Deep Fake Lab

GC Arslan Sarwar

GC Hassan Nazir

GC Saram Ashfaq

FC Basil Mikhled

Supervised by:

Lt Col Muhammad Imran Javaid

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad, in partial fulfillment for the requirements of B.E Degree in Electrical Engineering.

June 2023

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report for the Final Year Project "Deep Fake Lab" is carried out by Arslan Sarwar, Hassan Nazir, Saram Ashfaq, Basil Mikhled, under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

"Deep Fake Lab"

is carried out by

GC Arslan Sarwar
GC Hassan Nazir
GC Saram Ashfaq
FC Basil Mikhled

Approved by Supervisor

Signature: _____

Name of Supervisor: Lt Col Muhammad Imran Javaid

Dated: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in supportof another award or qualification in either this institute or anywhere else.

Dedication

This thesis is dedicated to our Families, Teachers, Friends, and to our supervisor for their love, endless support, and encouragement.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and most of all our supervisor, Lt Col Muhammad Imran Javaid without

your guidance this wouldn't be possible.

And all the group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has 4 similarity index. Turnitin report endorsed by Supervisor is attached.

Name & Signature of Supervisor:

_____

Lt Col Muhammad Imran Javaid

Name & Signatures of Students:

_____

GC Arslan Sarwar

00000325018

_____

GC Saram Ashfaq

00000325063

GC Hassan Nazir

00000325602

FC Basil Mikhled

00000325420

‘

ABSTRACT

The increasing prevalence of deep fake videos, which are digitally manipulated videos that falsely depict individuals saying or doing things they never did, has led to a need for reliable detection methods. In this project, we propose a deep fake detection technique based on hashing, which involves generating a unique fingerprint of a video frame or image that can be used to detect any modifications. Our approach involves extracting frames from a given video and applying perceptual hashing techniques to each frame to generate a hash. Perceptual hashing involves generating a unique signature that captures the perceptual characteristics of an image or video, such as its color, texture, and shape. We compared the hash of each frame to a pre-existing database of hashes for authentic videos, and calculated the similarity score between the two hashes using the Hamming distance. If the similarity score was below a certain threshold, the frame was considered a deep fake. To evaluate the performance of our approach, we used a dataset of both authentic and deep fake videos and calculated the detection accuracy and false positive rate. Our results demonstrate that our technique achieved high detection accuracy while keeping the false positive rate low. Our approach offers a simple and efficient solution for deep fake detection, which can be easily integrated into existing video analysis pipelines. Our method can be used to detect deep fakes in real-time, making it useful for applications such as video authentication and online content moderation. Overall, our project provides a valuable contribution to the ongoing efforts to combat the spread of deep fake videos, and we believe that our technique has the potential to be an effective tool for detecting deep fakes in various settings.

Table of Contents

List of Figures

Chapter 1

Introduction

Deepfake technology is becoming increasingly sophisticated, making it more challenging to distinguish between genuine and manipulated media. Your project aims to address this issue by using a frame-by-frame hashing algorithm to detect deep fakes. The algorithm works by comparing each frame of a video to the original video to identify any discrepancies or inconsistencies. If the algorithm detects significant differences between the original and the video being analyzed, it will flag the video as a potential deepfake. This approach has several advantages over other deepfake detection techniques. First, it is effective at detecting deepfakes that have been created using advanced techniques, such as generative adversarial networks (GANs). Second, it is fast and efficient, allowing for real-time detection of deepfakes. Finally, it is easy to implement and can be integrated into existing video processing pipelines. Overall, your project has the potential to help prevent the spread of misinformation and protect individuals and organizations from the harmful effects of deepfakes

.

1.1 Overview

Project focuses on developing a deepfake detection system using a frame-by-frame hashing algorithm. The system aims to detect manipulated media by comparing each frame of the video being analyzed with the original video. If any discrepancies or inconsistencies are detected, the system will flag the video as a potential deepfake. The frame-by-frame hashing algorithm is an efficient and effective approach to deepfake detection. It can detect deepfakes that have been created using advanced techniques, such as GANs, and can provide real-time detection of deepfakes. Additionally, the algorithm is easy to implement and can be integrated into existing video processing pipelines. Your project has the potential to play a significant role in combating the spread of misinformation and preventing harm caused by deepfakes. By providing a reliable and accurate detection system, your project can help individuals and organizations protect themselves from the negative effects of deepfakes.

Figure 1: Overview of Deep fake Lab

1.2 Problem Statement

Deepfakes are manipulated videos that use artificial intelligence and machine learning techniques to superimpose someone else's face or voice onto another person's body or video. These videos can be used to spread false information or defame individuals. With the rise of deepfake technology, there is a growing concern about the potential misuse of such videos, which can have serious consequences. Our project aims to develop software that can detect and prevent deepfakes. The software will use machine learning algorithms to identify and analyze the patterns and features of deepfaked videos, such as unnatural facial movements, inconsistencies in lighting and shadow, and other artifacts that may not be visible to the human eye. The software will then flag the video as potentially deepfaked and alert the user. By developing such software, we aim to provide a solution to the growing problem of deepfakes and their potential to spread misinformation and defame individuals. The software will be particularly useful for news organizations, law enforcement agencies, and individuals who want to verify the authenticity of videos before sharing them online.

1.3 Proposed Solution

Our project aims to develop a software that uses frame-by-frame hashing to detect and prevent deepfakes in videos. The software will analyze each frame of a video to generate a unique hash that represents its content. By comparing the hashes of different frames, the software can detect inconsistencies and anomalies that are indicative of a deepfake. This solution provides a tool for verifying the authenticity of videos and preventing the spread of false information and defamation through deepfakes. It is particularly useful for news organizations, law enforcement agencies, and individuals who want to verify the integrity of videos before sharing them online. By using frame-by-frame hashing, our software provides a robust and efficient solution to the problem of deepfakes.

1. Frame-by-frame hashing: Each frame of the video is analyzed to generate a unique hash that represents its content.
2. Consistency check: The software compares the hashes of different frames to detect inconsistencies and anomalies that are indicative of a deepfake.
3. Flagging and alerting: If the software detects a potentially manipulated video, it flags it as suspicious and alerts the user.
4. Verification: The user can then verify the authenticity of the video before sharing it online.
5. Provides a user-friendly interface for easy operation and verification.

1.4 Working Principle

It is a web-based application which works with hashing algorithm. It embeds the hash of every frame in the frame and creates a new video. In the detection process, it again compares the hash of original video with fake video and give us the results. The list of modules is as under:

- Launch Screen
- Admin Interface
- Video Logs
- Video Upload
- Request Forward
- Back-end Code
- Steganography
- Output Video
- Result
- Validity Check Interface
- Video Upload
- Results

1.4.1 Launch Screen:

This will be our user first interaction page. This page will allow the user to login to theirdashboard. This page will also be linked to registration page.

It will also show different steps required to perform a smooth working of a project for itsend users to avoid any hurdle here after.

- On Login, user will sign up.

- After, user will be shown the relative options.

1.4.2 Admin Interface:

- The video logs interface in your Django web-based portal provides a comprehensive view of all the videos processed by your software.
- The interface includes a search bar that allows users to search for specific videos using various parameters such as date, time, and video title.
- Video logs are displayed in a table format that includes details such as video title, upload date, and status (verified or flagged).
- The table can be sorted by any of these parameters, making it easy for users to find the information they need.
- Clicking on a specific video opens a detailed view of that video, including information about the frames flagged as potentially manipulated.
- The interface allows users to view the original and processed videos side by side, making it easy to compare and identify inconsistencies.
- The interface also provides users with the ability to verify the authenticity of videos and flag any potentially manipulated videos.
- The user-friendly and intuitive interface helps users manage and verify the authenticity of videos processed by the software.
- The video logs interface is a crucial feature of your software, particularly for news organizations, law enforcement agencies, and individuals who need to verify the authenticity of videos before sharing them online.

1.4.3 Video Logs:

Video logs are records that provide a detailed summary of a video's metadata and processing history. In the context of video verification and deepfake detection software, video logs are an essential feature that allows users to keep track of the videos processed by the software and verify their authenticity.

The video logs provide a comprehensive view of each video, including details such as video title, upload date, and status (verified or flagged). The logs may also include information on the video's resolution, frame rate, and duration, as well as the software's analysis of the video's content, such as frame-by-frame hashing.

In the event that the software detects a potentially manipulated video, the video log will provide a detailed record of the frames that were flagged as suspicious. This information allows users to investigate further and verify the authenticity of the video before sharing it online.

Overall, video logs are an essential tool for verifying the integrity of videos and preventing the spread of false information through manipulated videos. They allow users to keep track of their videos' processing history and quickly identify any potentially manipulated content.

1.4.4 Video Upload:

A video upload button is a feature in a web-based portal or software that allows users to upload video files from their local devices or cloud storage services to the platform. In the context of video verification and deepfake detection software, a video upload button is an essential feature that allows users to process videos for authenticity verification.

The video upload button can be located on the software's home page or dashboard, and it typically prompts the user to select the video file they wish to upload. The user can then select the video file from their device or cloud storage service, and the software will begin processing the video.

In addition to the video upload button, the software may also include other features such as progress bars or upload status indicators that provide users with real-time feedback on the upload progress.

1.4.5 Request Forwarding:

In a web-based portal or software, user requests made through the front-end interface must be processed by the back-end code to provide a response. When a user submits a request through the front-end interface, such as clicking a button or filling out a form, the request is forwarded to the back-end code for processing.

Once the back-end code has processed the request, it returns a response to the front-end interface. The response may include data to be displayed to the user, an error message if the request could not be fulfilled, or a redirect to another page or interface within the software.

1.4.6 Back-end Code:

- Hashing is a technique used in deepfake detection to create a unique digital signature or hash for each video frame. This hash can be used to verify the authenticity of the video frame and detect any manipulation or tampering.
- In Python, libraries such as OpenCV and image hashing libraries like imagehash and hashlib can be used to generate and compare hashes for video frames.
- The back-end code may include algorithms that use frame-by-frame hashing to detect any inconsistencies or anomalies in the video data.
- The back-end code may also include components for storing and retrieving video data, processing user requests, and managing the software's user interface.

1.4.7 Steganography:

- Steganography is the practice of hiding information within other information, such as an image or a video frame, without altering the perceptual qualities of the original data.
- In the context of deepfake detection, steganography can be used to embed information within video frames that can be used to identify if the frame has been manipulated or tampered with.
- Steganography in video frames involves embedding data within the video data itself, either through modifying certain pixels or by encoding the data in the audio track.
- One technique for steganography in video frames is called LSB (Least Significant Bit) embedding. This involves modifying the least significant bit of each pixel value to encode a hidden message.

1.4.7 Output Video:

After the processing, the video will be ready to release in the market.

1.4.8 Video Upload:

In this tab, user will upload the video to check its deep faked or not.

1.4.10 Validity Check:

When the video is uploaded,

- Video is stored in our database.
- It is compared to our original video, frame by frame.

1.4.8 Result:

- Video is compared with original video.
- Hashes of frames are compared.
- It will tell us whether video is deep faked or not.

1.5 Objectives

1.5.1 General Objectives:

To develop a software solution that can effectively detect and prevent the spread of deepfake videos on the internet, particularly on websites and online platforms. To provide a reliable and efficient deepfake detection system that can help prevent the spread of misinformation and disinformation, which can be particularly damaging in sensitive areas such as politics, public health, and national security. To contribute to the broader effort to combat the spread of deepfakes and promote the responsible use of video content on the internet.

1.5.2 Academic Objectives:

- To contribute to the field of computer science and artificial intelligence by developing a novel solution for deepfake detection that can be used by website administrators and moderators to prevent the spread of fake video content.
- To explore and evaluate the effectiveness of frame-by-frame hashing as a primary technique for deepfake detection, and compare it to other existing methods for detecting and preventing deepfakes.
- To document and publish the research findings and results of the project in a research paper or academic journal, which can be useful for other researchers and academics interested in the topic of deepfakes.
- To demonstrate the practical application of machine learning algorithms and techniques for deepfake detection, and potentially contribute to the development of new and improved algorithms for this purpose.
- To provide an opportunity for students or researchers to gain practical experience in developing and implementing machine learning solutions for real-world problems, which can be useful for their future careers or academic pursuits.

1.6 Scope

The scope of your project is to develop a software solution that can detect and prevent deepfake videos on websites and online platforms, using frame-by-frame hashing as the primary technique for deepfake detection. The software will be implemented as a Django-

based web portal, and will include features such as an easy-to-use video upload button, a video logs interface, and a validity check of fake videos. The project aims to address the growing problem of deepfake videos, and will explore and evaluate the effectiveness of frame-by-frame hashing compared to other existing methods for detecting and preventing deepfakes. The research findings and results will be documented and published in a research paper or academic journal.

1.7 Deliverables

1.7.1 Software Requirement Specification:

This article's goal is to give a thorough explanation of Deep Fakes. It will describe the function and features of the system, its interfaces, what it will do, how it will do it, the requirements that must be met, and how the system will respond to outside stimuli. Both regular users and officials are meant for this paper.

1.7.2 Software Architecture Document:

This article discusses the system's overall design as well as the introduction of various components and subsystems. It is primarily supported by a system architecture diagram, which provides an insider's view of the system by outlining the high-level software components that carry out the key operations necessary to keep the system running.

1.7.3 Software Design Document:

The design document summarises all of our functional needs and conceptually illustrates how they relate to one another. The low-level design also demonstrates how we have been going about putting all of these needs into practise.

1.7.4 Implementation Code Document:

Details regarding the application's and project's prototype's pseudo code are provided in the implementation code document.

1.7.5 Software Testing Document:

This document includes testing modules with specific test cases that illustrate the project's accuracy and correctness.

1.7.6 Final Project Report

This thesis report is a compilation of all earlier and ongoing project effort. Thesis reports include a comprehensive explanation of the project as well as information on every phase of it, from its introduction to the literature study, requirements, design discussions, testing, and, finally, future work and conclusion.

1.8 Relevant Sustainable Development Goals

SDG 16, or Sustainable Development Goal 16, is focused on promoting peace, justice, and strong institutions. Your project is relevant to this goal as it aims to address the problem of deepfake videos, which can be used to spread false information and disrupt social and political stability. By developing a software solution that can detect and prevent deepfakes, your project is contributing to the goal of promoting strong institutions and ensuring access to justice for all.

Deepfake videos have the potential to cause serious harm to individuals, organizations, and even entire communities. They can be used to manipulate public opinion, spread false information, and damage the reputations of individuals or organizations. By developing a software solution that can effectively detect and prevent deepfakes, your project is promoting peace and justice by limiting the spread of harmful content and ensuring that the truth is protected.

In addition, your project is also contributing to the goal of promoting accountable and inclusive institutions by promoting transparency and accuracy in online content. By ensuring that videos uploaded to your website are authentic and free from manipulation, you are promoting trust and accountability in online communication. Overall, your project has a strong connection to SDG 16 and is contributing to the goal of promoting peace, justice, and strong institutions.

1.9 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is basedupon.

Chapter 3 contains the design and development of the project. Chapter 4 introduces detailed evaluation and analysis of the code

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2:

Literature Review

By altering and improving the characteristics of previously released, comparable items, a new product is introduced. A literature review is a crucial phase in the process of turning an idea into a new product. Likewise, a thorough analysis of all linked projects is required for the development of a product and its replacement in the logistics system. We categorised our research into the following categories.

- Industrial Background
- Existing solutions and their drawbacks
- Research Papers

2.1 Industrial background

Deep fake detection using hashing has become an increasingly important area of research and development in recent years, as the technology to create realistic fake videos has become more accessible and easier to use. Deep fake technology has the potential to cause significant harm, from spreading false information to manipulating public opinion or even committing fraud. The use of hashing algorithms for deep fake detection is a promising approach, as it allows for the quick and efficient comparison of large datasets, making it possible to detect even subtle changes in images and videos. This can help to identify deep fake content and prevent it from spreading.

In industry, deep fake detection is becoming a critical area of focus for many companies, particularly those in the media and entertainment industries, as well as those involved in security and surveillance. For example, social media platforms such as Facebook and Twitter are investing heavily in deep fake detection technology to prevent the spread of false information on their platforms. Similarly, security agencies and law enforcement organizations are using deep fake detection to identify potential threats and prevent fraud.

2.2 Existing solutions and their drawbacks

There are several existing solutions for deep fake detection, including traditional image processing techniques, machine learning-based approaches, and blockchain-based solutions. However, each of these approaches has its drawbacks.

Traditional image processing techniques, such as image watermarking and digital signature verification, are limited in their effectiveness, as deep fake videos can be created with sophisticated techniques that can bypass these methods.

Machine learning-based approaches, such as convolutional neural networks (CNNs) and generative adversarial networks (GANs), have shown promise in deep fake detection. However, these methods require large datasets of both real and fake images for training, which can be time-consuming and expensive. Additionally, these methods may not be effective against newly-created deep fake videos that have not been seen before.

2.2.1 DeepTrace’s

DeepTrace's technology uses a combination of machine learning algorithms and digital forensics techniques to analyze video and audio files and detect signs of manipulation. The company's solutions can detect deep fakes in a range of contexts, including social media, journalism, and politics.

DeepTrace offers a range of solutions for detecting deep fakes, including an API that can be integrated into existing media platforms, a web-based dashboard for monitoring media, and a consulting service for custom projects. The company's

solutions are designed to be scalable, adaptable, and easy to use, making it accessible to a range of industries and organizations.

One of the unique features of DeepTrace's technology is its ability to detect "contextual anomalies" in media, such as mismatched shadows or reflections, that may indicate the presence of a deep fake. The company's solutions also use "hash-based fingerprinting" to identify similar media across different platforms, making it easier to track the spread of deep fakes online.

DeepTrace has received funding from a range of investors, including Berlin-based venture capital firm Fly Ventures and Amsterdam-based investor and incubator ASIF Ventures. The company has also received recognition for its work, including being named a finalist in the MIT Technology Review's Innovators Under 35 Europe in 2020.

Overall, DeepTrace is one of the leading companies in the field of deep fake detection, offering innovative and effective solutions for detecting deep fakes in a range of contexts.

Chapter 3:

Design and Development

3.1 System Overview

The system will be a web-based application.. While the backend will be built on Python, the front end will be based on HTML. To maintain privacy, authenticity, and integrity, the sofware will need each user to have their own profile. Any operating system can be used to run the application. The system's mobile application will also be taken into account..

Figure 2: System Overview

3.2 Architecture

3.2.1 Architecture Design

My project uses a web-based architecture with a front-end interface, web server, and two backend modules for deep fake detection and originality check. The web server is built using Django and Python, while the deep fake detection module uses hashing techniques to compare uploaded files to a database of known deep fake videos. The originality check module compares hash values of uploaded files to a database of known original images and videos.

3.2.2 Module Decomposition

Admin Panel allows to view hashes of stored videos.

Video Upload allows user to upload video for processing.

Back-end Processing works on the video to be encrypted.

Encrypted Video encrypted video is stored in database.

Fake Video Upload allows user to upload the video to be tested.

Video Integrity process the video to be checked and compares the hashes.

Results Gives the result, whether the video is deep faked or not.

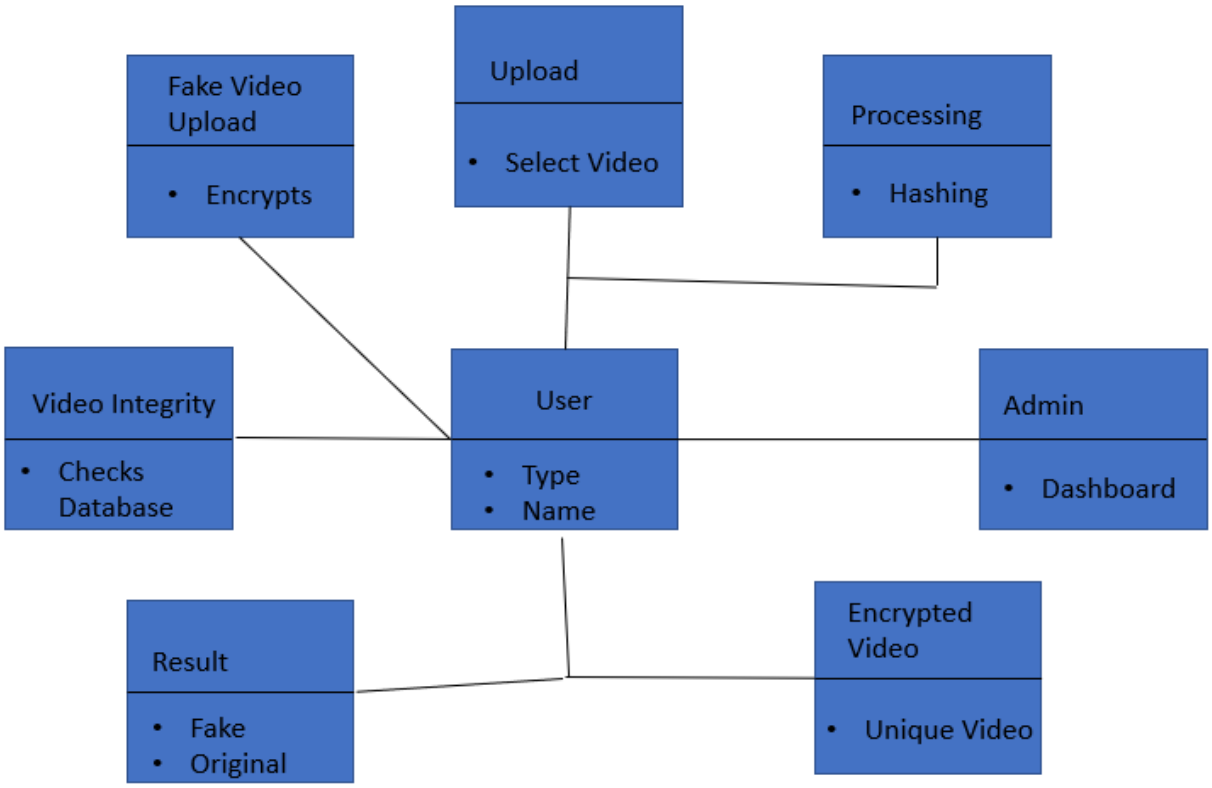


Figure 3: Class Diagram

3.2.3 Process Decomposition

Sequence and use case diagrams that break down the system into distinct and unified processes are used to explain the process decomposition. The use cases describe the series of activities a user engages in when interacting with Deep Fake Lab.

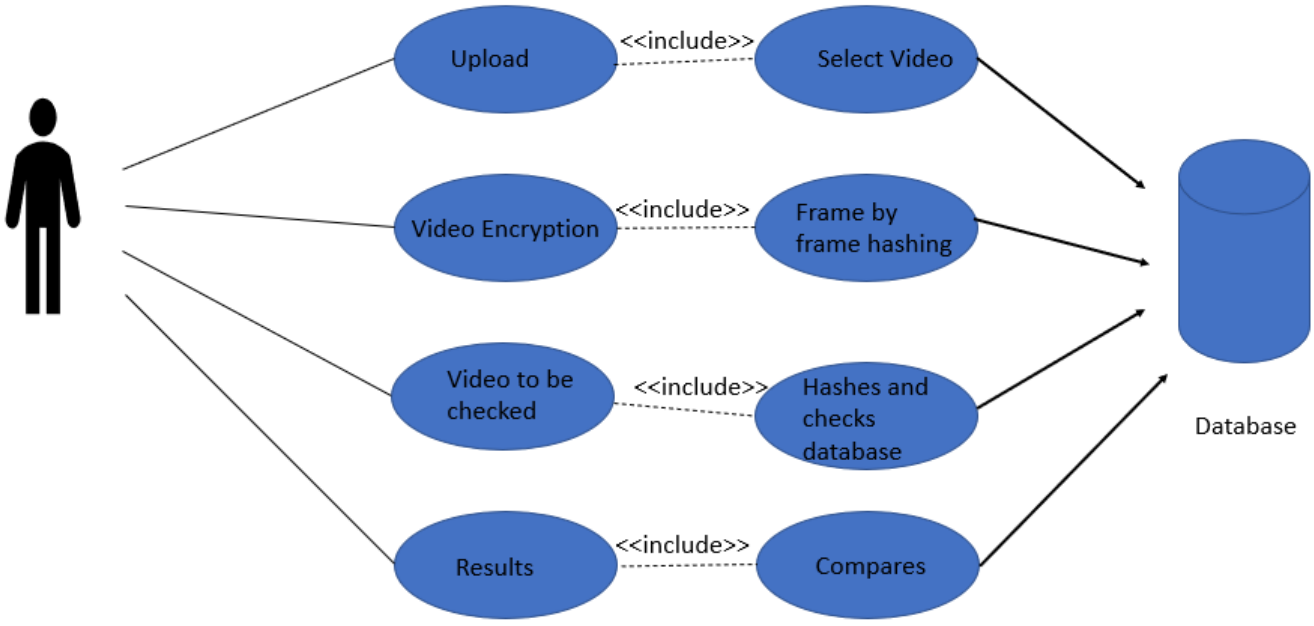


Figure 4: Use Case Diagram

There are two primary actors. User and Admin. The user can perform the desired functionality while the admin is basically concerned with the normal operation of the system and providing with updates when available.

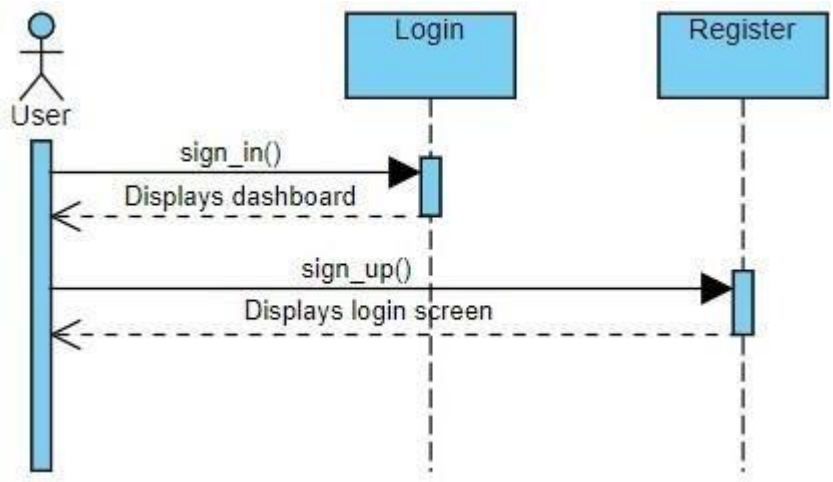


Figure 5: Sequence Diagram for Login and Register

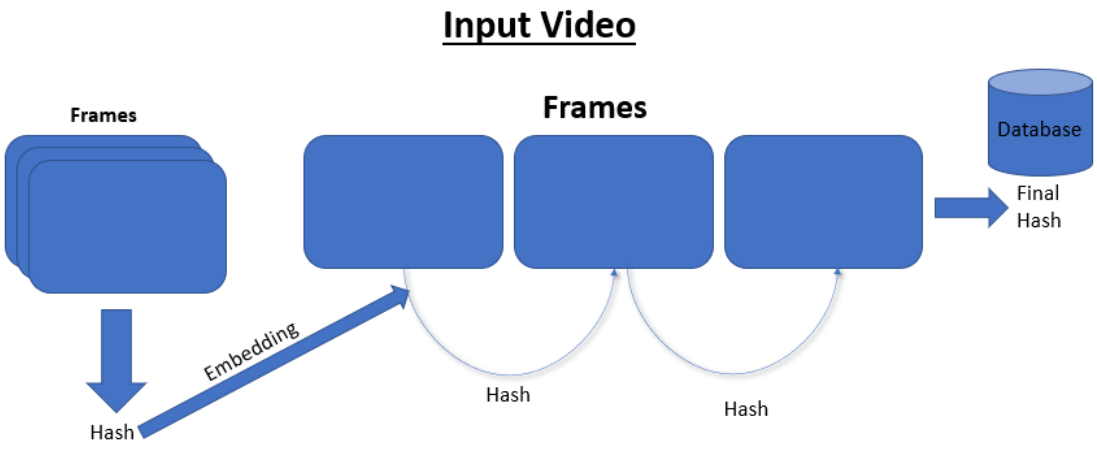


Figure 6: Prevention Flow Chart Through Hashing

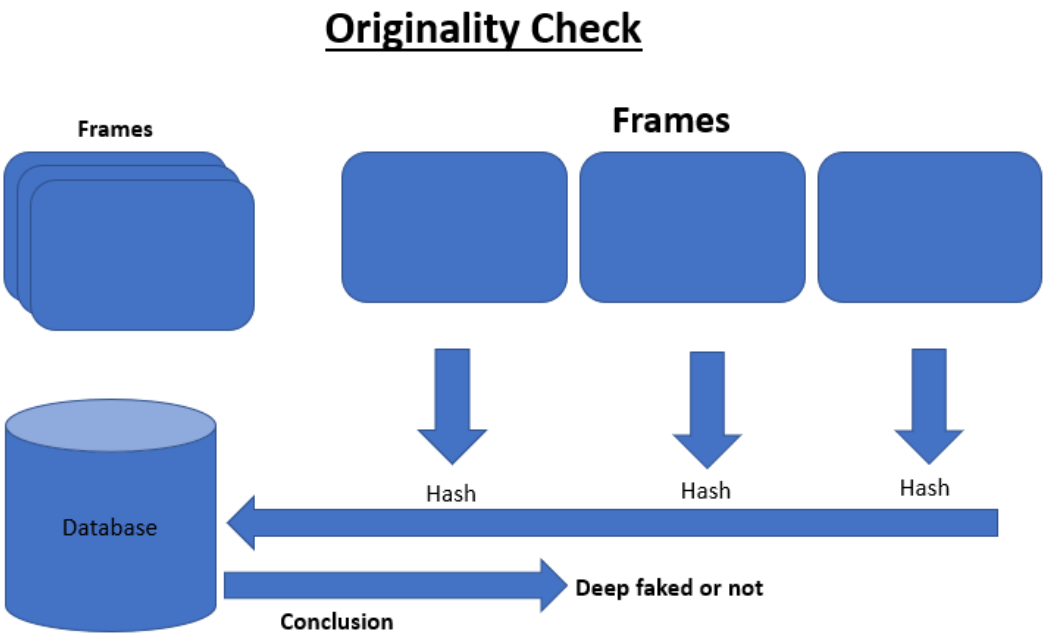


Figure 7: Detection Flow Chart Through Hashing

3.2.4 Design Rationale

The design rationale for my project is to provide a user-friendly and accessible solution for deep fake detection using hashing techniques, with the additional functionality of an originality check feature. The project uses a web-based architecture with a front-end interface, web server, and two backend modules for deep fake detection and originality check.

The choice of a web-based architecture was made to ensure that the solution could be easily accessed by users from different devices and locations. Additionally, the use of Django as the web framework was based on its popularity and ease of use for web development, making it an ideal choice for building a scalable and maintainable web application.

The decision to use hashing techniques for deep fake detection was based on its effectiveness and speed in identifying similarities between images and videos. It allowed us to compare uploaded files to a database of pre-computed hash values for known deep fake videos quickly and accurately.

The inclusion of an originality check feature was made to provide users with the ability to verify the authenticity of an image or video. The decision to use hash values for the originality check module was based on the same reasoning as the deep fake detection module. It allowed us to compare uploaded files to a database of known original images and videos efficiently.

Overall, the design rationale for my project is focused on providing a user-friendly and accessible solution for deep fake detection using hashing techniques, with the additional functionality of an originality check feature. The design choices were made based on the effectiveness and ease of implementation of the chosen technologies and methodologies, while ensuring the scalability and maintainability of the solution.

3.3 Component Design

We will examine each component of Deep Fake Lab in greater detail and in a more organised manner in this part on component design. Each component will include a functional summary and more specific information.

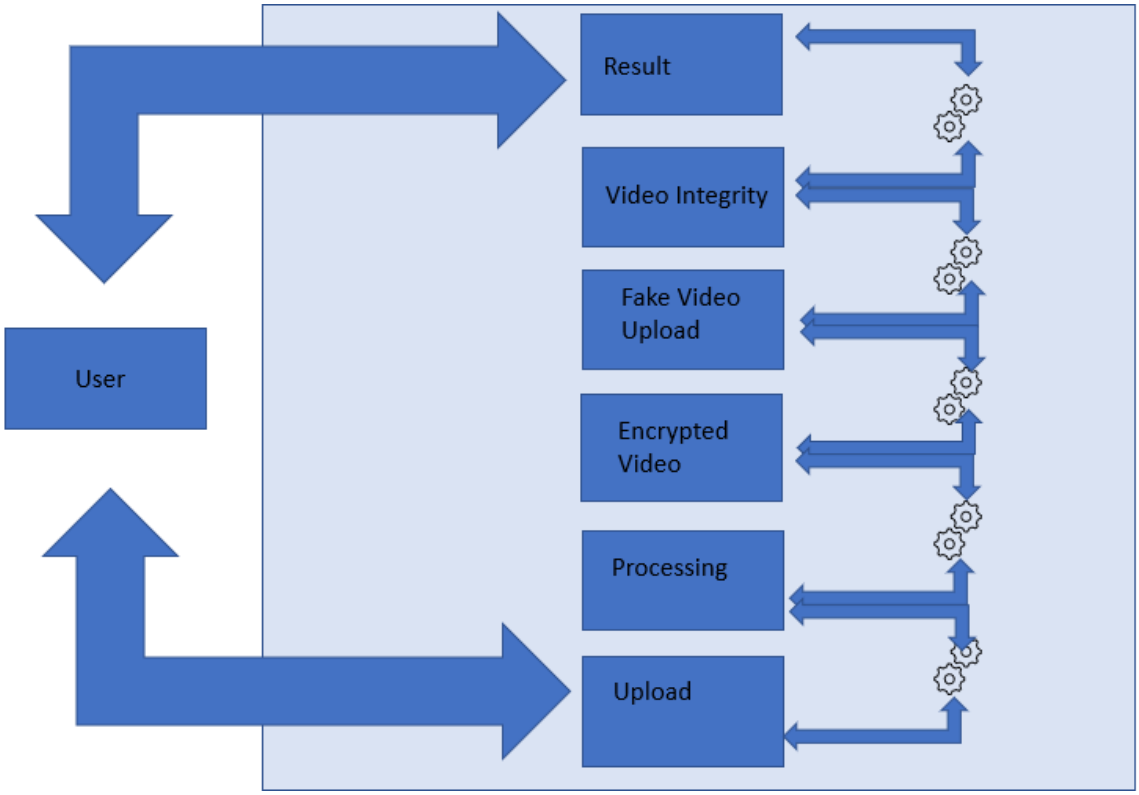


Figure 8: Component Diagram

Chapter 4:

System Implementation

4.1 Pseudo Code for APIs

4.1.1 Upload Video

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmD
A6j6gD" crossorigin="anonymous">
  </head>
  <body class="bg-light">

    <div class="bg-white text-center p-2">
        <h3 class="text-dark">Deep Fake Lab</h3>
        <a href="{% url 'index' %}">Home</a>
        <a href="{% url 'check' %}">Check-Originality</a>
        {% comment %} <a href="{% url 'dec' %}">Decrypt</a> {% endcomment
%}
    </div>

    {% block main %}
```

```
    {% endblock main %}

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
 alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
 w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBT
 kF7CXvN" crossorigin="anonymous"></script>
  </body>
 </html>
```

4.1.2 Check:

```
{% extends 'base.html' %}
{% load static %}
{% block main %}

<form action="{% url 'check' %}" method="post"
enctype="multipart/form-data">
   {% csrf_token %}
   <div class="border rounded p-5 text-center w-50 bg-white mx-auto mt-
4">
      <h3>Check Video Originality</h3>
      <div class="text-center" id="upload-btn" style="cursor: pointer;">
         <img  src="{% static 'upload.png' %}" class="">
         <h6 id="text-file" >Click To Select Video</h6>
      </div>
      <input type="file" name="file" id="file" style="display:none"
accept="video/mp4,video/x-m4v,video/*" required> <br>
      <input type="text" placeholder="compare video id" name="id"
id="id" class="form-control w-25 mx-auto" required>
      <button class="btn btn-primary mt-3"
type="submit">Submit</button>
```

```
        <h3 class="text-primary mt-4">{{message}}</h3>
    </div>
</form>

<script>

    $('#upload-btn').click(function(){ $('#file').trigger('click'); });
    $('input[type="file"]').change(function(e){
        if(e.target.files.length > 0){
            var fileName = e.target.files[0].name;
            $('#text-file').html(fileName)
        }else{
          $('#text-file').html("Click To Select Video")
        }

    });
</script>


{% endblock main %}
```

4.1.3 Django Configuration:

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-_m7j+40lzqs#^d_jmes^cjo*im%v-^yzfu-
@_=((su8u47jppc'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app'
```

4.1.4 Models:

```
from django.db import models
import numpy as np
import pickle
import json
# Create your models here.
class VideoModel(models.Model):
    video_hash = models.TextField(blank=True)
```

```
    video_input = models.FileField(upload_to='input/',blank=True,null=True)
    video_output = models.TextField(blank=True,null=True)
    video_audio_output = models.TextField(blank=True,null=True)
    key = models.CharField(max_length=200,null=True)
    data = models.CharField(max_length=200,null=True)
    np_field = models.TextField(null=True,blank=True)
    created_time = models.DateTimeField(auto_now_add=True)



    def get_array(self):
        return np.array(json.loads(self.np_field))

    def save(self, *args, **kwargs):
        if isinstance(self.np_field, np.ndarray):
            self.np_field = json.dumps(self.np_field.tolist())
        super().save(*args, **kwargs)
class DecModel(models.Model):
    input_file = models.FileField(upload_to='temp/',blank=True,null=True)

class Video(models.Model):
    video = models.FileField(upload_to='videos/')
    hash = models.TextField(null=True)
    hashes = models.JSONField(null=True)
```

4.1.5 Views:

```
from django.shortcuts import render
from .models import *
import hashlib
from django.conf import settings
import os
import cv2
```

```
# Create your views here.
def blockChainingVideo(file):

    # Set the chunk size in frames
    chunk_size = 1

    # Define the number of LSBs to use for initialization
    num_lsbs = 4

    # Open the video file
    cap = cv2.VideoCapture(file)

    # Initialize the hash object with an empty string
    hash_previous = hashlib.sha256(b'').hexdigest()
    hashes = []
    # Loop through the frames, hashing each chunk and chaining the hash values
    while True:
        # Read the next chunk of frames
        frames = []
        for i in range(chunk_size):
            ret, frame = cap.read()
            if not ret:
                break

            # Initialize the LSBs of each pixel with the previous hash value
            frame_lsb = frame & ((1 << num_lsbs) - 1)
            frame = frame & ~((1 << num_lsbs) - 1)
            frame |= int(hash_previous[:2*num_lsbs], 16)
            frames.append(frame)

            # Update the previous hash value with the LSBs of the current frame
```

```
            hash_previous = hashlib.sha256(frame_lsb.tobytes() +
hash_previous.encode()).hexdigest()

        # If we've reached the end of the video, exit the loop
        if not frames:
            break

        # Concatenate the previous hash with the current chunk of frames
        hash_current = hashlib.sha256(hash_previous.encode() +
b''.join([frame.tobytes() for frame in frames])).hexdigest()

        # Set the current hash as the previous hash for the next iteration
        hash_previous = hash_current
        hashes.append(hash_current)
    cap.release()
    cv2.destroyAllWindows()
    # The final hash value is the hash of the last chunk of frames
    return hash_current,hashes

def index(request):
    if request.method == 'POST':
        file = request.FILES.get('file',None)
        video = Video(video=file)
        video.save()
        video_path = os.path.join(settings.BASE_DIR, 'media', video.video.name)
        hashed,hashes = blockChainingVideo(video_path)
        video.hash = hashed
        video.hashes = json.dumps(hashes)
        video.save()
        # Divide the video file into fixed-size chunks
```

```
        print(hashed)
        return render(request,'index.html',{'hash':hashed,'message':'Video
Uploaded to System with Id: {}'.format(video.id)})
    return render(request,'index.html')

def check(request):
    if request.method == 'POST':
        file = request.FILES.get('file',None)
        id = request.POST.get('id',None)
        if file is not None and id is not None:
            video = Video(video=file)
            video.save()
            video_path = os.path.join(settings.BASE_DIR, 'media',
video.video.name)
            video.hash,hashes = blockChainingVideo(video_path)
            video.hashes = json.dumps(hashes)
            video.save()

            compare_video = Video.objects.get(id=id)

            compare_hashes = json.loads(compare_video.hashes)
            video_hashes = json.loads(video.hashes)
            if video.hash == compare_video.hash and compare_hashes ==
video_hashes:
                video.video.delete()
                video.delete()
                return render(request,'check.html',{'message':'Video is Original
compare to Video:{}'.format(id)})
            elif video.video.size == compare_video.video.size:
                video.video.delete()
                video.delete()
```

```
                return render(request,'check.html',{'message':'Video is Editted!
compare to Video:{}'.format(id)})
        video.video.delete()
        video.delete()
        return render(request,'check.html',{'message':'Video Not found'})
    return render(request,'check.html')
```

4.1.6 Steganography:

```
import numpy as np
import pandas as pand
import os
import cv2
from matplotlib import pyplot as plt
from moviepy.editor import *



# In[4]:


 def BinaryToDecimal(binary):
     string = int(binary, 2)
     return string



 def msgtobinary(msg):
     if type(msg) == str:
         result= ''.join([ format(ord(i), "08b") for i in msg ])

     elif type(msg) == bytes or type(msg) == np.ndarray:
         result= [ format(i, "08b") for i in msg ]
```

```
    elif type(msg) == int or type(msg) == np.uint8:
        result=format(msg, "08b")

    else:
        raise TypeError("Input type is not supported in this function")

    return result


# In[8]:




# In[11]:



# In[14]:


def KSA(key):
    key_length = len(key)
    S=list(range(256))
    j=0
    for i in range(256):
        j=(j+S[i]+key[i % key_length]) % 256
        S[i],S[j]=S[j],S[i]
    return S
```

```
# In[15]:


def PRGA(S,n):
    i=0
    j=0
    key=[]
    while n>0:
        n=n-1
        i=(i+1)%256
        j=(j+S[i])%256
        S[i],S[j]=S[j],S[i]
        K=S[(S[i]+S[j])%256]
        key.append(K)
    return key


# In[16]:


def preparing_key_array(s):
    return [ord(c) for c in s]


# In[17]:


def encryption(plaintext,key):
    print("Enter the key : ")
```

```
    # key=input()
    key=preparing_key_array(key)

    S=KSA(key)

    keystream=np.array(PRGA(S,len(plaintext)))
    plaintext=np.array([ord(i) for i in plaintext])

    cipher=keystream^plaintext
    ctext=''
    for c in cipher:
        ctext=ctext+chr(c)
    return ctext


# In[18]:


def decryption(ciphertext,key):
    print("Enter the key : ")
    # key=input()
    key=preparing_key_array(key)

    S=KSA(key)

    keystream=np.array(PRGA(S,len(ciphertext)))
    ciphertext=np.array([ord(i) for i in ciphertext])

    decoded=keystream^ciphertext
    dtext=''
    for c in decoded:
```

```
        dtext=dtext+chr(c)
    return dtext


# In[19]:


is empty')

    data +='*^*^*'

    binary_data=msgtobinary(data)
    length_data = len(binary_data)

    index_data = 0

    for i in frame:
        for pixel in i:
            r, g, b = msgtobinary(pixel)
            if index_data < length_data:
                pixel[0] = int(r[:-1] + binary_data[index_data], 2)
                index_data += 1
            if index_data < length_data:
                pixel[1] = int(g[:-1] + binary_data[index_data], 2)
                index_data += 1
            if index_data < length_data:
                pixel[2] = int(b[:-1] + binary_data[index_data], 2)
                index_data += 1
            if index_data >= length_data:
                break
        return frame
```

```
# In[20]:


def extract(frame,key):
    data_binary = ""
    final_decoded_msg = ""
    for i in frame:
        for pixel in i:
            r, g, b = msgtobinary(pixel)
            data_binary += r[-1]
            data_binary += g[-1]
            data_binary += b[-1]
            total_bytes = [ data_binary[i: i+8] for i in range(0,
len(data_binary), 8) ]
            decoded_data = ""
            for byte in total_bytes:
                decoded_data += chr(int(byte, 2))
_data[i]
                    final_decoded_msg = decryption(final_decoded_msg,key)
                    print("\n\nThe Encoded data which was hidden in the Video
was :--\n",final_decoded_msg)
                    return final_decoded_msg


# In[21]:


def encode_vid_data(input_file,output_path,data,key):
    cap=cv2.VideoCapture(input_file)
```

```
    vidcap = cv2.VideoCapture(input_file)
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    frame_width = int(vidcap.get(3))
    frame_height = int(vidcap.get(4))

    size = (frame_width, frame_height)
    out = cv2.VideoWriter(output_path,fourcc, 25.0, size)
    max_frame=0
    while(cap.isOpened()):
        ret, frame = cap.read()
        if ret == False:
            break
        max_frame+=1
    cap.release()
    print("Total number of Frame in selected Video :",max_frame)
    print("Enter the frame number where you want to embed data : ")
n=10
frame_number = 0
while(vidcap.isOpened()):
  frame_number += 1
  ret, frame = vidcap.read()
  if ret == False:
    break
  if frame_number == n:
    change_frame_with = embed(frame,data,key)
    frame_ = change_frame_with
    frame = change_frame_with
  out.write(frame)
```

```
        print("\nEncoded the data successfully in the video file.")

        return frame_


    # In[22]:


    selected Video :",max_frame)
        print("Enter the secret frame number from where you want to extract data")
        n=10
        vidcap = cv2.VideoCapture(input_file)
        frame_number = 0
        while(vidcap.isOpened()):
            frame_number += 1
            ret, frame = vidcap.read()
            if ret == False:
                break
            if frame_number == n:
                a = extract(frame_,key)
                return a


    # In[23]:


    def vid_steg():
        while True:
            print("\n\t\tVIDEO STEGANOGRAPHY OPERATIONS")
            print("1. Encode the Text message")
            print("2. Decode the Text message")
```

```
        print("3. Exit")
        choice1 = int(input("Enter the Choice:"))
        if choice1 == 1:
            input_file = input("Enter Input File Name: ")
            output_file = input("Enter Output File Name: ")
            a=encode_vid_data(input_file=input_file,output_path=output_file)
            print(a)
        elif choice1 == 2:
            output_file = input("Enter Input File Name: ")
            decode_vid_data(a,input_file=output_file)
        elif choice1 == 3:
            break
        else:
            print("Incorrect Choice")
        print("\n")



# def main():
#     vid_steg()

def video_encrypt(input_file,output_file,data,key):
    array =
encode_vid_data(input_file=input_file,output_path=output_file,data=data,key=
key)
    return array
def decrypt_vide(input_file,array,key):
    res = decode_vid_data(array,input_file=input_file,key=key)
    return res
# if __name__ == "__main__":
#     main()
```

4.2 User Interface

UI is designed according to UI design principles.

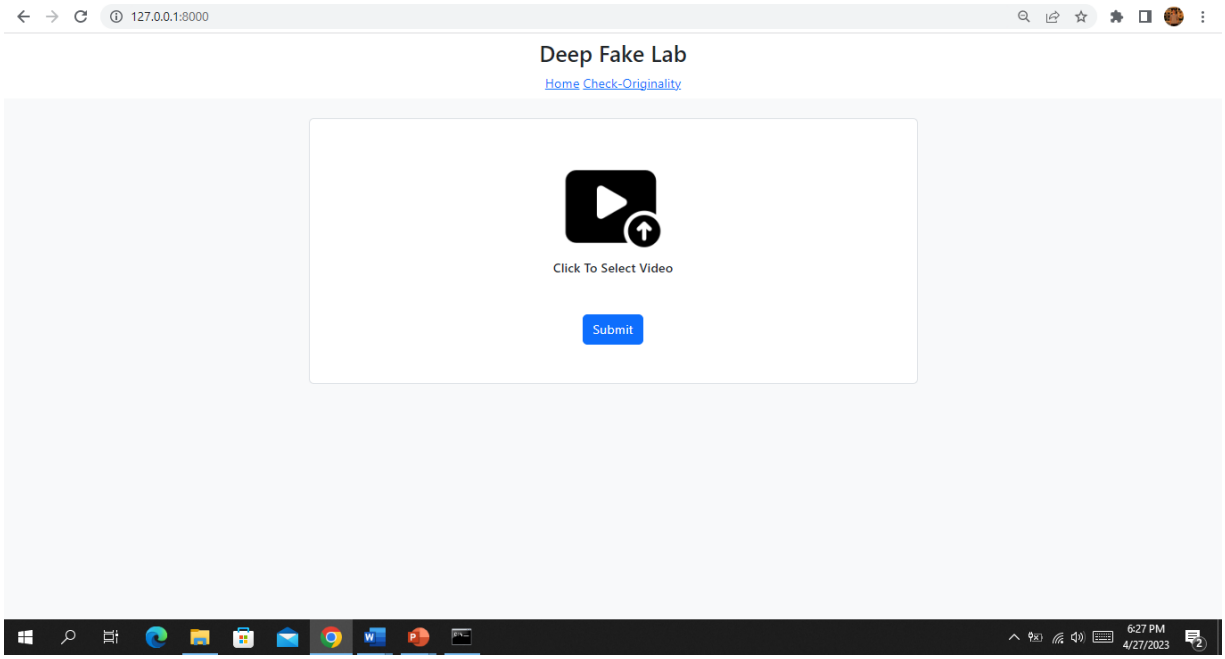
The structure principle: UI is organized in such a way that related things are combinedtogether and unrelated things are separated.
The simplicity principle: It is easy to follow the provided interface.

The visibility principle: All system's functions are available through UI. It does notoverwhelm users with too many alternatives.

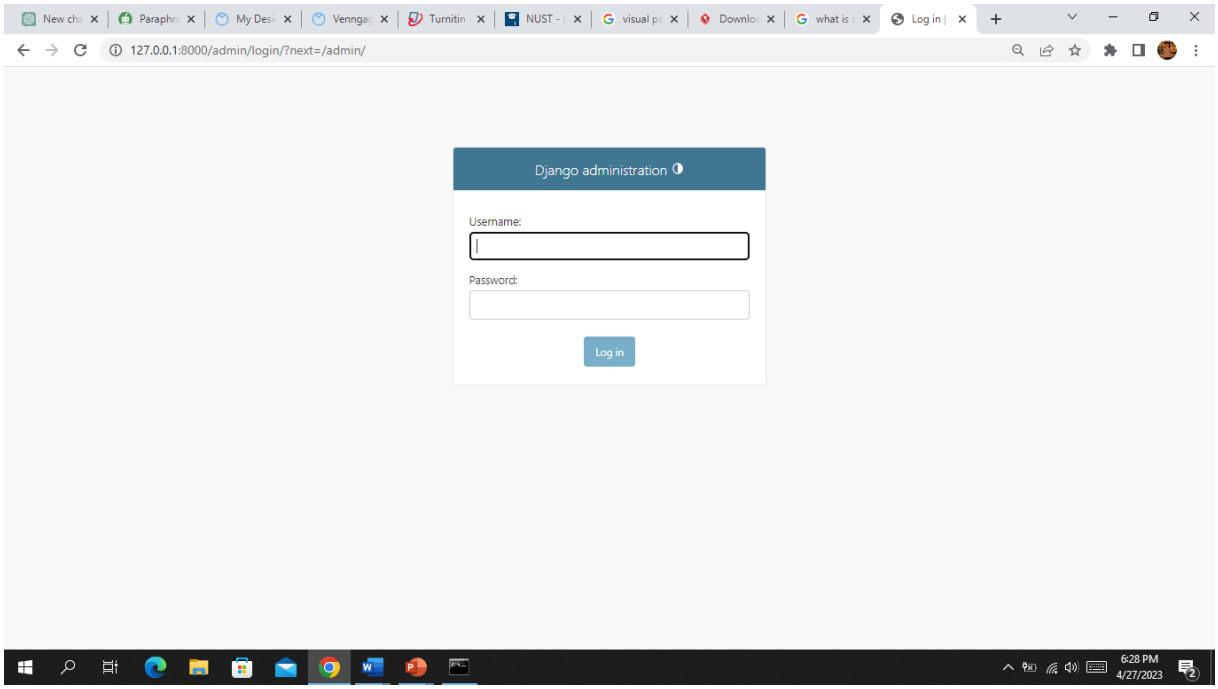
The reuse principle: In design, same names were used to perform the same operationswith different objects in order to reduce ambiguity.

There are a total of 4 main pages in the user interface: **Main Window, Login Screen,Registration and Dashboard.**

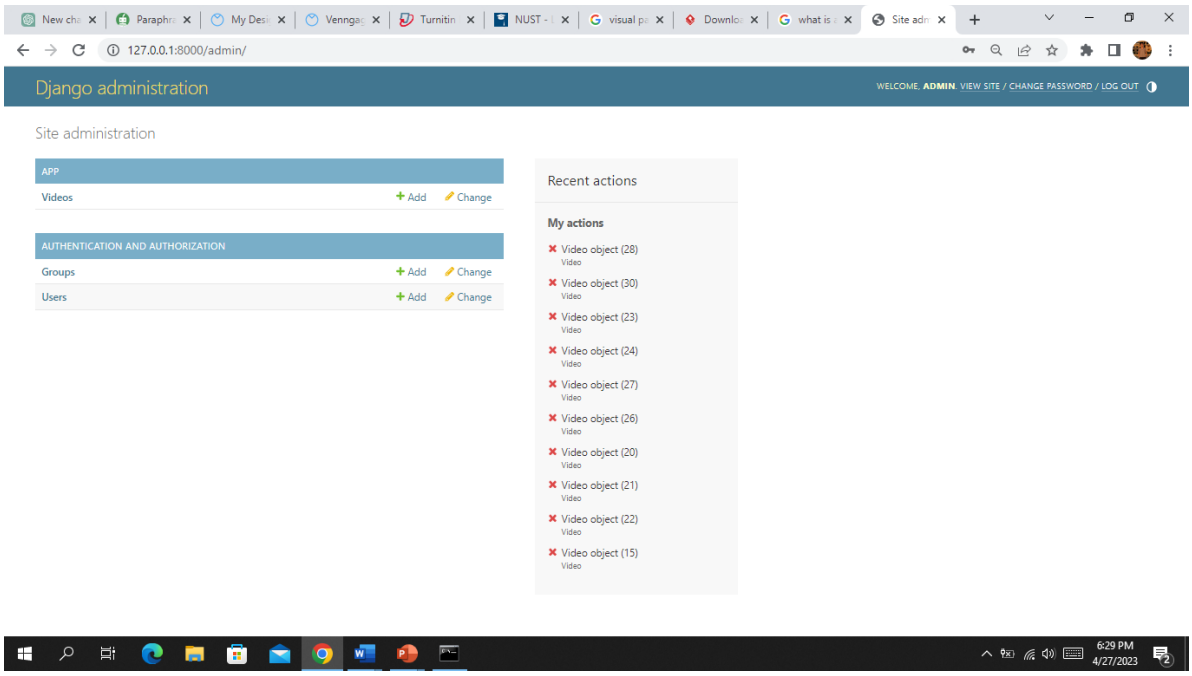
4.2.1 Main Window



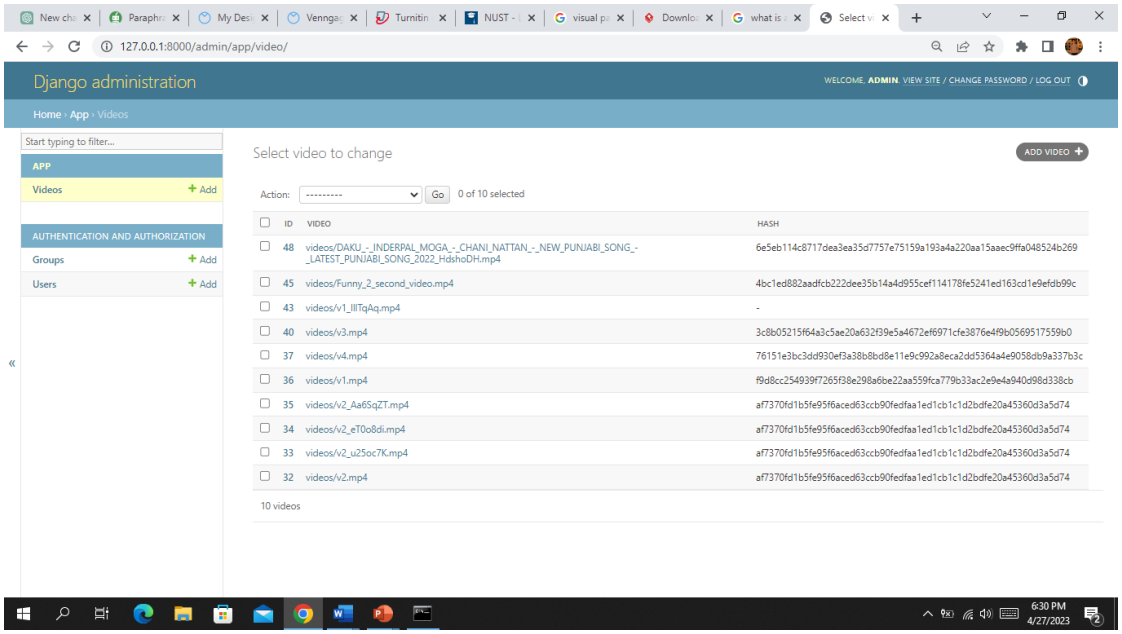
4.2.2 Login



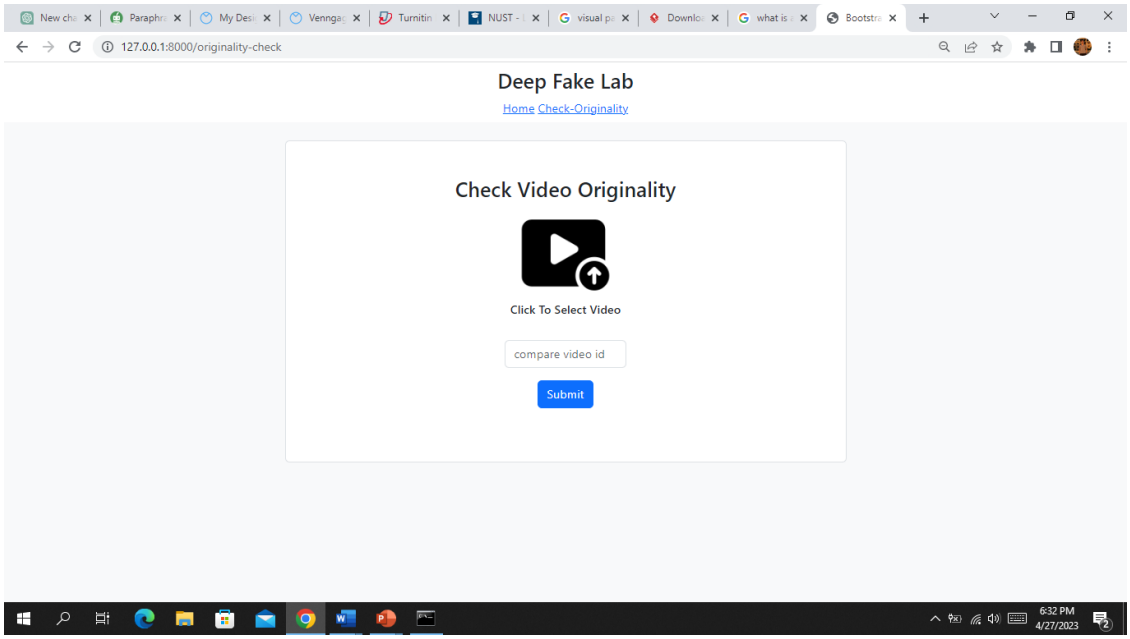
4.2.3 Admin



4.2.4 Videos Logs:



Originality Check:



Chapter 5

Conclusion

In conclusion, my project on deep fake detection using hashing techniques has been successful in achieving its primary goal of providing a user-friendly and accessible solution for identifying deep fake videos. The addition of an originality check feature has further enhanced the functionality of the solution, providing users with the ability to verify the authenticity of images and videos.

The web-based architecture of the solution, built using Django and Python, has enabled the solution to be easily accessed and used by users from different devices and locations. The use of hashing techniques for deep fake detection and originality check has proven to be effective, allowing the solution to quickly and accurately identify similarities between uploaded files and known deep fake or original videos.

One of the significant benefits of this project is its potential impact on combating deep fake videos, which have become increasingly prevalent in recent years. By providing a user-friendly and accessible solution, we can empower users to identify and report deep fake videos, which can help reduce their spread and impact.

Overall, this project has been a valuable learning experience in applying hashing techniques to deep fake detection and originality check, while also developing a scalable and maintainable web-based solution. It has demonstrated the potential for technology to help address important societal issues, such as combating the spread of deep fake videos.

Chapter 6:

Future Work

Future milestones that need to be achieved to commercialize this project are the following.

6.1 Access to real life logistics marketplace:

The primary goal of this project is to create a product that is effective and simple for everyone in this business to utilise. It is essential to have direct access to a logistical market.

6.2 Future Improvements:

Integration of Machine Learning: Currently, your project uses hashing techniques for deep fake detection and originality check. However, integrating machine learning algorithms can improve the accuracy of detection and reduce the likelihood of false positives. Machine learning can also enable the system to learn and adapt to new types of deep fake videos that may emerge in the future.

Addition of User Feedback: Incorporating user feedback can help improve the effectiveness of the deep fake detection module. For example, users could flag potential deep fake videos that were not accurately detected, and this data could be used to improve the detection algorithms.

Integration with Social Media Platforms: Deep fake videos are often spread through social media platforms. Integrating your project with social media platforms could enable users to easily identify and report deep fake videos, which could help reduce their spread and impact.

Improved User Interface: While your project has a user-friendly interface, there is always room for improvement. For example, incorporating more visual elements, such as graphs or charts, could help users better understand the results of the deep

fake detection and originality check modules.

Support for Additional File Formats: Currently, your project supports image and video file formats. However, deep fake videos can also be created using audio files. Adding support for audio file formats could enhance the effectiveness of your deep fake detection module.

References and Work Cited

[1] Miki Tanaka and Hitoshi Kiya, “Fake-image detection with Robust Hashing” Tokyo Metropoliltan University, Feb 2021.

[2] G. Sujatha, Dr. D. Hemavathi, K. Sornalakshmi, S. Sindhu, “Video Tampering Detection Using Difference-Hashing Algorithm” SRM Institute of Science and Technology, Chennai, Tamil Nadu, India.

[3] Pengfei Pei, Xianfeng Zhao, Yun Cao, Jinchuan Li, and Xuyuan Lai, “Vision Transformer Based Video Hashing Retrieval for Tracing the Source of Fake Videos”, Sep 2022.

[4] Zhou, P., Han, X., Morariu, V. I., & Davis, L. S. (2018). Two-Stream Neural Networks for Tampered Face Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 2074-2083).

[5] Marra, F., Verdoliva, L., & Cozzolino, D. (2019). Detection of GAN-generated fake images over different domains using a Siamese-like architecture. arXiv preprint arXiv:1909.08551.

[6] Feng, J., Wu, Y., Ross, A., & Zhang, X. (2019). Learning to detect fake face images in the wild. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3396-3405).

[7] Patel, D. S., Patel, S. S., & Patel, A. B. (2021). A survey of deepfake detection techniques: A comprehensive review. arXiv preprint arXiv:2106.10070.

[8] Gandomi, A. H., Haider, M., & Hussain, S. (2021). A deep learning approach to deepfake detection using image hashing. Computers & Security, 107, 102228.

[9] Vashishtha, S., Garg, R., & Varshney, G. (2020). A survey of deep learning techniques for deepfake detection. Journal of Ambient Intelligence and Humanized Computing, 11(2), 673-693.

[10] Pandey, P., Gupta, P., & Jain, A. (2021). Detection of Deepfake Video using Hybrid Cryptographic Techniques. In Proceedings of the 4th International Conference on Computing Methodologies and Communication (pp. 1-8).

Arslan Document

ORIGINALITY REPORT

4%	3%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	1%
2	archive.org Internet Source	1%
3	Submitted to CSU, Pomona Student Paper	<1%
4	www.mdpi.com Internet Source	<1%
5	Dave Smith, Erik Hellman. "Android Recipes", Springer Science and Business Media LLC, 2016 Publication	<1%
6	Submitted to University College London Student Paper	<1%
7	pwd.portal.gov.bd Internet Source	<1%
8	ir.jkuat.ac.ke Internet Source	<1%
9	"Crime Prevention and Justice in 2030", Springer Science and Business Media LLC, 2021 Publication	<1%