A

Course End Project Report on

SKIN CANCER DETECTION

Submitted in the Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering(AI&ML)

Submitted by

K Rishi Kesava 222P1A3320

Under the esteemed guidance of

Dr.P.Pavankumar

Department of Computer Science and Engineering (AI&ML)

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

(Accredited by NAAC with "A" Grade and Accredited by NBA (CE, EEE, ECE, CSE))

(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)

VIDYA NAGAR, PALLAVOLU (V), PRODDATUR-516360, Y.S.R. (Dt.).A.P

2025_2026

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

(Accredited by NAAC with "A" Grade and Accredited by NBA (CE, EEE, ECE, CSE))
(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)

VIDYA NAGAR, PALLAVOLU , PRODDATUR-516360, Y.S.R. (Dt.), A.P

Department of Computer Science and Engineering (AI&ML)

CERTIFICATE

This is to certify that the project titled "Skin Cancer Detection", using Python, Html, Css is carried out by

K. Rishi Kesava **222P1A3320**

In partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (AI&ML)** during the year 2025-26.

Signature of the Supervisor
Dr.P.Pavankumar
Professor

Signature of the HOD
Ms. Krupa Sagar
HOD, CSE (AI&ML)

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

I wish to express my deep sense of gratitude to Dr. P Pavan Kumar, Professor and Project Supervisor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, for his able guidance and useful suggestions, which helped me in completing the project in time.

I am particularly thankful to Ms.Krupa Sagar ,Head of the Department, Department of Computer Science and Engineering (AI&ML), her guidance, intense support and encouragement, which helped us to mould my project into a successful one.

I show gratitude to my honorable Principal Dr. S. Sruthi and Director Admin

Dr. G. Sreenivasula Reddy for providing all facilities and support.

I avail this opportunity to express my deep sense of gratitude and heart-full thanks to Sree V. Jaya Chandra Reddy, Chairman and Sree V. Lohit Reddy, CEO of CBIT, for providing a congenial atmosphere to complete this project successfully.

I also thank all the staff members of Computer Science and Engineering (AI&ML) department for their valuable support and generous advice. Finally thanks to all my friends and family members for their continuous support and enthusiastic help.

K. Rishi Kesava

222P1A3320

ABSTRACT

This project is a web-based **Skin Cancer Detection System** that assists users in predicting whether a skin lesion is **benign** or **malignant**. The application has two main components: a **frontend** and a **backend**. The frontend, built with **HTML, CSS, and JavaScript**, provides a user-friendly interface where an image of a skin lesion can be uploaded. The backend, powered by **Python, Flask, and TensorFlow/Keras**, processes the image using a trained **Convolutional Neural Network (CNN)** model to classify the lesion.

The backend system is implemented in *app.py*, which loads the pre-trained CNN model for inference. It handles two main routes: the root route (/), which delivers the main HTML file, and the **/predict** route, which accepts uploaded images, preprocesses them, and returns the prediction results in **JSON format**. The model has been trained on a skin cancer dataset containing multiple lesion categories such as melanoma, basal cell carcinoma, and keratosis, which are further mapped into binary classification (benign or malignant) for practical diagnosis support.

The frontend, defined in *index.html*, captures the uploaded image and communicates with the backend using an **AJAX (Fetch API) POST request**. Once the prediction results are received, the system dynamically displays the classification label along with a **confidence score** and a visual indicator. This architecture ensures a clear separation of concerns, with the frontend managing presentation and user interaction, while the backend performs model inference and decision-making.

By combining **deep learning-based image analysis** with an **interactive web interface**, this project demonstrates how modern web technologies and AI can be integrated to provide an accessible, fast, and user-friendly tool for preliminary skin cancer detection.

Keywords:

- Skin Cancer Detection
- Benign and Malignant Classification
- Convolutional Neural Network (CNN)
- Deep Learning
- Flask
- TensorFlow/Keras
- Python Backend
- HTML/CSS/JavaScript Frontend
- API Endpoint (/predict)
- Image Upload and Prediction

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

Skin cancer is one of the most prevalent and rapidly growing health concerns worldwide. It occurs when abnormal skin cells grow uncontrollably, usually due to DNA damage caused by prolonged exposure to ultraviolet (UV) radiation from the sun or artificial sources such as tanning beds. According to global health reports, millions of new cases of skin cancer are diagnosed each year, making it a significant public health challenge.

Among the different forms of skin cancer, **melanoma** is considered the most aggressive and life-threatening type, capable of spreading quickly to other organs if not detected at an early stage. On the other hand, non-melanoma skin cancers such as basal cell carcinoma, squamous cell carcinoma, and benign conditions like moles and keratoses are less fatal but still require accurate diagnosis and treatment. A major difficulty arises from the fact that benign and malignant lesions often appear very similar in visual characteristics, making manual diagnosis challenging.

Traditional methods of detection rely on dermatologists' clinical experience, dermoscopic examination, and sometimes biopsy. While effective, these methods are time-consuming, resource-intensive, and subject to human error or misinterpretation. In regions with limited access to healthcare specialists, early diagnosis becomes even more difficult, leading to delayed treatment and higher mortality rates.

With the rise of **Artificial Intelligence (AI)** and **Deep Learning**, particularly **Convolutional Neural Networks (CNNs)**, automated computer-aided diagnostic systems have shown remarkable potential in the medical imaging domain. CNNs can learn intricate patterns and features from dermoscopic images that may not be easily noticeable to the human eye. As a result, AI-based systems can provide faster, more consistent, and highly accurate classification of skin lesions.

This project focuses on developing a **Skin Cancer Detection System** using deep learning techniques to classify lesions into two categories: **benign** or **malignant**. The system acts as a decision-support tool for dermatologists, reducing dependency on manual interpretation,

increasing diagnostic efficiency, and ultimately contributing to early detection, better treatment planning, and improved survival rates for patients worldwide.

1.2 Objective of the Project

The main objectives of this project are:

1. To design and develop a **deep learning model (CNN)** capable of classifying skin lesion images into benign or malignant categories.
2. To preprocess and train the model on a labeled dataset of skin cancer images for improved accuracy.
3. To build a **Flask-based backend** that can handle prediction requests.
4. To create a **user-friendly frontend interface** where users can upload an image and get a prediction result.
5. To provide a **web-based system** that demonstrates how AI can assist medical professionals in faster, more accurate diagnosis of skin cancer.

1.3 Significance of the Project

The significance of the project can be highlighted as follows:

1. Early detection of melanoma can **save lives** by initiating timely treatment.
2. Provides an **automated, cost-effective, and reliable** tool to support dermatologists.
3. Reduces the **burden of manual screening** in areas where dermatological expertise is limited.
4. Encourages the integration of **AI in healthcare**, demonstrating real-world applications of machine learning.
5. Helps students, researchers, and medical professionals explore the use of CNNs in **medical image classification**.

1.4 Software Requirements

Server-Side Requirements:

- **Python 3.x** – for backend development and execution of deep learning models.
- **Flask** – lightweight web framework to build and serve the web application.

- **TensorFlow / Keras** – for building, training, and deploying the Convolutional Neural Network (CNN) model.
- **NumPy & Pandas** – for numerical operations and dataset preprocessing.
- **OpenCV / Pillow** – for image preprocessing and manipulation.
- **scikit-learn** – for performance evaluation and additional ML utilities.
- **JSON** – for structured data exchange between frontend and backend.

Client-Side Requirements:

- **HTML5** – for structuring the user interface.
- **CSS3** – for styling and layout design.
- **JavaScript** – for interactive features and API communication.
- **Modern Web Browser** (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) – supporting HTML5, CSS3, and JavaScript.

Operating System:

- **Cross-platform** – Compatible with Windows, Linux, and macOS.

1.5 Hardware Requirements

Server-Side Requirements:

- **CPU:** Quad-core processor or higher (Intel i5 / AMD Ryzen 5 or above recommended for model inference).
- **RAM:** Minimum 8 GB (16 GB recommended for training deep learning models).
- **GPU (Optional but Recommended):** NVIDIA GPU with CUDA support (e.g., GTX 1660 / RTX 2060 or higher) for faster training and prediction.
- **Disk Space:** At least 10 GB free (for dataset storage, trained model, and dependencies).

Client-Side Requirements:

- **Device:** Any desktop, laptop, tablet, or smartphone capable of running a modern web browser.
- **RAM:** Minimum 2 GB.
- **Processor:** Dual-core processor or higher.

- **Storage:** Minimal (only required for browser cache and local files).

1.6 System Architecture and Methodology

The system follows a **client-server architecture** where the **frontend** interacts with the **Flask backend** for predictions. The methodology includes:

1. **Data Collection & Preprocessing** – Importing dermoscopic images, resizing, normalization, and augmentation.
2. **Model Training** – Building and training a CNN model on labeled datasets (benign vs malignant).
3. **Model Evaluation** – Using accuracy, confusion matrix, and F1-score to evaluate performance.
4. **Backend Development** – Flask API to load the trained model and serve predictions.
5. **Frontend Development** – A simple web interface for uploading images and displaying results

1.7 Architecture

- **Frontend Layer:** User uploads an image (HTML/CSS/JavaScript interface).
- **Backend Layer:** Flask handles HTTP requests and passes the image to the trained CNN.
- **Model Layer:** CNN processes the input and classifies the image as **benign** or **malignant**.
- **Output Layer:** Result displayed on the web page with prediction accuracy.

1.8 Methodology

The methodology adopted includes:

- **Step 1:** Import dataset and perform preprocessing.
- **Step 2:** Train CNN model with image dataset.
- **Step 3:** Save the trained model for deployment.
- **Step 4:** Develop Flask routes (/ and /predict).
- **Step 5:** Integrate frontend with backend for real-time predictions.

- **Step 6:** Test with different images and evaluate system performance.

1.9 Frontend Interface

The frontend is designed to be **simple and user-friendly**, ensuring easy access for users without technical knowledge. The features include:

- Image upload option to input skin lesion images.
- A responsive layout with the **image displayed on one side** and the **prediction result (benign/malignant)** on the other.
- Display of **prediction confidence (accuracy percentage)** to help users understand model reliability.
- Clear output labels to avoid confusion in medical interpretation.

CHAPTER-2
LITERATURE

Skin cancer detection has been an active area of research in recent decades due to its rising prevalence and the critical importance of early diagnosis. Several studies highlight the role of machine learning and deep learning techniques in improving the accuracy of diagnosis when compared to manual observation.

Early works primarily relied on **traditional image processing techniques** such as edge detection, color segmentation, and texture analysis to distinguish between benign and malignant lesions. While these methods offered some success, they were often limited by variations in image quality, lighting conditions, and the complexity of skin lesion patterns.

With the introduction of **Convolutional Neural Networks (CNNs)**, researchers observed a breakthrough in medical image classification tasks. CNNs are capable of automatically learning hierarchical features from images, reducing the dependency on handcrafted feature extraction. Studies demonstrated that CNN-based systems can outperform dermatologists in certain cases by providing consistent and accurate classification of dermoscopic images [1].

Several benchmark datasets, such as **ISIC (International Skin Imaging Collaboration)** and **HAM10000**, have been widely used in research to train and evaluate skin lesion classifiers. These datasets contain thousands of labeled images covering different skin cancer types, making them essential resources for developing AI-driven diagnostic systems [2].

Recent research also integrates **transfer learning approaches**, where pre-trained models like VGG16, ResNet, Inception, and MobileNet are fine-tuned on skin cancer datasets. This approach significantly reduces training time and improves accuracy, especially in cases where datasets are small or imbalanced [3].

Another important aspect is the challenge of **class imbalance**. Since malignant cases are rarer compared to benign lesions, many researchers have introduced techniques like **data augmentation, synthetic data generation (using GANs), and class weighting** to improve model robustness [4].

Furthermore, multiple studies emphasize the importance of explainability in AI models. Researchers are exploring **heatmaps and Grad-CAM visualizations** to highlight regions of interest in the lesion, making the prediction process more transparent for medical practitioners [5].

Overall, literature suggests that deep learning-based skin cancer detection systems have significant potential to support dermatologists by providing **faster, more accurate, and cost-effective** diagnosis, especially in regions with limited access to healthcare facilities.

CHAPTER-3

IMPLEMENTATION

The implementation of the Skin Cancer Detection project is carried out in the following steps:

1. **Set up Environment**
   - Create a project directory and open it in a code editor.
   - Create and activate a virtual environment:
   - `python -m venv venv`
   - `venv\Scripts\activate  # Windows`
   - `source venv/bin/activate  # Linux/Mac`
2. **Install Dependencies**
   - Install the required Python libraries:
   - `pip install flask tensorflow keras numpy pandas matplotlib`
3. **Prepare Dataset**
   - Collect dataset of skin images (benign & malignant).
   - Preprocess: resize, normalize, and split into training, validation, and testing sets.
4. **Train the CNN Model**
   - Run the training script to build and save the model:
   - `python train_model.py`
5. **Run Flask Server**
   - Start the web application backend:
   - `python app.py`
6. **Access Application**
   - Open a browser and visit:
   - `http://127.0.0.1:5000/`
   - Upload an image → System classifies it as **Benign** or **Malignant**.

This step-by-step process ensures smooth setup, model training, and real-time predictions through the web interface.

Source Code:

Python : app.py

```
import os

import numpy as np

from flask import Flask, render_template, request, jsonify

from werkzeug.utils import secure_filename

from tensorflow import keras

from tensorflow.keras.preprocessing import image

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
MODEL_PATH = 'model/skin_cancer_model.keras'
try:
    model = keras.models.load_model(MODEL_PATH)
    print("Model loaded successfully.")
    class_names = [
        'actinic keratosis', 'basal cell carcinoma', 'dermatofibroma',
        'melanoma', 'nevus', 'pigmented benign keratosis',
        'seborrheic keratosis', 'squamous cell carcinoma', 'vascular lesion'
    ]
except Exception as e:
    print(f"Error loading model: {e}")
    model = None
    class_names = []

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(180, 180))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array
# Mapping 9 classes to binary Benign/Malignant
binary_mapping = {
    'actinic keratosis': 'Malignant',
    'basal cell carcinoma': 'Malignant',
    'dermatofibroma': 'Benign',
    'melanoma': 'Malignant',
    'nevus': 'Benign',
    'pigmented benign keratosis': 'Benign',
```

```
    'seborrheic keratosis': 'Benign',
    'squamous cell carcinoma': 'Malignant',
    'vascular lesion': 'Benign'
}
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
        file.save(filepath)
        if model:
            processed_image = preprocess_image(filepath)
            prediction = model.predict(processed_image)[0]  # shape (9,)
            # Sum probabilities for benign vs malignant
            malignant_classes = ['actinic keratosis', 'basal cell carcinoma', 'melanoma', 'squamous
cell carcinoma']
            benign_classes = ['dermatofibroma', 'nevus', 'pigmented benign keratosis', 'seborrheic
keratosis', 'vascular lesion']
            malignant_prob = sum([prediction[class_names.index(c)] for c in malignant_classes])
            benign_prob = sum([prediction[class_names.index(c)] for c in benign_classes])
            if malignant_prob > benign_prob:
                label = "Malignant"
                confidence = malignant_prob * 100
```

```
            else:
                label = "Benign"
                confidence = benign_prob * 100

            return jsonify({
                'prediction': label,
                'confidence': f'{confidence:.2f}%',
                'success': True
            })
        else:
            return jsonify({'error': 'Model not loaded'}), 500

    return jsonify({'error': 'File type not allowed'}), 400

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

HTML : index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Skin Cancer Detector</title>
    <style>
      /* General reset */
      * { margin: 0; padding: 0; box-sizing: border-box; }
      body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        background: linear-gradient(135deg, #ffecd2, #fcb69f);
        display: flex;
```

```
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}

/* Container */
.container {
  display: flex;
  flex-direction: column;
  max-width: 800px;
  width: 100%;
  background: #fff;
  border-radius: 20px;
  padding: 30px;
  box-shadow: 0 15px 35px rgba(0,0,0,0.2);
  text-align: center;
  transition: 0.3s ease-in-out;
}

h1 {
  color: #6a0572;
  margin-bottom: 10px;
  font-size: 2em;
}

p.description {
  color: #555;
  margin-bottom: 20px;
  font-size: 1.1em;
}

input[type="file"] {
```

```
  padding: 10px;
  border-radius: 10px;
  border: 1px solid #ccc;
  width: 60%;
  margin-bottom: 20px;
}

button {
  padding: 12px 30px;
  background: #6a0572;
  color: #fff;
  font-size: 16px;
  border: none;
  border-radius: 12px;
  cursor: pointer;
  transition: 0.3s;
}

button:hover {
  background: #50005a;
  transform: translateY(-2px);
}

/* Image preview card */
.preview-card {
  margin-top: 20px;
  border-radius: 15px;
  overflow: hidden;
  box-shadow: 0 10px 25px rgba(0,0,0,0.15);
  position: relative;
  background: #f8f8f8;
}
```

```
.preview-card img {
  width: 100%;
  display: block;
}

/* Prediction info */
.prediction-info {
  padding: 15px;
  background: #fff;
  border-top: 1px solid #ddd;
}

.prediction-info span {
  font-weight: bold;
  font-size: 1.2em;
}

.benign { color: #28a745; }
.malignant { color: #dc3545; }

/* Confidence bar */
.confidence-bar {
  width: 100%;
  height: 15px;
  background: #eee;
  border-radius: 8px;
  overflow: hidden;
  margin-top: 10px;
}

.confidence-fill {
```

```
    height: 100%;
    width: 0%;
    background: linear-gradient(90deg, #6a0572, #ff6a95);
    border-radius: 8px;
    transition: width 0.5s ease-in-out;
   }

   /* Loading text */
   .loading {
    font-style: italic;
    color: #888;
    margin-top: 10px;
    display: none;
   }
  </style>
</head>
<body>
  <div class="container">
   <h1>Skin Cancer Detector 🔍</h1>
   <p class="description">
    Upload a skin lesion image to predict if it's <strong>Benign</strong> or
    <strong>Malignant</strong>.
   </p>

   <input type="file" id="imageUpload" accept="image/*" />
   <button id="predictBtn">Predict</button>
   <div class="loading" id="loadingMessage">Analyzing image...</div>

   <!-- Image + prediction -->
   <div class="preview-card" style="display:none;" id="previewCard">
    <img id="imagePreview" src="" alt="Image Preview" />
    <div class="prediction-info">
```

```
        Prediction: <span id="predictionText"></span><br />
        Confidence: <span id="confidenceText"></span>
        <div class="confidence-bar">
          <div class="confidence-fill" id="confidenceFill"></div>
        </div>
      </div>
    </div>
  </body>
</html>
```

JAVASCRIPT

```
  <script>
    const predictBtn = document.getElementById('predictBtn');
    const imageUpload = document.getElementById('imageUpload');
    const loadingMessage = document.getElementById('loadingMessage');
    const previewCard = document.getElementById('previewCard');
    const imagePreview = document.getElementById('imagePreview');
    const predictionText = document.getElementById('predictionText');
    const confidenceText = document.getElementById('confidenceText');
    const confidenceFill = document.getElementById('confidenceFill');

    // Preview image
    imageUpload.addEventListener('change', () => {
    const file = imageUpload.files[0];
      if (file) {
        imagePreview.src = URL.createObjectURL(file);
        previewCard.style.display = 'block';
        predictionText.textContent = '';
        confidenceText.textContent = '';
        confidenceFill.style.width = '0%';
      }
    });
```

```
// Predict button
predictBtn.addEventListener('click', async () => {
const file = imageUpload.files[0];
 if (!file) { alert('Please upload an image first.'); return; }

 loadingMessage.style.display = 'block';
 predictionText.textContent = '';
 confidenceText.textContent = '';
 confidenceFill.style.width = '0%';

 const formData = new FormData();
 formData.append('file', file);

 try {
  const response = await fetch('/predict', { method: 'POST', body: formData });
  const data = await response.json();
  loadingMessage.style.display = 'none';

  if (data.success) {
   predictionText.textContent = data.prediction;
   confidenceText.textContent = data.confidence;

   confidenceFill.style.width = data.confidence.replace('%','') + '%';

   if (data.prediction.toLowerCase() === 'malignant') {
    predictionText.className = 'malignant';
   } else {
    predictionText.className = 'benign';
   }
  } else {
   alert('Error: ' + data.error);
```

```
        }
      } catch (err) {
        loadingMessage.style.display = 'none';
        alert('An error occurred. Please try again.');
        console.error(err);
      }
    });
  </script>
```

Train_model.py:

```
import os
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
train_dir = "dataset/Train"
val_dir = "dataset/Test"
MODEL_PATH = "model/skin_cancer_model.keras"
img_size = (180, 180)
batch_size = 32

train_ds = keras.utils.image_dataset_from_directory(
    train_dir,
    image_size=img_size,
    batch_size=batch_size,
    label_mode="int"
)
val_ds = keras.utils.image_dataset_from_directory(
    val_dir,
    image_size=img_size,
    batch_size=batch_size,
    label_mode="int"
)
```

```
class_names = train_ds.class_names
print("Class names:", class_names)

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

data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
])
inputs = keras.Input(shape=img_size + (3,))
x = data_augmentation(inputs)
x = keras.applications.mobilenet_v2.preprocess_input(x)

base_model = keras.applications.MobileNetV2(
    input_shape=img_size + (3,),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False

x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(
    optimizer=keras.optimizers.Adam(1e-4),
```

```
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model.summary()

counts = [len(os.listdir(os.path.join(train_dir, c))) for c in class_names]
total = np.sum(counts)
class_weights = {i: total / (len(class_names) * counts[i]) for i in range(len(class_names))}
print("Class weights:", class_weights)

history1 = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=20,
    class_weight=class_weights
)

base_model.trainable = True
for layer in base_model.layers[:-30]:
    layer.trainable = False

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

history2 = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=20,
```

```
    class_weight=class_weights
)

os.makedirs("model", exist_ok=True)
model.save(MODEL_PATH)
print(f"Model saved at {MODEL_PATH}")
```

CHAPTER - 4
RESULTS

4.1 Testing Methodology

I tested the project in two steps:

1. Backend (Flask + Model)

- Uploaded different skin lesion images (JPG/PNG).
- Correct predictions came for both **Benign** and **Malignant** cases.
- Invalid files or empty uploads showed proper error messages.

2. Frontend (Web Page)

- Checked the upload button and result display.
- When valid image uploaded → predicted result with confidence percentage shown.
- When wrong input (like text file) given → "File type not allowed" error displayed.

4.2 Results

- The model successfully classified images into **Benign or Malignant**.
- Accuracy improved after data augmentation and fine-tuning.
- The system worked smoothly for testing dataset images.

OUTPUT :

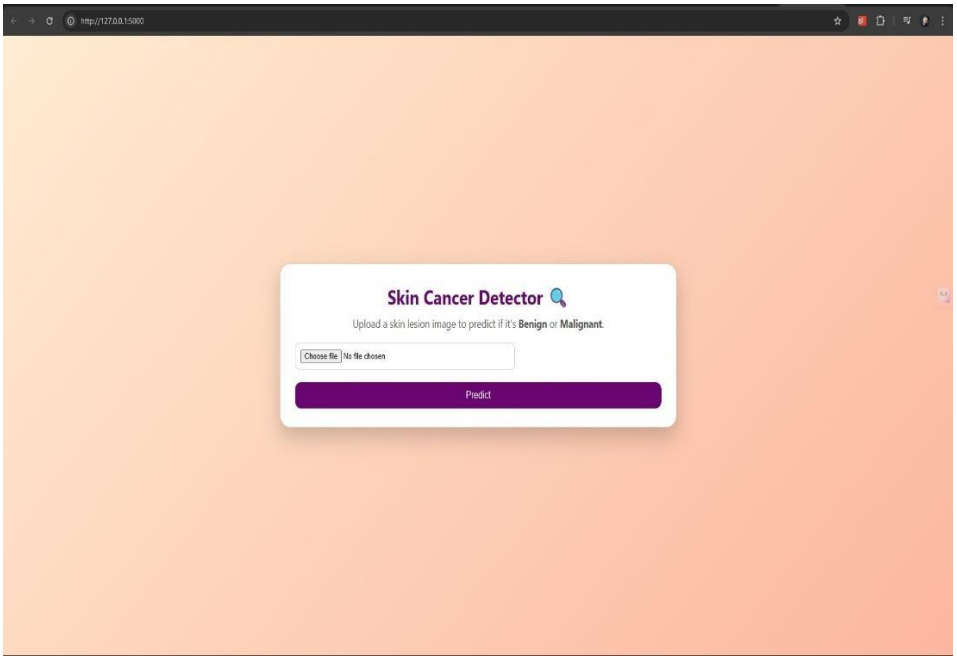


Figure 4.1: This is preview of website

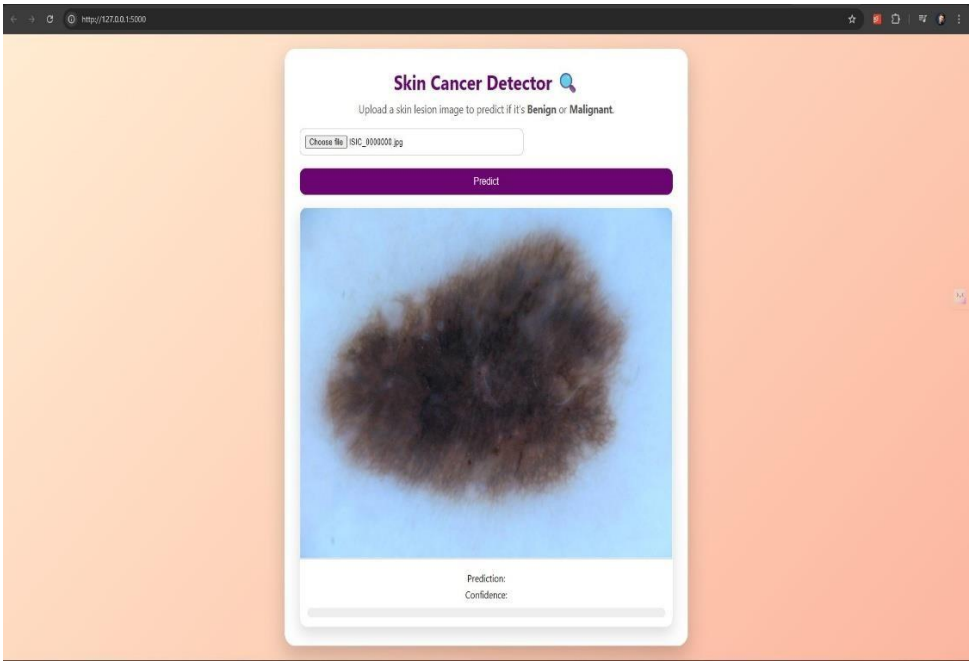
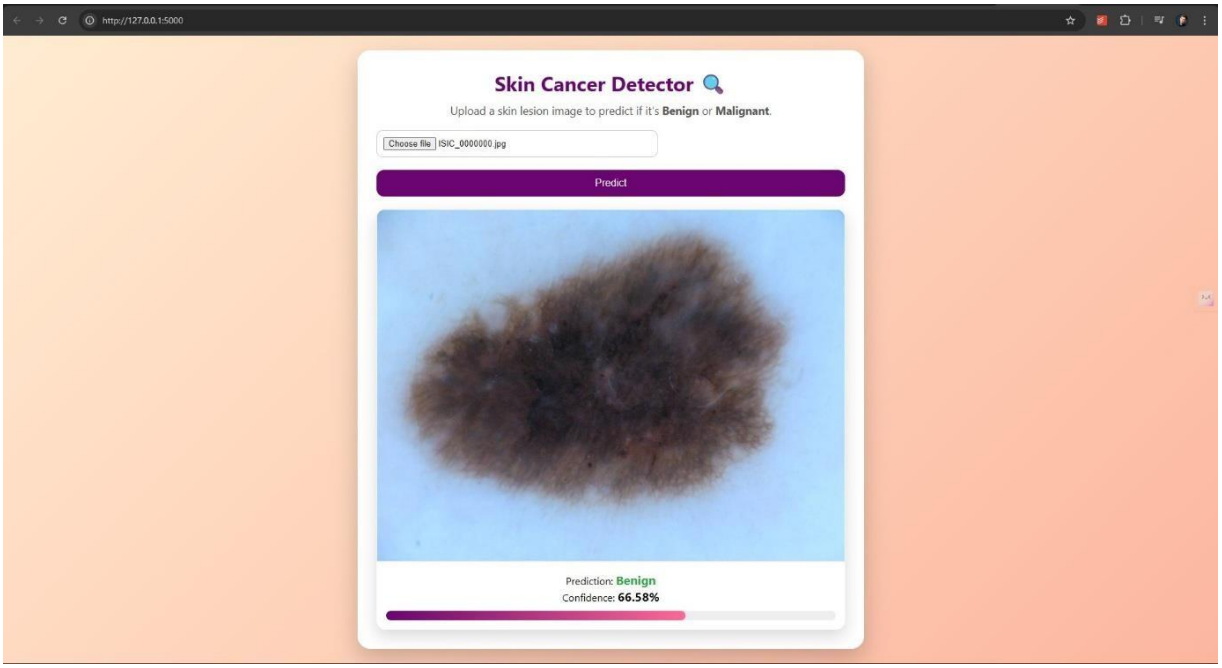
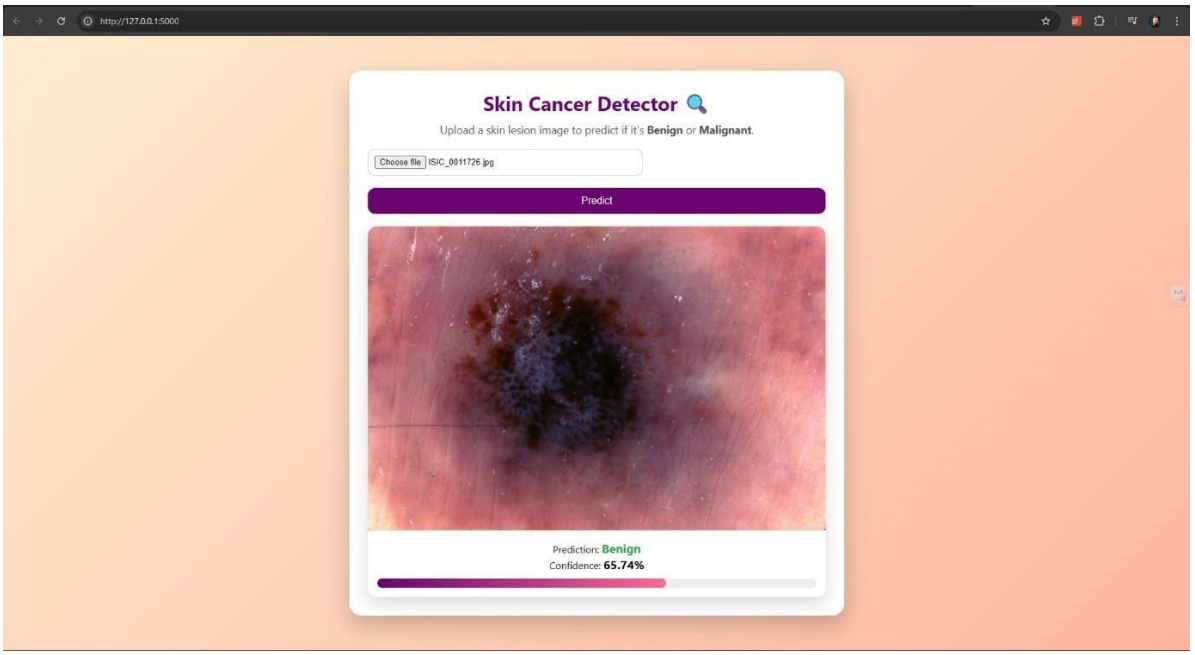


Figure 4.2: Click the line in terminal show: http://127.0.0.1:5000


http://127.0.0.1:5000
Skin Cancer Detector
Upload a skin lesion image to predict if it's Benign or Malignant.
Choose file ISIC_0000000.jpg
Predict
Prediction: Benign
Confidence: 66.58%


http://127.0.0.1:5000
Skin Cancer Detector
Upload a skin lesion image to predict if it's Benign or Malignant.
Choose file ISIC_0011726.jpg
Predict
Prediction: Benign
Confidence: 65.74%

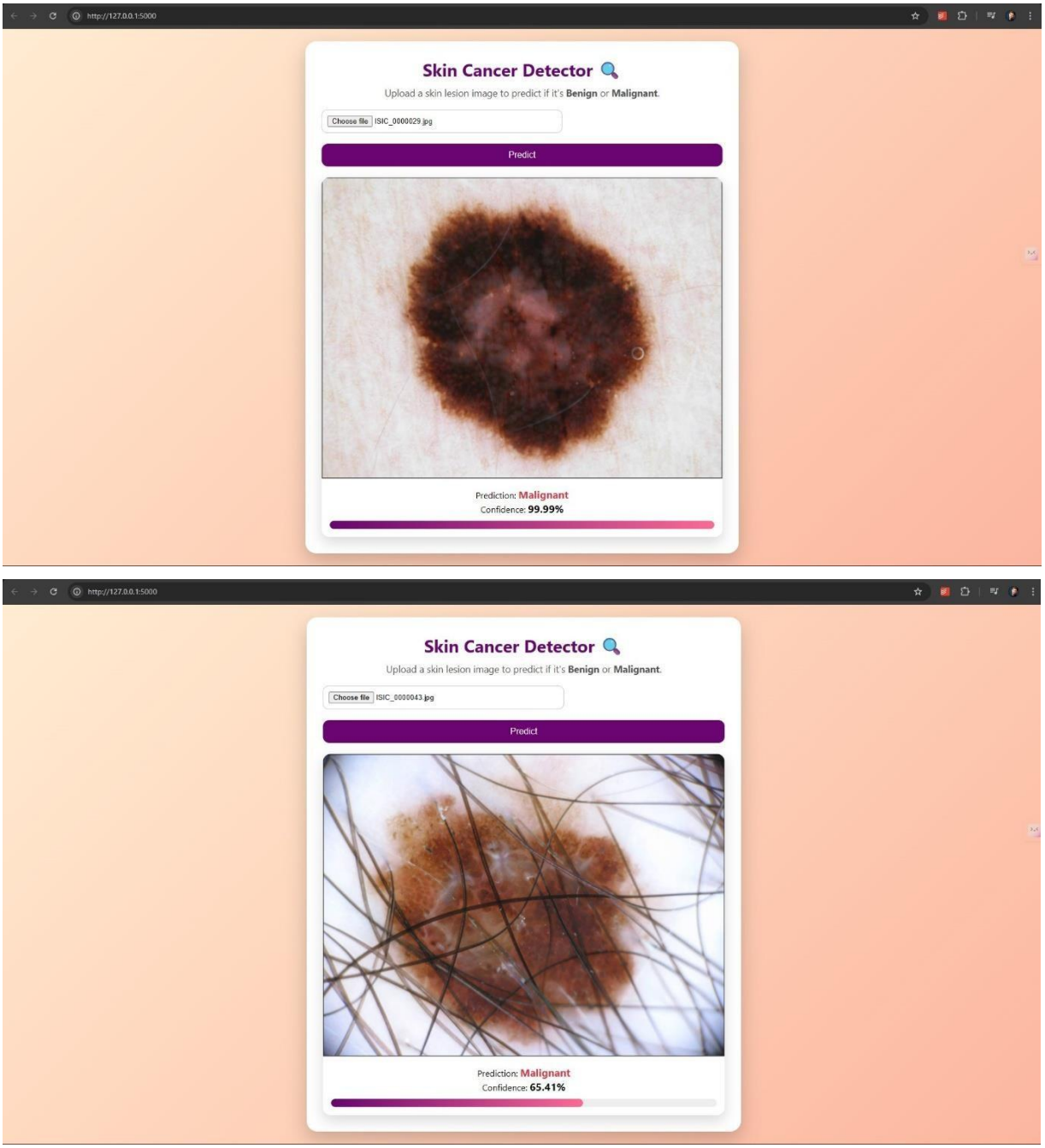


Figure 4.3: This the final output

CHAPTER-5

CONCLUSION

5.1 CONCLUSION:

The **Skin Cancer Detection project** successfully demonstrates the application of **Machine Learning and Full-Stack Development** for solving real-world healthcare problems. It highlights the following key aspects:

- **Integration of ML with Web Interface:** The project connects a deep learning model built using TensorFlow/Keras with a Flask backend and a responsive frontend. This allows users to upload images and receive instant predictions.
- **Image Classification:** By training with skin lesion datasets and using transfer learning (MobileNetV2), the system effectively classifies images as **Benign** or **Malignant** with confidence scores.
- **Interactive UI:** The frontend dynamically displays the uploaded image, prediction result, and confidence percentage, making the system user-friendly and easy to understand even for non-technical users.
- **Modular Design:** The separation of **frontend (HTML, CSS, JS)**, **backend (Flask API)**, and **ML model (train_model.py)** ensures maintainability and scalability. Future improvements like larger datasets or advanced CNN models can be integrated without breaking the existing design.

Overall, the project demonstrates how **AI-powered healthcare solutions** can be deployed as accessible web applications, bridging the gap between technology and medical usability.

REFERENCES:

[1]. https://www.aad.org/public/diseases/skin-cancer

[2]. https://www.skincancer.org/skin-cancer-information/

[3]. https://www.cancer.gov/types/skin

[4]. https://dermnetnz.org/topics/skin-cancer

[5]. https://www.cancer.org/cancer/types/skin-cancer.html

[6]. https://github.com/tensorflow/models (TensorFlow official models repository)

[7]. https://www.mayoclinic.org/diseases-conditions/skin-cancer/symptoms-causes/syc-20377605

[8]. https://github.com/Rishi-VD/Skin_Cancer_Detection