INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN KASHMERE GATE, DELHI-110006



**IT WORKSHOP-II**

**Subject : IT**

**Subject Code : MCA-110**

***SUBMITTED BY:*** ***SUBMITTED TO:***

**Aanchal Arora Ms. Urmila** **(00104092024) Signature: ………**

**Ishita Jain (02704092024)**

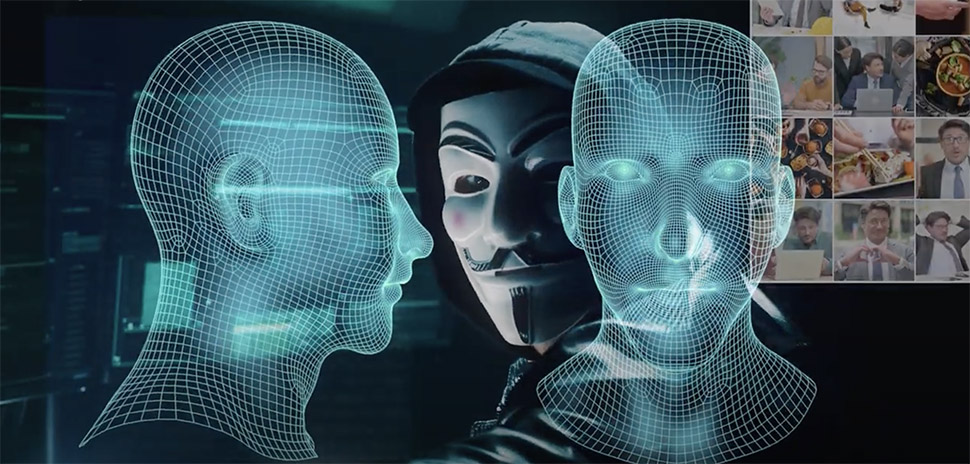
|  |  |  |
| --- | --- | --- |
| **S no** | **Topics** | **Page No** |
|  | Title of the project | 3 |
|  | Problem Definition | 4 |
|  | Introduction | 5 |
|  | Aim & Objective | 6 |
|  | Project Category | 7 |
|  | Tools & platform | 8 |
|  | DFD’S | 9 |
|  | DFD’S Design | 10-12 |
|  | Input to the system | 13 |
|  | Module Description | 14 |
|  | Output From The system | 15 |
|  | Testing/Security | 16-17 |
|  | Future Scope | 18 |
|  | Bibliography | 19 |

**INDEX**

2

**TITLE OF THE PROJECT**

*Deepfake Detection with Machine Learning*



"Your Digital Shield Against Deepfakes."

3

**PROBLEM STATEMENT**

In today’s digital world, fake videos, known as **deepfakes**, are becoming a serious threat. These AI-generated videos can make people appear to say or do things they never did. Deepfakes are often used to spread **false news, trick people into scams, damage reputations, or manipulate elections**. As technology improves, these fake videos look more real, making it difficult to tell what is true and what is fake.

**Real Incidents in India:**

* **Political Deepfakes (2024 Elections):** Deepfake videos were used in election campaigns to create fake speeches of famous politicians, misleading voters.
* **Celebrity Deepfakes:** Bollywood actor **Anil Kapoor** fought a legal case against the misuse of his image through AI-generated deepfakes.
* **Fraud Using Deepfakes:** A **man in Kerala** lost money after scammers used deepfake video calls to impersonate his friend and trick him.
* **Extortion Case:** A **senior citizen in Ghaziabad** was blackmailed using a deepfake video, causing him severe stress.

These incidents show how dangerous deepfakes can be and why we need a strong system to detect them.

**Our Solution – Deepfake Detection Using Machine Learning**

Our project aims to build **an AI-powered tool that can scan videos and detect whether they are real or fake.** By using advanced machine learning techniques, this system will help prevent fraud, misinformation, and privacy violations. It will be useful for news agencies, law enforcement, and social media platforms to **protect truth and stop the spread of fake content.**

4

**INTRODUCTION**

Our **Deepfake Detection System** is an AI-powered tool that checks if a video is *real or fake*. Deepfake videos use advanced technology to change faces, making it seem like someone is doing something they never actually did. These fake videos can spread false information and cause serious problems.

Our system works by ***analysing the video*.** It first ***breaks the video into frames* (images)** and then uses **Machine Learning (ML) models to study facial movements and patterns**. One of the key models used is ***GRU (Gated Recurrent Unit), which is a type of Recurrent Neural Network (RNN).*** GRU helps the system detect **unnatural changes like weird blinking, face distortions, and mismatched lighting**, which are common in deepfake videos.

After the analysis, the system provides a **detection result** showing whether the video is **real or fake**. This tool helps **news agencies, social media platforms, and law enforcement** detect deepfakes and **protect people from fake content and online fraud.**

5

**AIM & OBJECTIVE**

**Aim:**

The aim of this project is to develop an **AI-powered Deepfake Detection System** that can quickly and accurately identify fake videos that are difficult to detect with the human eye. This system will help prevent the **misuse of AI-generated deepfake videos** and protect users from cyber threats and misinformation.

**Objectives:**

1. **Fast and Efficient Detection:** The system can analyse **thousands of video frames within seconds** to detect deepfakes.
2. **AI-Powered Analysis:** Uses **Machine Learning models, including GRU (a type of RNN),** to study facial movements and detect unnatural patterns.
3. **User-Friendly Process:** The user simply **uploads a video**, and the system quickly **provides a result** stating whether the video is real or fake.
4. **Prevent Cyber Threats:** Helps in **stopping fake news, online fraud, and identity theft** caused by deepfake videos.
5. **Protects Digital Integrity:** Supports **law enforcement, media agencies, and social platforms** in ensuring authentic content.

6

**TOOLS AND PLATFORM**

***1. Programming Language***

- Python→ Core language for model implementation

***2. Frameworks & Libraries***

- OpenCV→ Video processing & frame extraction

- TensorFlow/Keras → Deep learning model (GRU-based sequence analysis)

- NumPy & Pandas → Data manipulation

- Matplotlib → Visualization of frames

- Scikit-learn → Train-test split & evaluation metrics

- JSON (Built-in Python module) → Parsing metadata from JSON files

***3. Model Development***

- InceptionV3 (Pre-trained CNN) → Feature extraction from frames

- GRU (Gated Recurrent Units) → Sequential model for analysing video frames

- Adam Optimizer & Binary Cross-Entropy Loss → Model training optimization

***4. Backend & API***

- Flask→ Web API for handling video input & returning predictions

***5. Data Storage***

- JSON Files → Storing metadata (video paths, labels, frame info)

- Local File System→ Storing training/test datasets

***6. Platform & Environment***

- Jupyter Notebook → Model development & testing

- Local System → Running the trained model

- OneDrive/Desktop Directory→ Managing datasets

7

**PROJECT CATEGORY**

The project category for your Deepfake Detection Model falls under:

🔹 Artificial Intelligence & Machine Learning

🔹 Computer Vision

🔹 Cybersecurity & Digital Forensics

🔹 Deep Learning

🔹 Multimedia Forensics

8

**DATA FLOW DIAGRAM**

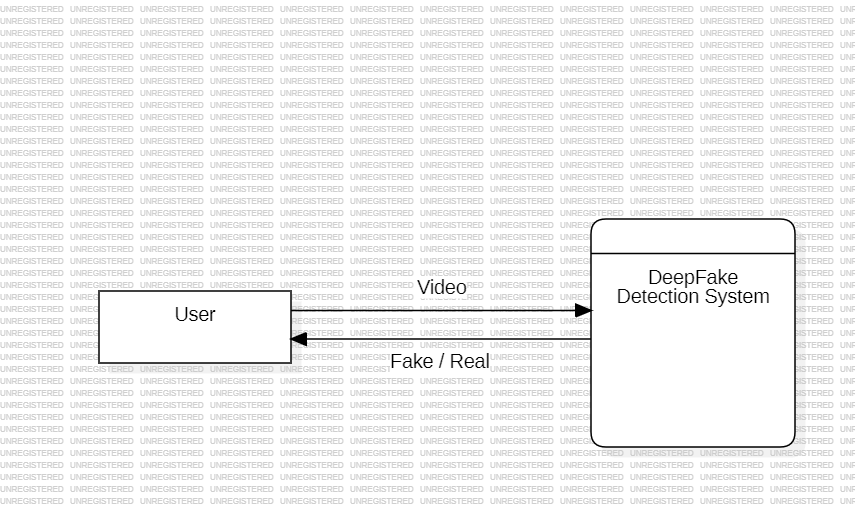
A data flow diagram (DFD) is a graphical representation of the "flow" of data through information, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD’S can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a [flowchart](http://en.wikipedia.org/wiki/Flowchart)).

9

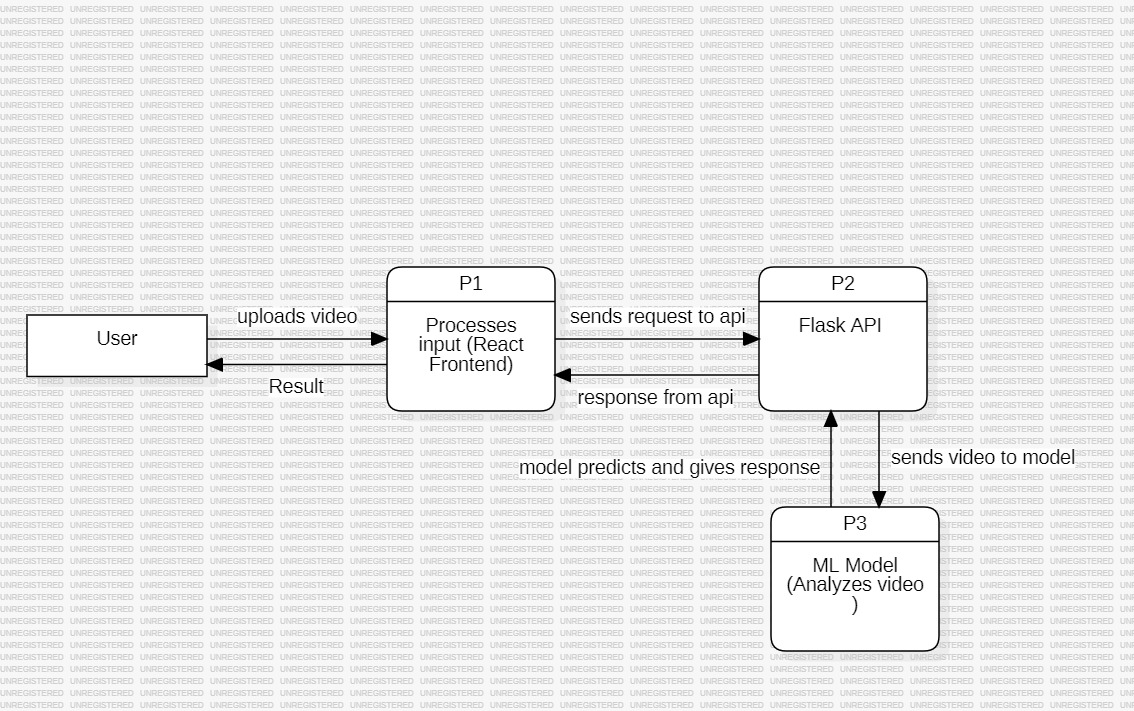
**DFD’S Design**

**0 Level DFD:**

****

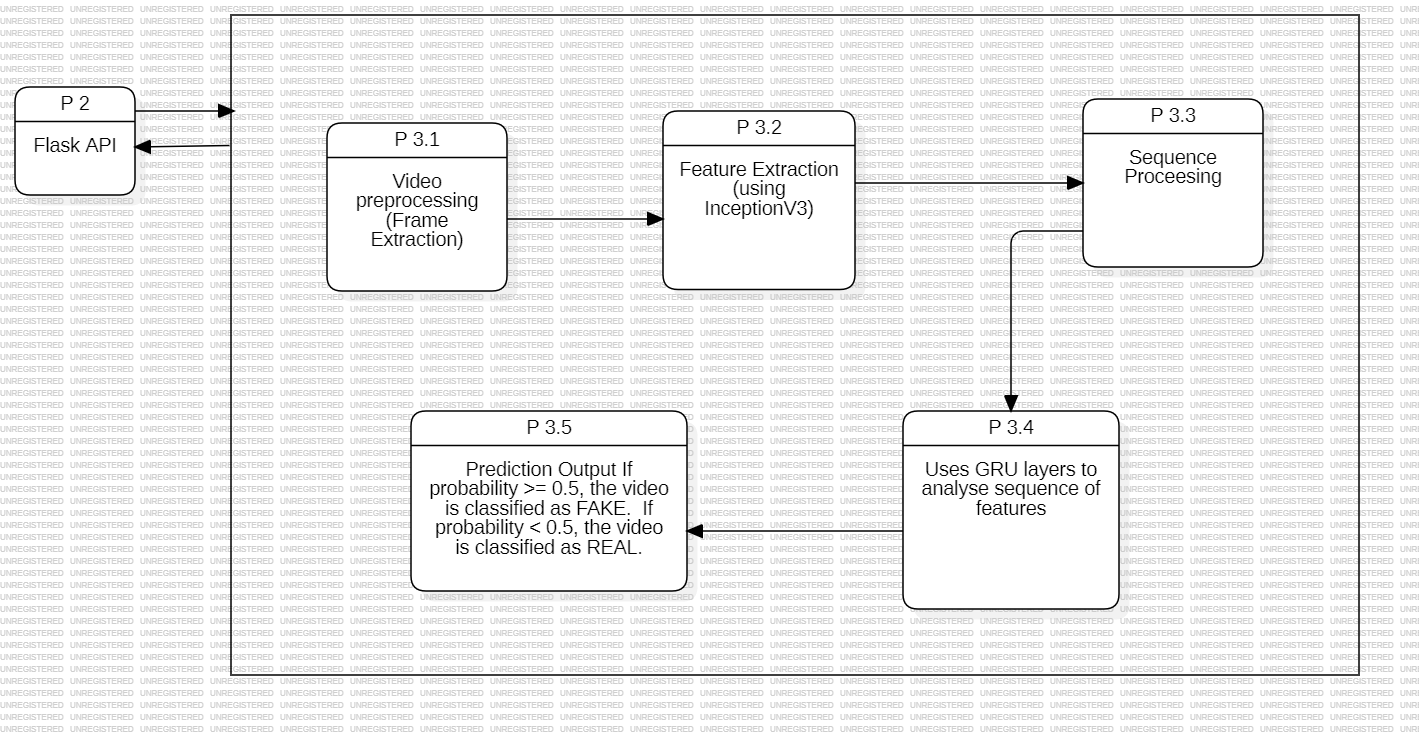
10

**User 1 Level DFD:**

****

11

**User 2 Level DFD:**

****

12

**Input to the system**

The input to the system is a video file uploaded by the user.

**File Format:** .mp4

13

**Module Description**

The **Deepfake Detection Module** is designed to analyze uploaded videos and determine whether they contain AI-generated (deepfake) content. This module provides an intuitive interface for users to upload a video, play it within the application, and receive a prediction based on a trained deepfake detection model.

**Key Features:**

* ***Video Upload****:* Users can upload a video file for analysis.
* ***Video Playback****:* The uploaded video can be played within the application to review the content.
* ***Deepfake Prediction***: The system processes the video using a deepfake detection model, which analyzes frames and provides a classification result (e.g., Real or Deepfake).
* ***User-friendly Interface***: A simple and interactive UI to facilitate easy navigation and operation.
* ***Performance Metrics****:* Optionally, the module can display confidence scores or heatmaps to highlight suspected manipulated areas.

14

**Output from the system**

**Prediction Result**

* **Label:** "REAL" or "FAKE"
* **Confidence Score:** Probability of prediction (e.g., 0.92 means 92% confidence)
* **Processing Time:** Time taken for analysis

15

**CODING**

**FLASK API CODE**

from flask import Flask, request, jsonify, render\_template

from werkzeug.utils import secure\_filename

import numpy as np

import tensorflow as tf

import cv2

import os

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

# Load the pre-trained model

model = tf.keras.models.load\_model('./model/deepfake\_video\_model.h5')

# Define constants

IMG\_SIZE = 224

MAX\_SEQ\_LENGTH = 20

NUM\_FEATURES = 2048

#Define the feature extractor (InceptionV3)

def build\_feature\_extractor():

feature\_extractor = tf.keras.applications.InceptionV3(

weights="imagenet",

include\_top=False,

pooling="avg",

input\_shape=(IMG\_SIZE, IMG\_SIZE, 3),

)

preprocess\_input = tf.keras.applications.inception\_v3.preprocess\_input

inputs = tf.keras.Input((IMG\_SIZE, IMG\_SIZE, 3))

preprocessed = preprocess\_input(inputs)

outputs = feature\_extractor(preprocessed)

return tf.keras.Model(inputs, outputs)

feature\_extractor = build\_feature\_extractor()

# Utility function to load and process video

def load\_video(path, max\_frames=0, resize=(IMG\_SIZE, IMG\_SIZE)):

cap = cv2.VideoCapture(path)

frames = []

try:

while True:

ret, frame = cap.read()

if not ret:

break

frame = crop\_center\_square(frame)

frame = cv2.resize(frame, resize)

frame = frame[:, :, [2, 1, 0]] # Convert BGR to RGB

frames.append(frame)

if len(frames) == max\_frames:

break

finally:

cap.release()

return np.array(frames)

# Function to crop the center square of a video frame

def crop\_center\_square(frame):

y, x = frame.shape[0:2]

min\_dim = min(y, x)

start\_x = (x // 2) - (min\_dim // 2)

start\_y = (y // 2) - (min\_dim // 2)

return frame[start\_y : start\_y + min\_dim, start\_x : start\_x + min\_dim]

def prepare\_single\_video(frames):

frames = frames[None, ...]

frame\_mask = np.zeros(shape=(1, MAX\_SEQ\_LENGTH,), dtype="bool")

frame\_features = np.zeros(shape=(1, MAX\_SEQ\_LENGTH, NUM\_FEATURES), dtype="float32")

for i, batch in enumerate(frames):

video\_length = batch.shape[0]

length = min(MAX\_SEQ\_LENGTH, video\_length)

for j in range(length):

frame\_features[i, j, :] = feature\_extractor.predict(batch[None, j, :])

frame\_mask[i, :length] = 1 # 1 = not masked, 0 = masked

return frame\_features, frame\_mask

@app.route('/')

def home():

return render\_template('index.html')

# Endpoint to predict if the video is deepfake or not

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

def predict():

if 'video' not in request.files:

return render\_template('index.html', prediction="No video file provided")

video = request.files['video']

if video.filename == '':

return render\_template('index.html', prediction="No file selected")

filename = secure\_filename(video.filename)

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

video\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

video.save(video\_path)

frames = load\_video(video\_path)

frame\_features, frame\_mask = prepare\_single\_video(frames)

# Predict using the model

prediction = model.predict([frame\_features, frame\_mask])[0]

result = 'FAKE' if prediction >= 0.51 else 'REAL'

confidence = float(prediction) # Convert to Python float for JSON serialization

os.remove(video\_path) # Clean up the uploaded video

prediction\_msg = f"The video is {result} with {confidence:.2f}% confidence"

return render\_template('index.html', prediction=prediction\_msg)

# Function to crop the center square of a video frame

def crop\_center\_square(frame):

y, x = frame.shape[0:2]

min\_dim = min(y, x)

start\_x = (x // 2) - (min\_dim // 2)

start\_y = (y // 2) - (min\_dim // 2)

return frame[start\_y:start\_y + min\_dim, start\_x:start\_x + min\_dim]

# Create the uploads folder if it doesn’t exist and run the app

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

if not os.path.exists('uploads'):

os.makedirs('uploads')

app.run(debug=True)

**INDEX.html CODE**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Deepfake Video Analyzer</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;700&display=swap" rel="stylesheet" />

<style>

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: 'Poppins', sans-serif;

background:

linear-gradient(rgba(33, 147, 176, 0.6), rgba(109, 213, 237, 0.6)),

url('https://s24806.pcdn.co/wp-content/uploads/2024/07/TrendMicro-Deepfake-videostill2-970-copy.jpg');

background-size: cover;

background-position: center;

min-height: 100vh;

display: flex;

align-items: center;

justify-content: center;

}

.container {

background-color: white;

border-radius: 20px;

box-shadow: 0 10px 40px rgba(0, 0, 0, 0.15);

padding: 50px 40px;

width: 90%;

max-width: 550px;

text-align: center;

}

h1 {

font-size: 2rem;

color: #333;

margin-bottom: 10px;

}

.subtitle {

color: #666;

font-size: 0.95rem;

margin-bottom: 30px;

}

input[type="file"] {

padding: 14px;

width: 100%;

border: 2px dashed #bbb;

border-radius: 10px;

margin-bottom: 20px;

font-size: 15px;

background-color: #f9f9f9;

cursor: pointer;

transition: border-color 0.3s ease;

}

input[type="file"]:hover {

border-color: #2193b0;

}

input[type="submit"] {

padding: 14px 30px;

background: linear-gradient(135deg, #667eea, #764ba2);

color: white;

font-size: 16px;

border: none;

border-radius: 8px;

font-weight: 600;

cursor: pointer;

transition: all 0.3s ease;

}

input[type="submit"]:hover {

background: linear-gradient(135deg, #5a67d8, #6b46c1);

transform: scale(1.03);

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);

}

.result {

margin-top: 30px;

padding: 18px 20px;

font-size: 1.1rem;

font-weight: bold;

border-radius: 12px;

animation: fadeIn 0.6s ease-in-out;

}

.real {

background-color: #e6f4ea;

color: #2e7d32;

border: 2px solid #a5d6a7;

}

.fake {

background-color: #ffebee;

color: #c62828;

border: 2px solid #ef9a9a;

}

footer {

margin-top: 40px;

font-size: 12px;

color: #999;

}

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(10px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

</style>

</head>

<body>

<div class="container">

<h1>Deepfake Video Analyzer</h1>

<p class="subtitle">Upload a video and let our AI tell you if it’s <strong>REAL</strong> or <strong>FAKE</strong>.</p>

<form action="/predict" method="post" enctype="multipart/form-data">

<input type="file" name="video" accept="video/\*" required />

<br />

<input type="submit" value="Analyze Now" />

</form>

{% if prediction %}

<div class="result {% if 'REAL' in prediction %}real{% else %}fake{% endif %}">

{{ prediction }}

</div>

{% endif %}

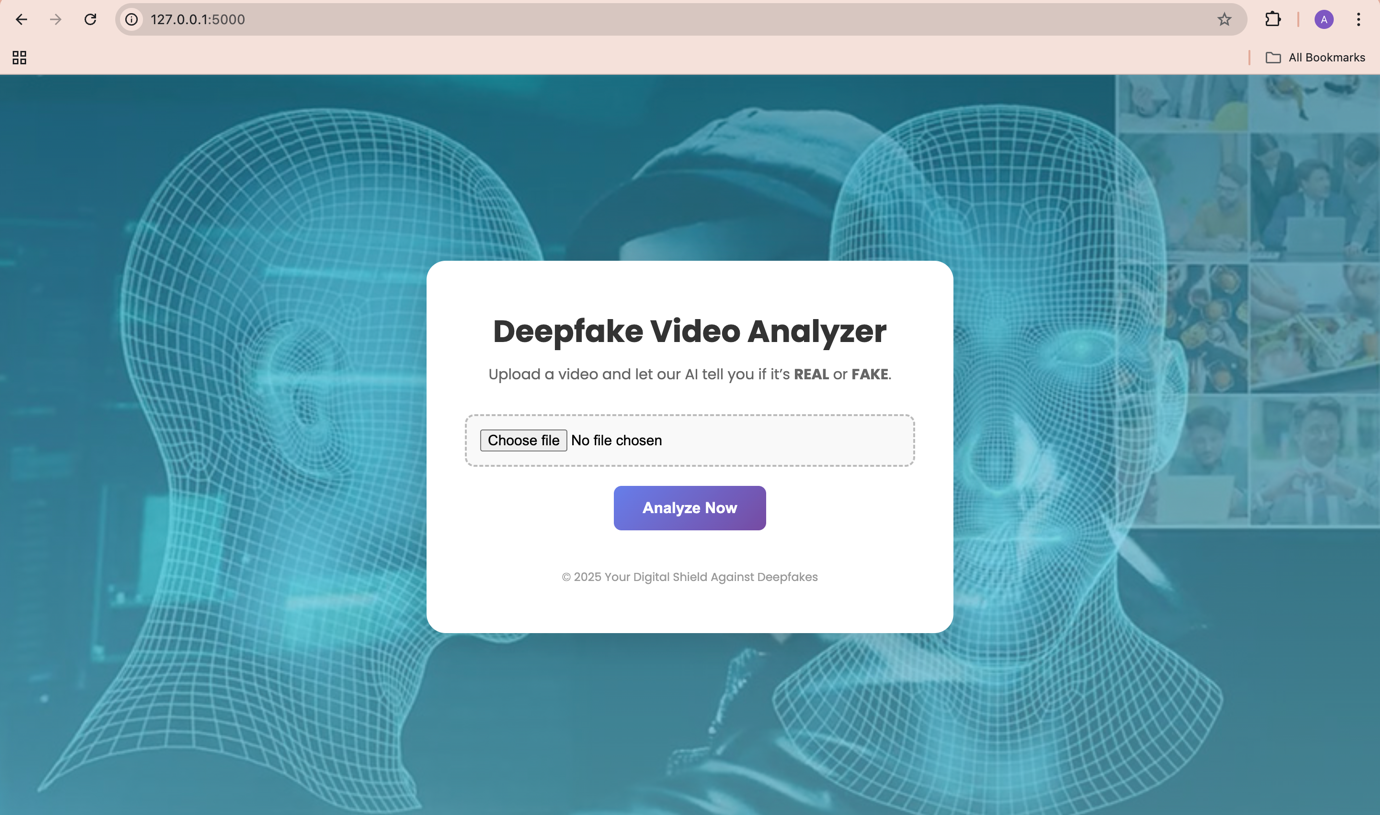
<footer>© 2025 Your Digital Shield Against Deepfakes</footer>

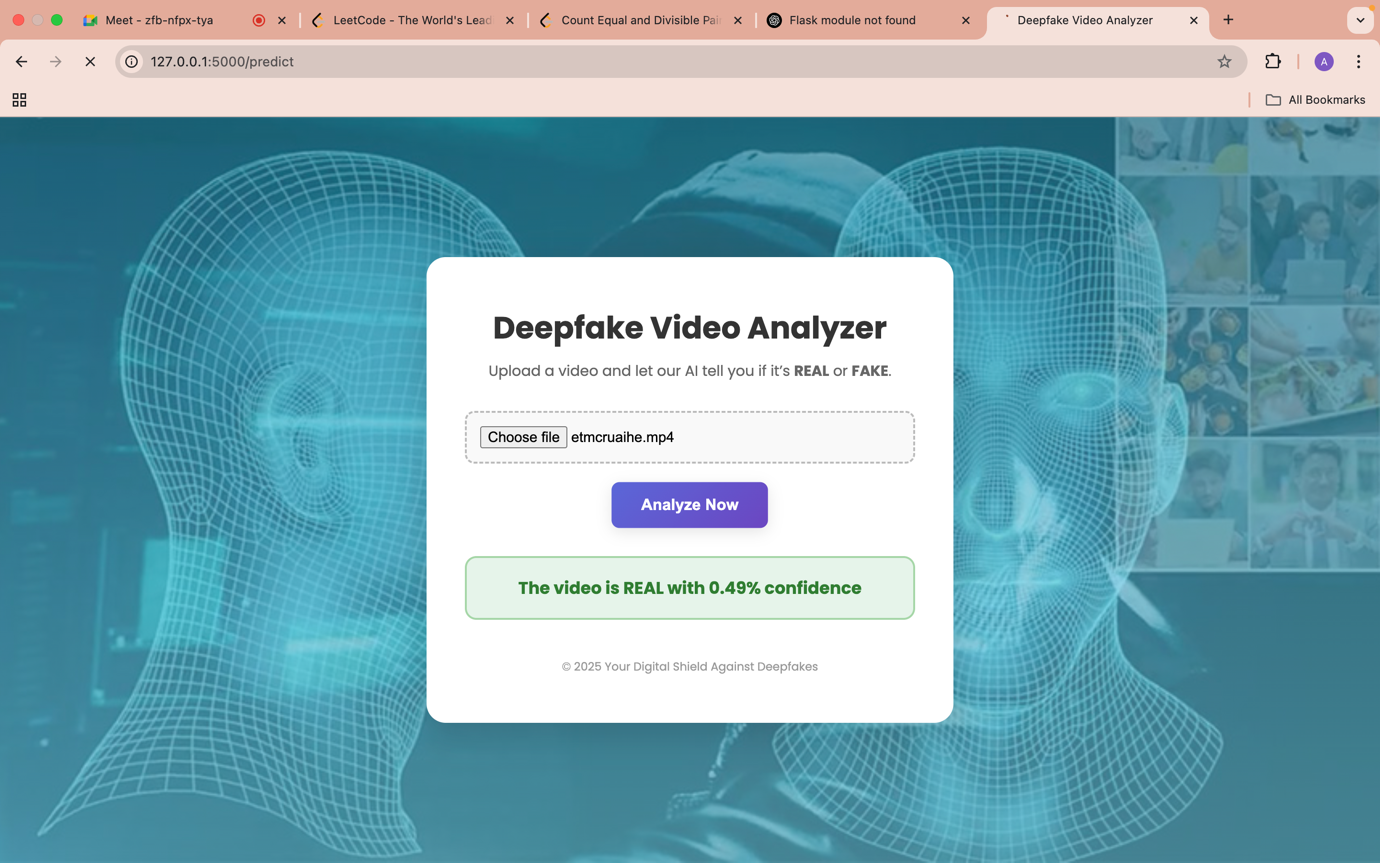
</div>

</body>

</html>

**USER INTERFACE**

****

**Testing / Security**

**Software Testing in Deepfake Detection System**

***1. Introduction***

Software testing ensures the reliability, accuracy, and security of the Deepfake Detection System. This section discusses the testing methodologies applied, including verification, validation, black-box testing, and white-box testing.

***2. Verification and Validation***

**2.1 Verification**

Verification is the process of evaluating whether the system meets its specified design and requirements before actual execution. It ensures that the system is being developed correctly.

**Techniques Used:**

* Requirement Reviews: Ensuring that the system aligns with functional specifications.
* Design Reviews: Evaluating the architecture, data flow, and security implementations.
* Code Reviews: Checking for proper coding practices, syntax errors, and logical consistency.

**2.2 Validation**

Validation ensures that the final product meets the user's expectations and requirements. It checks if the system functions as intended in real-world scenarios.

**Techniques Used:**

* Functional Testing: Verifying that the model correctly classifies videos as "REAL" or "FAKE."
* Integration Testing: Ensuring seamless interaction between components (Flask API, Deep Learning Model, and UI).
* System Testing: Testing the full system workflow, including video processing, feature extraction, and prediction.

***3. Black Box Testing***

Black box testing is performed without knowledge of the internal code structure. It focuses on input-output behaviour.

**Techniques Used:**

* **Equivalence Partitioning:** Testing with different categories of video inputs (e.g., real videos, manipulated videos, low/high-resolution videos).
* **Boundary Value Analysis:** Checking performance for edge cases, such as extremely short or long videos.
* **User Interface Testing:** Ensuring that users can upload videos, receive predictions, and interpret results correctly.

***4. White Box Testing***

White box testing is performed with knowledge of the internal code, focusing on code logic and performance.

**Techniques Used:**

* **Unit Testing:** Testing individual components such as feature extraction, frame processing, and model inference.
* **Code Coverage Analysis:** Ensuring all parts of the model and API are tested, covering all branches, loops, and conditions.
* **Security Testing:** Checking vulnerabilities in the API, preventing injection attacks, and ensuring secure data transmission.

***5. Conclusion***

By applying a combination of verification, validation, black-box, and white-box testing techniques, the Deepfake Detection System ensures robustness, reliability, and security. These testing approaches help optimize model accuracy, improve user experience, and safeguard against adversarial attacks.

**Future Scope**

The Deepfake Detection System can be further enhanced by incorporating additional features and technologies, including:

* ***Voice Recognition:*** Analysing voice deepfakes alongside video deepfakes to improve accuracy.
* ***Multi-modal Analysis:*** Combining facial recognition, speech analysis, and behavioural patterns for better deepfake detection.
* ***Real-time Detection:*** Enhancing model efficiency to provide instant deepfake verification during live streams.
* ***Blockchain Integration:*** Using blockchain technology to verify the authenticity of videos and prevent tampering.
* ***Adaptive AI Models:*** Implementing self-learning models that continuously improve detection based on evolving deepfake techniques.

18

**Bibliography**

1. Goodfellow, Ian, et al. "Deep Learning." MIT Press, 2016.
2. Chollet, François. "Deep Learning with Python." Manning Publications, 2017.
3. Mirsky, Yisroel, and Wenke Lee. "The Creation and Detection of Deepfakes: A Survey." ACM Computing Surveys, 2021.
4. Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." arXiv preprint arXiv:1412.6980, 2014.
5. Zhang, Y., et al. "A Review of Deep Learning-Based Methods for Deepfake Detection." IEEE Access, 2020.
6. TensorFlow Documentation: <https://www.tensorflow.org/>
7. OpenCV Documentation: <https://docs.opencv.org/>
8. Keras API Reference: <https://keras.io/api/>
9. Scikit-Learn Documentation: <https://scikit-learn.org/stable/>

10.Python Official Documentation: https://docs.python.org/3/

19