Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the Mini project titled **“Metamorphic Vision ”** being submitted by Abhinav Burnapelly, Harsha Nagudu bearing 1602-21-733-001,1602-21-733-015 in partial fulfilment of the requirements of the V semester, Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

S. Vinay Kumar,	**Dr. T.Adilakshmi,**
Assistant Professor,	**Professor&HOD,**
Dept. of CSE,	**Dept. of CSE.**
(Faculty I/c)	

(External Examiner)

Acknowledgement:

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Mr. R Sateesh Kumar**, Department of Computer Science and Engineering. Whose guidance was unforgettable and innovative ideas as well as his constructive suggestions has made the presentation of my thesis a grand success.

We are thankful to **Dr. T. Adilakshmi**, Head of Department (CSE), **Vasavi College of Engineering** for their help during our course work.

Finally at last but not least express our heart full thanks to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete my seminar work successively.

We convey our heartful thanks to our external guide Kalpana mam who has helped us throughout our project work.

TABLE OF CONTENTS

1.Introduction:

1.1 Introduction:

The "Metamorphic Vision Project" represents a cutting-edge endeavor in the field of image analysis, aiming to redefine how we ascertain the authenticity of visual content. Leveraging advanced metamorphic vision techniques and state-of-the-art algorithms, this project introduces a transformative approach to detecting potential alterations or manipulations in images. With a foundation in advanced image processing technologies and the integration of sophisticated machine learning models, the Metamorphic Vision Project stands as a promising initiative set to elevate the standards of image verification, offering users a reliable tool for distinguishing between genuine and morphed images. The primary goal of our project is to Correctly predict if an image is morphed image or not.

1.2 Motivation:

1.Addressing Digital Misinformation

2.Technological Innovation for Image Analysis

3.Building a Shield Against Malicious Uses

4. Empowering Users with a Reliable Tool

1.3 Problem Defination:

The "Metamorphic Vision" project addresses the critical challenge of detecting manipulated or morphed images in the digital landscape. In an era where image manipulation technologies pose a significant threat to digital authenticity, the project aims to develop sophisticated algorithms and leverage deep learning to accurately identify alterations in visual content. The primary goal is to provide users with a reliable tool for discerning between authentic and manipulated images, contributing to the mitigation of misinformation and ensuring the integrity of digital media.

2.Literature Survey:

1. "Digital Image Forensics: A Brief Overview" - Fridrich et al. (2009)

This foundational paper provides insights into the challenges and techniques in digital image forensics. It covers various methods for detecting image manipulations, laying the groundwork for understanding the landscape of image verification.

2. "Deep Learning for Image Tampering Detection: A Comprehensive Review" - Cozzolino et al. (2018)

- Focusing on the intersection of deep learning and image tampering detection, this comprehensive review explores the advancements in deep neural networks for identifying manipulated images. The paper discusses various architectures and strategies employed for enhancing detection accuracy.

3. "Metamorphic Testing: A Review of Challenges and Opportunities" - Chen et al. (2018)

- While not directly focused on image verification, this paper introduces the concept of metamorphic testing, which involves applying transformations to inputs and analyzing the output changes. The principles of metamorphic testing could be adapted to the image verification domain.

4. "Deepfake Detection: A Review" - Li and Lyu (2019)

- Delving into the specifics of deepfake detection, this review examines the evolution of deepfake technology and the corresponding advancements in detection methods. Understanding the current state-of-the-art in deepfake detection is crucial for the "Metamorphic Vision" project.

5. "Applications of Deep Learning in Image Authentication: A Comprehensive Review" - Paudel et al. (2020)

- Focusing on the broader applications of deep learning in image authentication, this review outlines the diverse methodologies and technologies used for ensuring the authenticity of digital images. It provides valuable insights into the potential integration of deep learning techniques in the proposed project.

6. "Recent Advances in Image Tampering Detection: A Comprehensive Survey" - Al-Qershi et al. (2021)

- Offering a contemporary perspective, this survey provides an overview of recent advances in image tampering detection techniques. It explores various methodologies, including traditional methods and those leveraging deep learning, providing insights for the "Metamorphic Vision" project.

These selected works form a foundational understanding of the challenges, methodologies, and advancements in the domain of image verification. The survey spans traditional and deep learning-based approaches, offering a comprehensive view that can guide the development and innovation within the "Metamorphic Vision" project.

3.System design

1. User Authentication:

- Users will be required to sign in or create an account to access the system. This ensures secure access and allows for personalized features.

2. Dashboard:

- Upon login, users are greeted with a dashboard displaying relevant information, such as recent analysis history, authentication confidence levels, and any system updates.

3. Image Upload:

- The primary interface allows users to easily upload images for verification. It supports various image formats and provides clear instructions for the upload process.

4. Image Analysis Section:

- Once an image is uploaded, users can view the progress of the analysis. The system provides real-time updates on the analysis status, indicating whether the image is authentic or potentially manipulated.

5. Detailed Analysis Report:

- Users can access a detailed analysis report outlining specific areas or features within the image that indicate potential morphing. This section provides transparency into the system's assessment.

6. Authentication Confidence Levels:

- The interface includes indicators of confidence levels associated with the authenticity determination. This allows users to gauge the reliability of the system's assessment.

7. User Preferences:

- Users can customize settings, such as notification preferences, analysis thresholds, or preferred output formats. This contributes to a personalized user experience.

8. User Feedback Mechanism:

- The system incorporates a user feedback mechanism, allowing users to provide insights on the analysis results. This feedback loop enhances user engagement and contributes to system improvement.

9. Security Measures:

- The interface emphasizes the security measures in place, reassuring users about the protection of their data. This may include SSL encryption, secure data storage practices, and adherence to privacy regulations.

10. Help and Support:

- A dedicated section for user assistance, FAQs, and contact information for customer support. Clear instructions and tooltips throughout the interface contribute to a user-friendly experience.

11. Responsiveness:

- The interface is designed to be responsive, ensuring accessibility across various devices and screen sizes.

12. System Status Updates:

- Users receive system status updates, ensuring they are aware of any maintenance or improvements that may impact their experience.

13. Integration of Advanced Features (Optional):

- For future scalability, consider integrating advanced features such as batch processing, integration with external APIs, or even AR-based visualization for a more interactive experience.

By focusing on a clean, intuitive, and transparent interface, the "Metamorphic Vision" project aims to provide users with a seamless and trustworthy experience in verifying the authenticity of digital images.

4.Implementation:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>File Upload</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f4;
    }

    .container {
      background-color: white;
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .custom-file-upload {
      display: block;
      width: 100%;
      margin: 15px 0;
      border: 2px dashed #ddd;
      border-radius: 5px;
      padding: 10px;
      text-align: center;
      cursor: pointer;
      color: #666;
    }
```

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

.btn-custom {
  background-color: #044275;
  color: white;
  border-radius: 5px;
  padding: 10px 20px;
  cursor: pointer;
  display: block;
  width: 100%;
  text-align: center;
  margin-top: 15px;
}

.btn-custom:hover {
  background-color: #0b314f;
}
.loader {
  border: 5px solid #f3f3f3;
  border-top: 5px solid #3498db;
  border-radius: 50%;
  width: 50px;
  height: 50px;
  animation: spin 2s linear infinite;
  display: none; /* Hide by default */
  margin: auto;
}

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

#previewImg {
  max-width: 100%;
  height: auto;
  display: none; /* Hide by default */
```

```
      margin-bottom: 10px;
    }
  </style>
</head>
<body>
  <div class="container">
    <form id="uploadForm">
      <label for="file" class="custom-file-upload">
        Upload Image
        <input type="file" name="file" id="file" onchange="previewFile()">
      </label>
      <button type="submit" class="btn-custom">Submit</button>
    </form>

    <div id="imagePreview">
      <img id="previewImg" alt="Image preview...">
      <div class="loader" id="loader"></div>
    </div>

    <form id="predictForm" action="{{ url_for('predict') }}" method="POST">
      <input type="submit" class="btn-custom" value="Predict">
    </form>

    <button id="displayBtn" class="btn-custom">Fetch Result</button>
    <div id="result"></div>
  </div>

  <script>
    function previewFile() {
      const preview = document.getElementById('previewImg');
      const file = document.getElementById('file').files[0];
      const reader = new FileReader();

      reader.addEventListener("load", function () {
        // Convert image file to base64 string
        preview.src = reader.result;
        preview.style.display = 'block';
      }, false);

      if (file) {
        reader.readAsDataURL(file);
      }
    }

    document.getElementById('uploadForm').addEventListener('submit', (e) => {
      e.preventDefault();
      const formData = new FormData();
      const fileInput = document.getElementById('file');
      formData.append('file', fileInput.files[0]);

      document.getElementById('loader').style.display = 'block'; // Show loading
symbol

      fetch('http://localhost:5000/ref', {
        method: 'POST',
        body: formData,
      })
      .then(res => res.json())
      .then(data => {
        console.log(data);
        document.getElementById('loader').style.display = 'none'; // Hide loading
symbol
      })
      .catch(error => {
        console.error('Error:', error);
        document.getElementById('loader').style.display = 'none'; // Hide loading
symbol on error
      });
    });
  </script>
</body>
</html>
```

Myjs.js

```
const express = require("express");
const multer = require('multer');
const cors = require('cors');
const path = require('path');
const fs = require('fs');
const { parse } = require("csv-parse");

const app = express();
app.use(cors()); // Allows incoming requests from any IP

const folderPath3 = path.join(__dirname, 'test/testing');

fs.readdir(folderPath3, (err, files) => {
  if (err) {
    console.error('Error reading folder:', err);
    res.status(500).json({ error: 'Error reading folder' });
    return;
  }

  files.forEach(file => {
    const filePath = path.join(folderPath3, file);
    fs.unlinkSync(filePath); // Delete each file in the folder
  });
});

const storage2 = multer.diskStorage({
    destination: function (req, file, callback) {
        callback(null, path.join(__dirname, 'test/testing'));
    },
    filename: function (req, file, callback) {
        callback(null, file.originalname);
    }
});

const upload2 = multer({ storage: storage2 });

app.post("/ref", upload2.single("file"), (req, res) => {

    console.log(req.body);
    console.log(req.file);
    console.log('good boy')

    res.json({ message: "File(s) uploaded successfully" });
});
```

```
//----------------------------------------------------------------------------------------------
--------------------

app.get("/result",(req,res)=>{

        // specify the path of the CSV file
        const path = "submission.csv";

        // Create a readstream
        // Parse options: delimiter and start from line 1
        const result=[];
        fs.createReadStream(path)
        .pipe(parse({ delimiter: ",", from_line: 1 }))
        .on("data", function (row) {
            // executed for each row of data
            console.log(row);
            result.push(row);
        })
        .on("error", function (error) {
            // Handle the errors
            console.log(error.message);
        })
        .on("end", function () {
            if(result.length>1){res.json({ result: result[1][1] });

            }
            console.log("File read successful");
        });
    });
//----------------------------------------------------------------------------------------------
--------------------
app.listen(5000, function(){
    console.log("Server running on port 5000");
});
```

Submission.csv

```
filename,label
Image,0.8991285800933838
```

Myapp.js

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from facenet_pytorch.models.inception_resnet_v1 import get_torch_home
torch_home = get_torch_home()
import os
import torch
import cv2
import time
import glob
from PIL import Image
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from tqdm.notebook import tqdm
from facenet_pytorch import MTCNN, InceptionResnetV1, extract_face
from tqdm import tqdm
from flask import Flask,render_template,redirect,url_for

app = Flask(__name__)

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

@app.route('/predict',methods=['POST'])
def predict():
    import numpy as np
    import matplotlib.pyplot as plt
    from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D,
MaxPooling2D, BatchNormalization, Dropout, Reshape, Concatenate,
LeakyReLU
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.models import Model

    # Height and width refer to the size of the image
    # Channels refers to the amount of color channels (red, green, blue)
```

```
image_dimensions = {'height':256, 'width':256, 'channels':3}
# Create a Classifier class

class Classifier:
    def __init__(self):
        self.model = 0

    def predict(self, x):
        return self.model.predict(x)

    def fit(self, x, y):
        return self.model.train_on_batch(x, y)

    def get_accuracy(self, x, y):
        return self.model.test_on_batch(x, y)

    def load(self, path):
        self.model.load_weights(path)
# Create a MesoNet class using the Classifier

class Meso4(Classifier):
    def __init__(self, learning_rate = 0.001):
        self.model = self.init_model()
        optimizer = Adam(lr = learning_rate)
        self.model.compile(optimizer = optimizer,
                        loss = 'mean_squared_error',
                        metrics = ['accuracy'])

    def init_model(self):
        x = Input(shape = (image_dimensions['height'],
                        image_dimensions['width'],
                        image_dimensions['channels']))

        x1 = Conv2D(8, (3, 3), padding='same', activation = 'relu')(x)
        x1 = BatchNormalization()(x1)
        x1 = MaxPooling2D(pool_size=(2, 2), padding='same')(x1)

        x2 = Conv2D(8, (5, 5), padding='same', activation = 'relu')(x1)
        x2 = BatchNormalization()(x2)
        x2 = MaxPooling2D(pool_size=(2, 2), padding='same')(x2)

        x3 = Conv2D(16, (5, 5), padding='same', activation = 'relu')(x2)
        x3 = BatchNormalization()(x3)
        x3 = MaxPooling2D(pool_size=(2, 2), padding='same')(x3)

        x4 = Conv2D(16, (5, 5), padding='same', activation = 'relu')(x3)
        x4 = BatchNormalization()(x4)
        x4 = MaxPooling2D(pool_size=(4, 4), padding='same')(x4)

        y = Flatten()(x4)
        y = Dropout(0.5)(y)
        y = Dense(16)(y)
        y = LeakyReLU(alpha=0.1)(y)
        y = Dropout(0.5)(y)
        y = Dense(1, activation = 'sigmoid')(y)

        return Model(inputs = x, outputs = y)

    # Instantiate a MesoNet model with pretrained weights
meso = Meso4()
meso.load('./weights/Meso4_DF')
# Prepare image data

# Rescaling pixel values (between 1 and 255) to a range between 0 and 1
dataGenerator = ImageDataGenerator(rescale=1./255)

# Instantiating generator to feed images through the network
# generator = dataGenerator.flow_from_directory(
#     './data/',
#     target_size=(256, 256),
#     batch_size=1,
#     class_mode='binary')
# # Recreating generator after removing '.ipynb_checkpoints'
# dataGenerator = ImageDataGenerator(rescale=1./255)

generator = dataGenerator.flow_from_directory(
    './test/',
    target_size=(256, 256),
    batch_size=1,
        class_mode='binary')

    # Re-checking class assignment after removing it
    generator.class_indices
    # Rendering image X with label y for MesoNet
    X, y = generator.next()

# Evaluating prediction
    a=(meso.predict(X)[0][0]-0.1)
    import pandas as pd

    # Step 1: Read the CSV file into a DataFrame
    csv_file_path = 'submission.csv'
    df = pd.read_csv(csv_file_path)

    # Step 2: Replace the existing row with new data
    new_row_data = {"filename":"Image",'label':a}
    df.iloc[0] = new_row_data # Assuming there's only one row in the
DataFrame

    # Step 3: Write the updated DataFrame back to the CSV file
    print(df)
    df.to_csv(csv_file_path, index=False)

    return render_template("index.html")

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

5.Results:

Fig 5.1 :HomePage

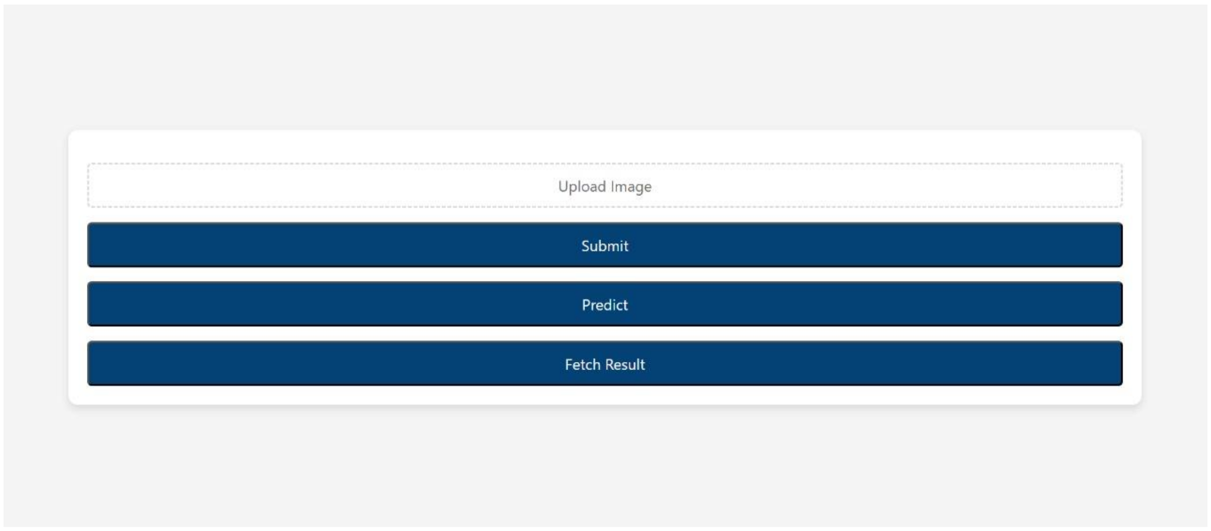


Fig 5.2 Displaying Image

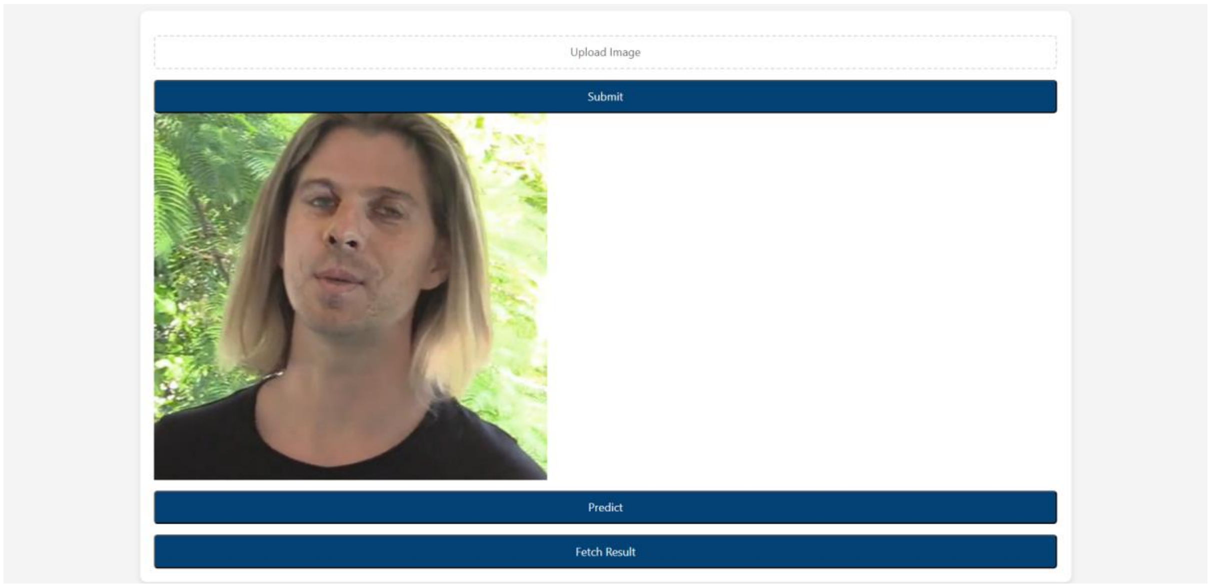
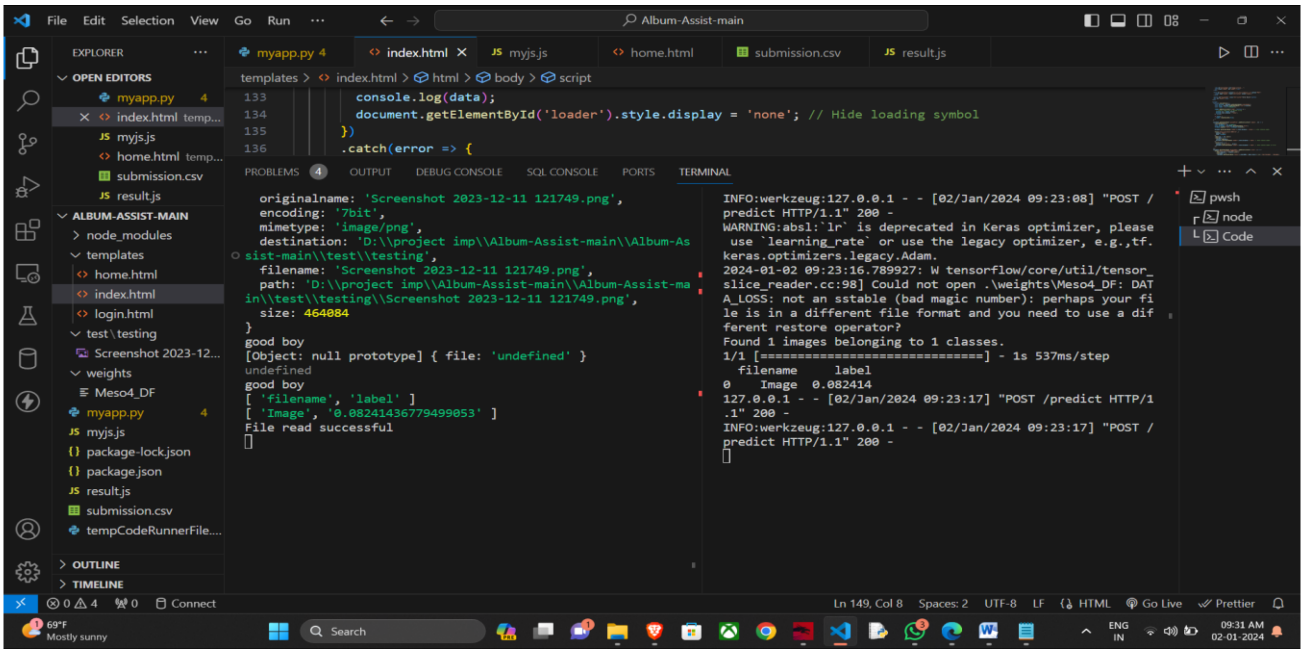


Fig 5.3

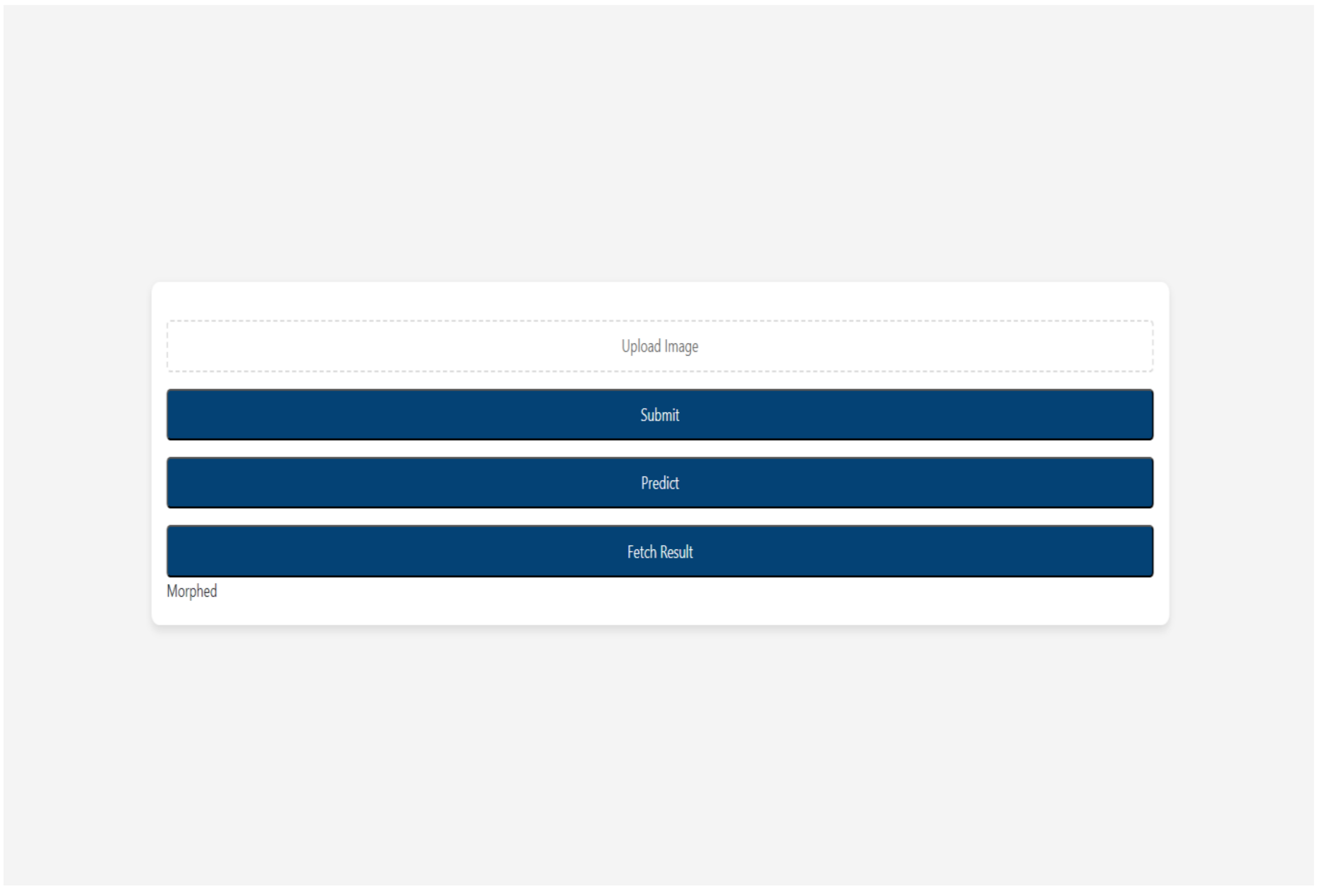


```
console.log(data);
document.getElementById('loader').style.display = 'none'; // Hide loading symbol
})
.catch(error => {
```

```
  originalname: 'Screenshot 2023-12-11 121749.png',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: 'D:\\project imp\\Album-Assist-main\\Album-As
sist-main\\test\\testing',
  filename: 'Screenshot 2023-12-11 121749.png',
  path: 'D:\\project imp\\Album-Assist-main\\Album-Assist-ma
in\\test\\testing\\Screenshot 2023-12-11 121749.png',
  size: 464084
}
good boy
[Object: null prototype] { file: 'undefined' }
undefined
good boy
[ 'filename', 'label' ]
[ 'Image', '0.08241436779499053' ]
File read successful
```

```
INFO:werkzeug:127.0.0.1 - - [02/Jan/2024 09:23:08] "POST /
predict HTTP/1.1" 200 -
WARNING:absl:`lr` is deprecated in Keras optimizer, please
 use `learning_rate` or use the legacy optimizer, e.g.,tf.
keras.optimizers.legacy.Adam.
2024-01-02 09:23:16.789927: W tensorflow/core/util/tensor_
slice_reader.cc:98] Could not open .\weights\Meso4_DF: DAT
A_LOSS: not an sstable (bad magic number): perhaps your fi
le is in a different file format and you need to use a dif
ferent restore operator?
Found 1 images belonging to 1 classes.
1/1 [==============================] - 1s 537ms/step
  filename     label
0    Image  0.082414
127.0.0.1 - - [02/Jan/2024 09:23:17] "POST /predict HTTP/1
.1" 200 -
INFO:werkzeug:127.0.0.1 - - [02/Jan/2024 09:23:17] "POST /
predict HTTP/1.1" 200 -
```

Fig 5.4 Output



6. Conclusion:

Looking ahead, the future scope of the "Metamorphic Vision" project is expansive, focusing on continuous improvement and adaptation to emerging trends:

FUTURE SCOPE

Enhanced Algorithmic Capabilities:

- The project will explore further enhancements to metamorphic vision algorithms, ensuring adaptability to evolving manipulation techniques and maintaining a high level of accuracy.

Integration of Advanced Deep Learning Models:

- Exploring the integration of advanced deep learning models to continually improve the system's ability to discern sophisticated morphing patterns.

Real-time Analysis Enhancements:

- The project aims to refine real-time image analysis, providing users with even quicker and more efficient results.

REFERENCES :

[1] "Express.js," Fast, unopinionated, minimalist web framework for Node.js. [Online]. Available: https://expressjs.com/

[2] "React.js," A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/

[3] "Node.js," Node.js. [Online]. Available: https://nodejs.org/

[4] "npm," npm | build amazing things. [Online]. Available: https://www.npmjs.com/

[5] OpenAI. (n.d.). GPT-3.5. [Online]. Available: https://beta.openai.com/signup/

[6] "Visual Studio Code," Visual Studio Code - Code Editing. Redefined. [Online]. Available: https://code.visualstudio.com/

[7] “Tensorflow” ,Tensorflow model for Deep learning model – https://tensorflow.org

[8] “Python Flask” Flask - https://flask.palletsprojects.com/en/3.0.x/