100)}%</p>
                    `;
                } else if (data.result) {
                    resultMessage = data.result;

                    if (data.result.includes('Cancer')) {
                        resultClass = 'highlight-positive';
                    } else {
                        resultClass = 'highlight-negative';
                    }

                    resultText.innerHTML = `
                        <p class="result-item"><strong>Detection Result:</strong> <span class="${resultClass}">${resultMessage}</span></p>
                        <p class="result-item"><strong>Confidence:</strong> ${Math.round(data.confidence * 100)}%</p>
                    `;
                } else if (data.error) {
                    resultText.innerHTML = `<p class="error-message">Error from server: ${data.error}</p>`;
                } else {
                    resultText.innerHTML = `<p class="error-message">Unexpected response format.</p>`;
                }

                resultText.classList.add('show');

            } catch (error) {
                resultText.innerHTML = `<p class="error-message">An error occurred: ${error.message}</p>`;
                resultText.classList.add('show');
            } finally {
                loader.style.display = "none";
            }
        }
    </script>
</body>
</html>
```

c. Hasil



Basal Cell Carcinoma (BCC) Skin Cancer Detection

Choose Your Skin Image

Selected image: bcc2.jpeg

Detect Now

Detection Result: Basal Cell Carcinoma

Confidence: 96%

Basal Cell Carcinoma (BCC) Skin Cancer Detection

Choose Your Skin Image

Selected image: jerawat.jpeg

Detect Now

Detection Result: Non-Basal Cell Carcinoma

Confidence: 23%

Basal Cell Carcinoma (BCC) Skin Cancer Detection

Choose Your Skin Image

Selected image: kering.jpeg

Detect Now

Detection Result: Non-Basal Cell Carcinoma

Confidence: 32%

Choose Your Skin Image

Selected image: ruam.jpg

Detect Now

Detection Result: Basal Cell Carcinoma

Confidence: 94%