# MINI PROJECT REPORT

**On**

**“Desgin and develop a tool for digital forensic of video”**

A Report Submitted for a mini project : Laboratory Practice-IV in 7th Semester Computer Engineering.

**Third Year (COMPUTER ENGINEERING)**

**Academic Year 2025-26**

**Submitted by-**

**Kartik Anil Nagare**

**Roll No.: B22050 Div.: B**

****

**Zeal Education Society's**

**Zeal College of Engineering & Research Narhe, Pune – 411041**

**Department of Computer Engineering**

**Zeal Education Society’s**

**Zeal College of Engineering & Research Department of Computer Engineering**

****

# CERTIFICATE

Certified that the project entitled “Design and develop a tool for digital forensic of video”is a bonafide work carried out by Kartik Anil Nagare (B22050). It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

**Prof. Rupali Jadhav Prof. A. V. Mote**

Project Guide H. O. D

# ACKNOWLEDGMENT

We take this opportunity to thank our project guide **Prof. Rupali Jadhav** mam and Head of the department **Prof. A. V. Mote** mam for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of Computer Engineering Department for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, Internet access and important books.

# INDEX

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **CONTENT** | **Page No.** |
| **1.** | **Abstract** | **5** |
| **2.** | **Software Requirement** | **6** |
| **3.** | **Introduction** | **7** |
| **4.** | **Problem Statement** | **8** |
| **5.** | **Objective And Outcome** | **8** |
| **6.** | **Implementation Code** | **9** |
| **7.** | **Output** | **27** |
| **8.** | **Conclusion** | **28** |
| **9.** | **References** | **29** |

**ABSTRACT**

In today’s fast-paced digital world, videos have become one of the most powerful and commonly used forms of evidence in cybercrime investigations. However, with the rapid advancement of sophisticated editing tools and AI-generated fake videos (such as deepfakes), tampering with video content has become increasingly easy and convincing. This growing challenge makes it difficult to determine whether a video is authentic or has been altered, raising serious concerns about trust and reliability.

This project aims to design and develop an **intelligent video forensic tool** that assists investigators, law enforcement agencies, and digital forensic experts in verifying the authenticity of video evidence. The tool performs a comprehensive analysis by examining **hidden metadata**, conducting **frame-by-frame inspection**, and detecting various forms of manipulation such as **cuts, splices, overlays, or inserted fake elements**. Additionally, it incorporates **machine learning algorithms** to identify deepfakes, detect inconsistencies, and highlight unusual patterns that might indicate tampering. The system supports a wide range of video formats and offers an intuitive, user-friendly interface that allows investigators to **review, annotate, and analyze videos along a clear timeline**. Furthermore, the tool automatically generates **detailed, well-structured forensic reports** that can be used as credible digital evidence in courtrooms, legal proceedings, or cybercrime investigations, ultimately helping strengthen the process of digital truth verification.

# SOFTWARE AND HARDWARE REQUIREMENT

**Software Requirements:**

1. Python Version: Python 3.7.0
2. Libraries/Packages:
   * OpenCV: For reading, processing, and analyzing video frames
   * PyMediaInfo: To extract metadata from video files like codec, duration, frame rate
   * FFmpeg: For video format conversion, decoding, and low-level video
   * processing.
   * NumPy: Version 1.x or higher – for numerical operations
   * Hashlib: To calculate cryptographic hashes (SHA-256, MD5) for file integrity verification.
3. Operating System: Windows, macOS, or Linux
4. Text Editor/IDE: Any Python-supported IDE can be used such as: VSCode

**Hardware Requirements:**

1. Processor:
   * Minimum: Intel Core i3 or equivalent
   * Recommended: Intel Core i5/i7 or AMD equivalent
2. Memory (RAM):
   * Minimum: 4 GB
   * Recommended: 8 GB or more
3. Storage:
   * Minimum: 2 GB free space
   * Recommended: 5 GB or more
4. Graphics (GPU):
   * Minimum: Integrated graphics (e.g., Intel HD Graphics)

# INTRODUCTION

In today’s world, videos have become a powerful and essential part of daily life. They are shared widely on social media, featured in news reports, stored on personal devices, and frequently used as key evidence in criminal investigations. However, the growing availability of advanced editing tools and the rise of AI-generated fake videos, such as deepfakes, have made it increasingly difficult to determine whether a video is genuine or manipulated. This has created serious challenges for investigators, journalists, and cybersecurity professionals who rely on videos as a source of truth.

To address this issue, this project focuses on developing a **video forensic tool** designed to verify the authenticity and integrity of digital videos. The tool examines a video’s **metadata**, performs a **frame-by-frame analysis**, and generates a unique **digital fingerprint (hash)** to detect any signs of tampering or unauthorized modifications.

The main objective of this project is to make it easier to uncover the truth behind any video. Whether it is used in courtrooms, for verifying news content, or during cybercrime investigations, this tool ensures that digital video evidence remains **reliable, accurate, and trustworthy**.

**PROBLEM STATEMENT**

With the rapid advancement of powerful video editing tools and AI-generated deepfakes, verifying the authenticity of videos has become more challenging than ever. This growing issue poses significant risks in critical areas such as cybercrime investigations, law enforcement, legal proceedings, and digital media verification. Misleading or tampered videos can spread false information, compromise investigations, and undermine trust in digital evidence. To address this problem, there is a strong need for a **reliable, accurate, and user-friendly video forensic tool** that can thoroughly analyze video files and detect signs of manipulation. Such a tool would help professionals quickly identify altered content, ensure the integrity of digital evidence, and support informed decision-making in investigations and media verification.

**OBJECTIVES**

The main objective of this project is to develop a **user-friendly video forensic tool** that assists investigators and cybersecurity professionals in analyzing digital video files efficiently and accurately. The tool is designed to **extract and display essential metadata**, including details such as video duration, resolution, frame rate, codec type, and device information, which can provide valuable insights during investigations. It also enables **frame-by-frame analysis** to detect potential signs of tampering, such as frame drops, black frames, or splicing, which are common indicators of manipulation.

For better visual inspection, the tool offers **real-time video playback synchronized with forensic data**, allowing investigators to examine suspicious frames more effectively. Moreover, it is designed to **support multiple widely used video formats.**

.

**OUTCOMES**

The project successfully developed a **simple, efficient, and reliable video forensic tool** that enables users to analyze digital videos with ease. The tool can **extract and display metadata**, **verify video integrity using cryptographic hash values**, and **perform frame-by-frame analysis** to detect possible signs of tampering. Additionally, it provides **video playback for visual inspection**, allowing investigators to examine suspicious segments more closely. Overall, the tool helps users quickly determine whether a video is **authentic or has been altered**, making it a valuable asset for digital investigations, media verification, and legal proceedings.

**IMPLEMENTATION CODE**

**main.py**

from flask import Flask, render\_template, request, jsonify, send\_file, url\_for

import cv2

import hashlib

import os

import json

import base64

from werkzeug.utils import secure\_filename

from pymediainfo import MediaInfo

import threading

import time

from datetime import datetime

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

app.config['MAX\_CONTENT\_LENGTH'] = 500 \* 1024 \* 1024 # 500MB max file size

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

app.config['SECRET\_KEY'] = 'your-secret-key-here'

# Ensure upload directory exists

os.makedirs(app.config['UPLOAD\_FOLDER'], exist\_ok=True)

# Global variables for analysis status

analysis\_progress = {}

video\_frames = {}

ALLOWED\_EXTENSIONS = {'mp4', 'avi', 'mov', 'mkv', 'wmv', 'flv', 'webm'}

def allowed\_file(filename):

return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

def extract\_metadata(video\_path):

try:

media\_info = MediaInfo.parse(video\_path)

data = {}

for track in media\_info.tracks:

if track.track\_type == "Video":

data['video'] = {

'codec': track.codec or 'N/A',

'duration': f"{track.duration or 'N/A'} ms",

'width': f"{track.width or 'N/A'} px",

'height': f"{track.height or 'N/A'} px",

'frame\_rate': f"{track.frame\_rate or 'N/A'} fps",

'bit\_rate': f"{track.bit\_rate or 'N/A'} bps",

'color\_space': track.color\_space or 'N/A',

'scan\_type': track.scan\_type or 'N/A'

}

elif track.track\_type == "Audio":

data['audio'] = {

'codec': track.codec or 'N/A',

'sample\_rate': f"{track.sampling\_rate or 'N/A'} Hz",

'channels': track.channel\_s or 'N/A',

'bit\_rate': f"{track.bit\_rate or 'N/A'} bps"

}

return data if data else {'error': 'No metadata available'}

except Exception as e:

return {'error': f'Error extracting metadata: {str(e)}'}

def calculate\_file\_hash(path, file\_id):

try:

hash\_func = hashlib.sha256()

file\_size = os.path.getsize(path)

processed = 0

with open(path, "rb") as f:

while chunk := f.read(8192):

hash\_func.update(chunk)

processed += len(chunk)

progress = (processed / file\_size) \* 100

analysis\_progress[file\_id]['hash\_progress'] = progress

hash\_value = hash\_func.hexdigest()

result = {

'algorithm': 'SHA-256',

'file\_size': f"{file\_size:,} bytes",

'hash': hash\_value,

'hash\_breakdown': [hash\_value[i:i+16] for i in range(0, len(hash\_value), 16)]

}

return result

except Exception as e:

return {'error': f'Error calculating hash: {str(e)}'}

def analyze\_video\_frames(video\_path, file\_id):

try:

cap = cv2.VideoCapture(video\_path)

frame\_count = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

fps = cap.get(cv2.CAP\_PROP\_FPS)

dark\_frames = []

bright\_frames = []

normal\_frames = 0

frame\_number = 0

analysis\_limit = min(frame\_count, 1000)

while frame\_number < analysis\_limit:

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, frame\_number)

ret, frame = cap.read()

if not ret:

break

brightness = frame.mean()

if brightness < 30:

dark\_frames.append({'frame': frame\_number, 'brightness': round(brightness, 2)})

elif brightness > 200:

bright\_frames.append({'frame': frame\_number, 'brightness': round(brightness, 2)})

else:

normal\_frames += 1

frame\_number += 10

progress = (frame\_number / analysis\_limit) \* 100

analysis\_progress[file\_id]['frame\_progress'] = progress

cap.release()

result = {

'total\_frames': f"{frame\_count:,}",

'frame\_rate': f"{fps:.2f} fps",

'duration': f"{frame\_count/fps:.2f} seconds",

'dark\_frames': dark\_frames[:20], # Limit to first 20

'bright\_frames': bright\_frames[:20], # Limit to first 20

'normal\_frames': normal\_frames,

'summary': {

'dark\_count': len(dark\_frames),

'bright\_count': len(bright\_frames),

'normal\_count': normal\_frames

}

}

return result

except Exception as e:

return {'error': f'Error analyzing frames: {str(e)}'}

def get\_video\_frame(video\_path, frame\_number=0):

try:

cap = cv2.VideoCapture(video\_path)

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, frame\_number)

ret, frame = cap.read()

if ret:

# Resize frame

frame = cv2.resize(frame, (640, 480))

\_, buffer = cv2.imencode('.jpg', frame)

frame\_base64 = base64.b64encode(buffer).decode('utf-8')

cap.release()

return frame\_base64

cap.release()

return None

except Exception as e:

return None

def perform\_full\_analysis(video\_path, file\_id):

analysis\_progress[file\_id] = {

'status': 'running',

'metadata\_done': False,

'hash\_progress': 0,

'frame\_progress': 0,

'results': {}

}

try:

# Extract metadata

analysis\_progress[file\_id]['status'] = 'Extracting metadata...'

metadata = extract\_metadata(video\_path)

analysis\_progress[file\_id]['results']['metadata'] = metadata

analysis\_progress[file\_id]['metadata\_done'] = True

# Calculate hash

analysis\_progress[file\_id]['status'] = 'Calculating file hash...'

hash\_info = calculate\_file\_hash(video\_path, file\_id)

analysis\_progress[file\_id]['results']['hash'] = hash\_info

# Analyze frames

analysis\_progress[file\_id]['status'] = 'Analyzing video frames...'

frame\_analysis = analyze\_video\_frames(video\_path, file\_id)

analysis\_progress[file\_id]['results']['frames'] = frame\_analysis

# Get first frame for preview

first\_frame = get\_video\_frame(video\_path, 0)

analysis\_progress[file\_id]['results']['preview\_frame'] = first\_frame

analysis\_progress[file\_id]['status'] = 'complete'

except Exception as e:

analysis\_progress[file\_id]['status'] = f'error: {str(e)}'

@app.route('/')

def index():

return render\_template('index.html')

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

def upload\_file():

if 'file' not in request.files:

return jsonify({'error': 'No file uploaded'}), 400

file = request.files['file']

if file.filename == '':

return jsonify({'error': 'No file selected'}), 400

if file and allowed\_file(file.filename):

filename = secure\_filename(file.filename)

timestamp = str(int(time.time()))

filename = f"{timestamp}\_{filename}"

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

try:

file.save(filepath)

file\_id = timestamp

# Start analysis in background thread

analysis\_thread = threading.Thread(

target=perform\_full\_analysis,

args=(filepath, file\_id)

)

analysis\_thread.daemon = True

analysis\_thread.start()

return jsonify({

'success': True,

'file\_id': file\_id,

'filename': file.filename

})

except Exception as e:

return jsonify({'error': f'Failed to save file: {str(e)}'}), 500

return jsonify({'error': 'Invalid file type'}), 400

@app.route('/progress/<file\_id>')

def get\_progress(file\_id):

if file\_id not in analysis\_progress:

return jsonify({'error': 'File not found'}), 404

return jsonify(analysis\_progress[file\_id])

@app.route('/results/<file\_id>')

def get\_results(file\_id):

if file\_id not in analysis\_progress:

return jsonify({'error': 'File not found'}), 404

if analysis\_progress[file\_id]['status'] == 'complete':

return jsonify(analysis\_progress[file\_id]['results'])

return jsonify({'status': 'not\_ready'})

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

app.run(debug=True, host='0.0.0.0', port=5001)

index.py  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Video Forensic Analysis Tool</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.0/css/bootstrap.min.css" rel="stylesheet">

<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" rel="stylesheet">

<style>

:root {

--primary-color: #2c3e50;

--secondary-color: #3498db;

--success-color: #2ecc71;

--danger-color: #e74c3c;

--warning-color: #f39c12;

--info-color: #17a2b8;

--light-bg: #f8f9fa;

--dark-bg: #343a40;

}

body {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

min-height: 100vh;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

.main-container {

background: rgba(255, 255, 255, 0.95);

backdrop-filter: blur(10px);

border-radius: 20px;

box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);

margin: 20px;

padding: 0;

overflow: hidden;

}

.header {

background: linear-gradient(135deg, var(--primary-color) 0%, var(--secondary-color) 100%);

color: white;

padding: 30px;

text-align: center;

}

.header h1 {

margin: 0;

font-size: 2.5rem;

font-weight: 700;

text-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);

}

.header p {

margin: 10px 0 0 0;

opacity: 0.9;

font-size: 1.1rem;

}

.upload-section {

padding: 40px;

text-align: center;

}

.upload-zone {

border: 3px dashed #dee2e6;

border-radius: 15px;

padding: 60px 20px;

transition: all 0.3s ease;

cursor: pointer;

background: var(--light-bg);

position: relative;

overflow: hidden;

}

.upload-zone:hover {

border-color: var(--secondary-color);

background: rgba(52, 152, 219, 0.05);

transform: translateY(-2px);

}

.upload-zone.dragover {

border-color: var(--success-color);

background: rgba(46, 204, 113, 0.1);

transform: scale(1.02);

}

.upload-icon {

font-size: 4rem;

color: var(--secondary-color);

margin-bottom: 20px;

}

.upload-text {

font-size: 1.3rem;

color: #6c757d;

margin-bottom: 15px;

}

.file-input {

display: none;

}

.btn-upload {

background: linear-gradient(135deg, var(--secondary-color) 0%, var(--success-color) 100%);

border: none;

padding: 12px 30px;

font-size: 1.1rem;

font-weight: 600;

border-radius: 50px;

color: white;

transition: all 0.3s ease;

}

.btn-upload:hover {

transform: translateY(-2px);

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);

color: white;

}

.progress-section {

display: none;

padding: 40px;

}

.progress-card {

background: white;

border-radius: 15px;

padding: 30px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

margin-bottom: 20px;

}

.progress-item {

display: flex;

align-items: center;

margin-bottom: 20px;

padding: 15px;

border-radius: 10px;

background: var(--light-bg);

}

.progress-icon {

font-size: 1.5rem;

width: 40px;

text-align: center;

margin-right: 15px;

}

.progress-content {

flex: 1;

}

.progress-title {

font-weight: 600;

margin-bottom: 5px;

}

.results-section {

display: none;

padding: 40px;

}

.results-tabs {

background: white;

border-radius: 15px;

overflow: hidden;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);

}

.nav-tabs {

border-bottom: none;

background: var(--primary-color);

}

.nav-tabs .nav-link {

border: none;

color: rgba(255, 255, 255, 0.7);

font-weight: 600;

padding: 15px 25px;

transition: all 0.3s ease;

}

.nav-tabs .nav-link.active {

background: white;

color: var(--primary-color);

border-radius: 0;

}

.nav-tabs .nav-link:hover {

color: white;

background: rgba(255, 255, 255, 0.1);

}

.tab-content {

padding: 30px;

min-height: 400px;

}

.info-card {

background: var(--light-bg);

border-radius: 10px;

padding: 20px;

margin-bottom: 15px;

}

.info-label {

font-weight: 600;

color: var(--primary-color);

font-size: 0.9rem;

text-transform: uppercase;

letter-spacing: 1px;

}

.info-value {

font-size: 1.1rem;

color: #495057;

margin-top: 5px;

}

.hash-display {

font-family: 'Courier New', monospace;

background: #f8f9fa;

padding: 15px;

border-radius: 8px;

word-break: break-all;

font-size: 0.9rem;

border-left: 4px solid var(--info-color);

}

.frame-alert {

padding: 10px 15px;

border-radius: 8px;

margin-bottom: 10px;

font-size: 0.9rem;

}

.frame-alert.dark {

background: rgba(108, 117, 125, 0.1);

border-left: 4px solid #6c757d;

}

.frame-alert.bright {

background: rgba(255, 193, 7, 0.1);

border-left: 4px solid var(--warning-color);

}

.video-preview {

text-align: center;

margin: 20px 0;

}

.video-preview img {

max-width: 100%;

border-radius: 10px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);

}

.status-badge {

padding: 5px 12px;

border-radius: 20px;

font-size: 0.85rem;

font-weight: 600;

}

.status-processing {

background: rgba(255, 193, 7, 0.2);

color: var(--warning-color);

}

.status-complete {

background: rgba(46, 204, 113, 0.2);

color: var(--success-color);

}

.status-error {

background: rgba(231, 76, 60, 0.2);

color: var(--danger-color);

}

.spinner {

animation: spin 1s linear infinite;

}

@keyframes spin {

from { transform: rotate(0deg); }

to { transform: rotate(360deg); }

}

.fade-in {

animation: fadeIn 0.5s ease-in;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.summary-stats {

display: flex;

justify-content: space-around;

text-align: center;

margin: 20px 0;

}

.stat-item {

background: white;

padding: 15px;

border-radius: 10px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

flex: 1;

margin: 0 10px;

}

.stat-value {

font-size: 1.5rem;

font-weight: 700;

color: var(--primary-color);

}

.stat-label {

font-size: 0.9rem;

color: #6c757d;

margin-top: 5px;

}

@media (max-width: 768px) {

.main-container {

margin: 10px;

}

.header h1 {

font-size: 2rem;

}

.upload-zone {

padding: 40px 15px;

}

.upload-icon {

font-size: 3rem;

}

.summary-stats {

flex-direction: column;

}

.stat-item {

margin: 5px 0;

}

}

</style>

</head>

<body>

<div class="container-fluid">

<div class="main-container">

<!-- Header -->

<div class="header">

<h1><i class="fas fa-search"></i> Video Forensic Analysis Tool</h1>

<p>Professional video analysis for digital forensics investigations</p>

</div>

<!-- Upload Section -->

<div id="upload-section" class="upload-section">

<div class="upload-zone" id="upload-zone">

<div class="upload-icon">

<i class="fas fa-cloud-upload-alt"></i>

</div>

<div class="upload-text">

Drag and drop your video file here or click to browse

</div>

<div class="mt-3">

<small class="text-muted">

Supported formats: MP4, AVI, MOV, MKV, WMV, FLV, WEBM<br>

Maximum file size: 500MB

</small>

</div>

<input type="file" id="file-input" class="file-input" accept=".mp4,.avi,.mov,.mkv,.wmv,.flv,.webm">

<button type="button" class="btn btn-upload mt-3" onclick="document.getElementById('file-input').click()">

<i class="fas fa-folder-open me-2"></i>Select Video File

</button>

</div>

</div>

<!-- Progress Section -->

<div id="progress-section" class="progress-section">

<div class="progress-card fade-in">

<h4 class="mb-4"><i class="fas fa-cogs me-2"></i>Analysis in Progress</h4>

<div id="selected-file" class="mb-4">

<strong>File:</strong> <span id="filename"></span>

</div>

<div id="progress-items">

<!-- Progress items will be inserted here -->

</div>

<div class="mt-4">

<div id="current-status" class="text-center">

<span class="status-badge status-processing">

<i class="fas fa-spinner spinner me-2"></i>Initializing...

</span>

</div>

</div>

</div>

</div>

<!-- Results Section -->

<div id="results-section" class="results-section">

<div class="results-tabs fade-in">

<ul class="nav nav-tabs" id="results-tabs" role="tablist">

<li class="nav-item" role="presentation">

<button class="nav-link active" id="overview-tab" data-bs-toggle="tab" data-bs-target="#overview" type="button" role="tab">

<i class="fas fa-chart-line me-2"></i>Overview

</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="metadata-tab" data-bs-toggle="tab" data-bs-target="#metadata" type="button" role="tab">

<i class="fas fa-info-circle me-2"></i>Metadata

</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="hash-tab" data-bs-toggle="tab" data-bs-target="#hash" type="button" role="tab">

<i class="fas fa-fingerprint me-2"></i>Hash Analysis

</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="frames-tab" data-bs-toggle="tab" data-bs-target="#frames" type="button" role="tab">

<i class="fas fa-film me-2"></i>Frame Analysis

</button>

</li>

</ul>

<div class="tab-content" id="results-tab-content">

<div class="tab-pane fade show active" id="overview" role="tabpanel">

<div id="overview-content">

<!-- Overview content will be inserted here -->

</div>

</div>

<div class="tab-pane fade" id="metadata" role="tabpanel">

<div id="metadata-content">

<!-- Metadata content will be inserted here -->

</div>

</div>

<div class="tab-pane fade" id="hash" role="tabpanel">

<div id="hash-content">

<!-- Hash content will be inserted here -->

</div>

</div>

<div class="tab-pane fade" id="frames" role="tabpanel">

<div id="frames-content">

<!-- Frame analysis content will be inserted here -->

</div>

</div>

</div>

</div>

<div class="text-center mt-4">

<button type="button" class="btn btn-upload" onclick="resetTool()">

<i class="fas fa-redo me-2"></i>Analyze Another Video

</button>

</div>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.0/js/bootstrap.bundle.min.js"></script>

<script>

let currentFileId = null;

let progressInterval = null;

// File upload handling

const uploadZone = document.getElementById('upload-zone');

const fileInput = document.getElementById('file-input');

uploadZone.addEventListener('click', () => {

fileInput.click();

});

uploadZone.addEventListener('dragover', (e) => {

e.preventDefault();

uploadZone.classList.add('dragover');

});

uploadZone.addEventListener('dragleave', (e) => {

e.preventDefault();

uploadZone.classList.remove('dragover');

});

uploadZone.addEventListener('drop', (e) => {

e.preventDefault();

uploadZone.classList.remove('dragover');

const files = e.dataTransfer.files;

if (files.length > 0) {

handleFileUpload(files[0]);

}

});

fileInput.addEventListener('change', (e) => {

if (e.target.files.length > 0) {

handleFileUpload(e.target.files[0]);

}

});

function handleFileUpload(file) {

const formData = new FormData();

formData.append('file', file);

document.getElementById('filename').textContent = file.name;

showSection('progress');

fetch('/upload', {

method: 'POST',

body: formData

})

.then(response => response.json())

.then(data => {

if (data.success) {

currentFileId = data.file\_id;

startProgressMonitoring();

} else {

showError(data.error || 'Upload failed');

}

})

.catch(error => {

showError('Upload failed: ' + error.message);

});

}

function startProgressMonitoring() {

progressInterval = setInterval(() => {

if (currentFileId) {

fetch(`/progress/${currentFileId}`)

.then(response => response.json())

.then(data => {

updateProgress(data);

if (data.status === 'complete') {

clearInterval(progressInterval);

loadResults();

} else if (data.status && data.status.startsWith('error')) {

clearInterval(progressInterval);

showError(data.status);

}

})

.catch(error => {

console.error('Progress monitoring error:', error);

});

}

}, 1000);

}

function updateProgress(data) {

const statusElement = document.getElementById('current-status');

const progressItems = document.getElementById('progress-items');

if (data.status === 'complete') {

statusElement.innerHTML = '<span class="status-badge status-complete"><i class="fas fa-check me-2"></i>Analysis Complete</span>';

} else if (data.status && data.status.startsWith('error')) {

statusElement.innerHTML = `<span class="status-badge status-error"><i class="fas fa-exclamation-triangle me-2"></i>${data.status}</span>`;

} else {

statusElement.innerHTML = `<span class="status-badge status-processing"><i class="fas fa-spinner spinner me-2"></i>${data.status || 'Processing...'}</span>`;

}

// Update progress items

progressItems.innerHTML = `

<div class="progress-item">

<div class="progress-icon ${data.metadata\_done ? 'text-success' : 'text-muted'}">

<i class="fas ${data.metadata\_done ? 'fa-check-circle' : 'fa-info-circle'}"></i>

</div>

<div class="progress-content">

<div class="progress-title">Metadata Extraction</div>

<div class="text-muted">Extracting video and audio properties</div>

</div>

</div>

<div class="progress-item">

<div class="progress-icon ${data.hash\_progress === 100 ? 'text-success' : 'text-muted'}">

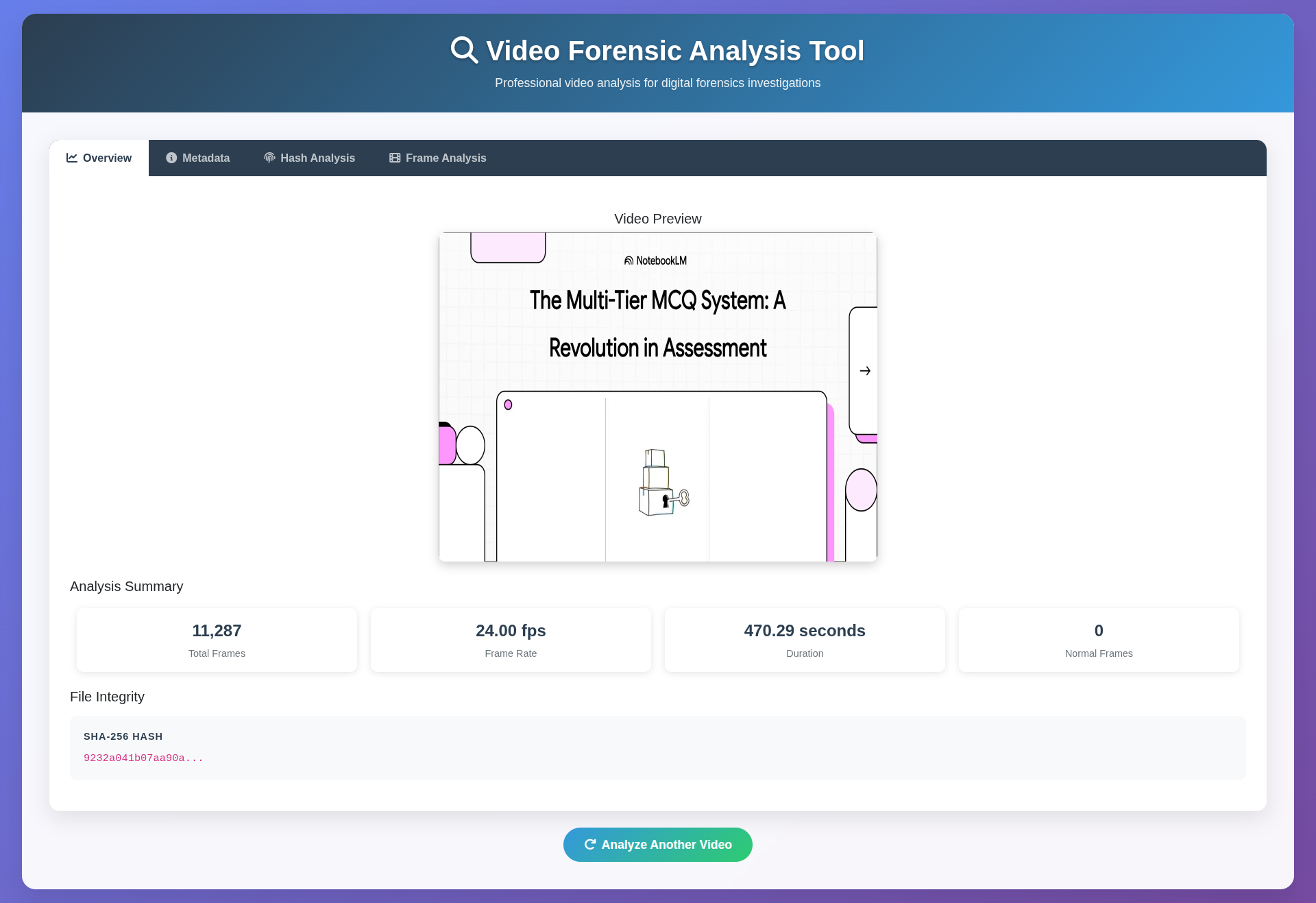
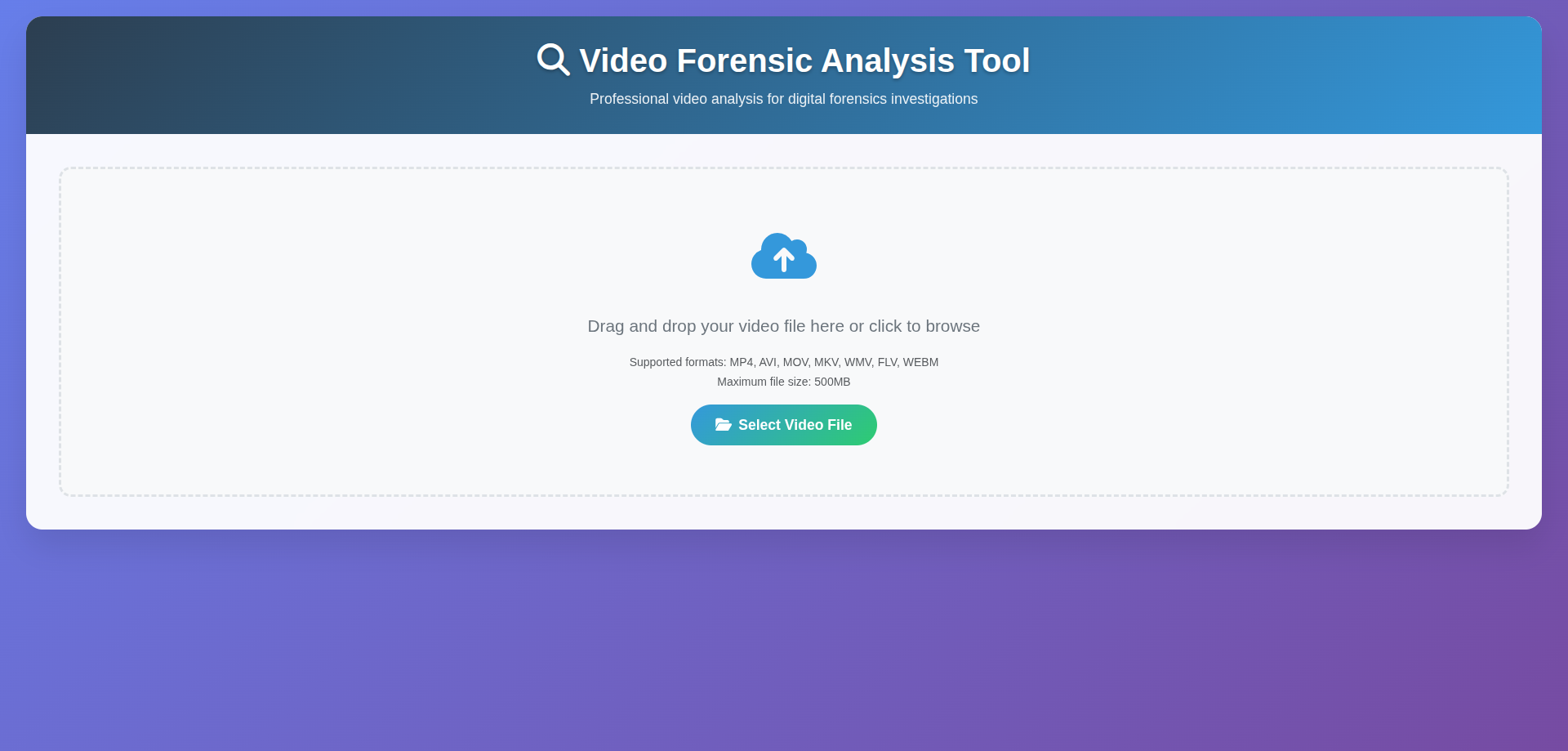
<i class="fas ${data.hash\_progress === 100 ? 'fa-check-circle' : 'fa-fingerprint'}"></i>

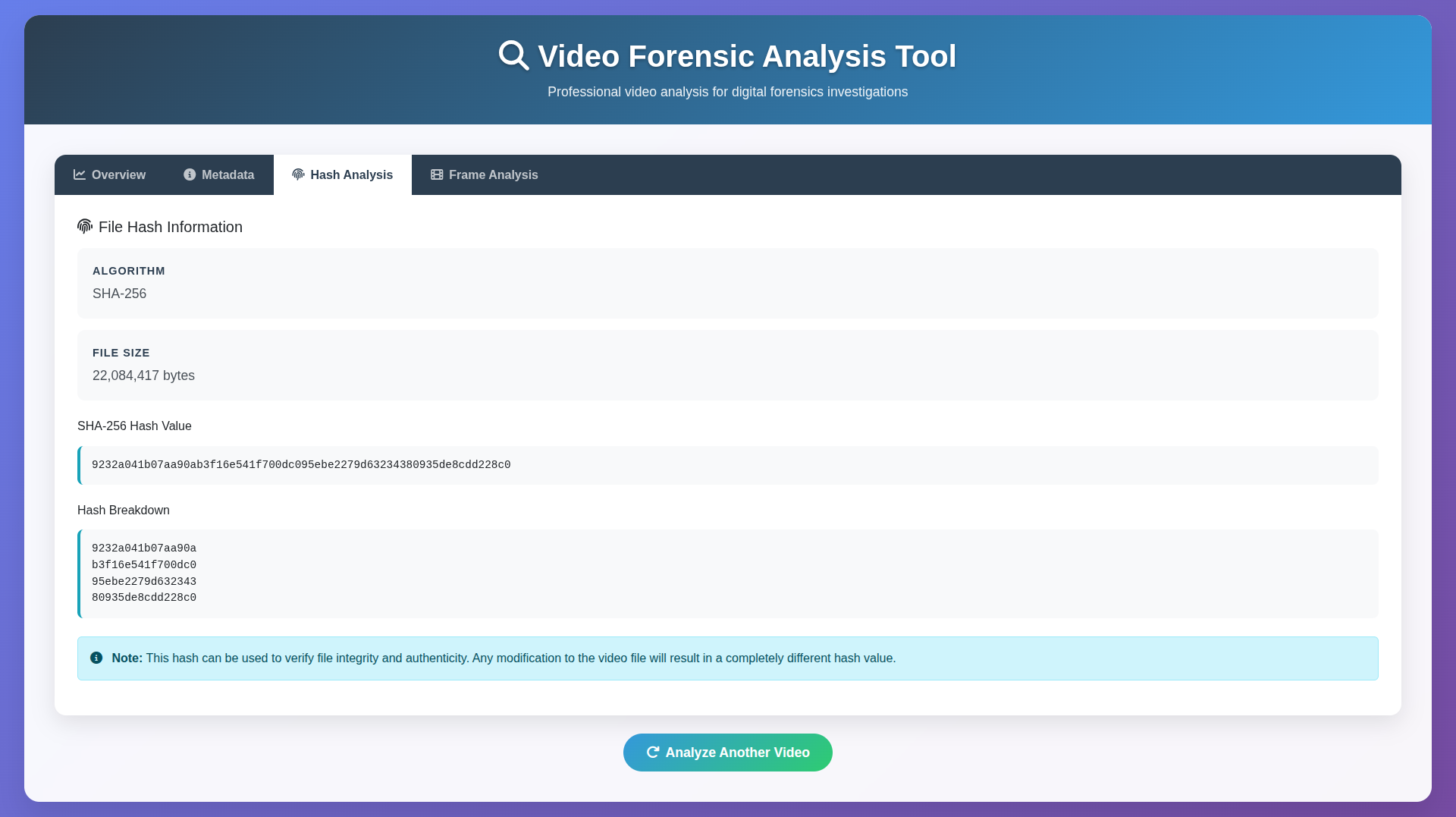
</div

</body>

</html>

**OUTPUT**

****



# CONCLUSION

This project successfully demonstrates the design and implementation of a **digital video forensic tool** aimed at detecting video tampering and verifying the authenticity of video evidence. By integrating **metadata extraction, SHA-256 hash verification, and frame-by-frame analysis**, the tool provides a dependable method for identifying potential manipulations in video files. It features an **intuitive graphical interface** and supports **real-time video playback**, making it accessible to users with varying levels of technical expertise. This tool serves as a valuable resource in **digital forensics, cybercrime investigations, and any domain where confirming the originality and integrity of video content is critical**.

.

## REFERENCES: -

1. Matplotlib Documentation. (n.d.). Matplotlib - Python plotting library. Retrieved from https://matplotlib.org/
2. Scikit-learn Documentation. (n.d.). Scikit-learn - Machine learning library for Python. Retrieved from https://scikit-learn.org/
3. OpenCV Documentation. https://opencv.org/ (Used for video frame processing and playback.)
4. Python hashlib Library Documentation. https://docs.python.org/3/library/hashlib.html (Used for generating SHA-256 hash of video files.)