Deep Fake Lab

By:

GC Muhammad Fahad Asif

GC Ahmed Abdullah

GC Ibad ur Rehman

GC Ali Hassan Awan

Supervised by:

Col (R) Dr. Imran Touqir (PhD)

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 2024

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 Muhammad Fahad Asif, Ahmed Abdullah, Ibad ur Rehman, Ali Hassan Awan 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 Muhammad Fahad Asif
GC Ibad ur Rehman
GC Ali Hassan Awan
GC Ahmed Abdullah

Approved by Supervisor

Signature: _____

Name of Supervisor: Col (R) Dr. Imran Touqir

Dated: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of 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 supervisors, Col (R) Dr. Imran Touqir

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:

_____

Col (R) Dr. Imran Touqir

Name & Signatures of Students:

_____

GC Muhammad Fahad Asif

00000359294

_____

GC Ahmed Abdullah

00000359317

GC Ibad ur Rehman

00000359301

GC Ali Hassan Awan

00000359290

ABSTRACT

The widespread use of deepfake technology has created serious obstacles for multimedia content integrity, necessitating the urgent necessity for reliable detection methods. This study examines how well-advanced techniques like watermark embedding, metadata verification, and hashing work when combined with a Deep Fake Lab to improve the detection of manipulated media. The solution under consideration makes use of cryptographic hashing techniques to generate unique IDs for authentic multimedia content. Advantages of referring to hashes; The algorithm may also detect hashes indicating possible deepfake manipulations by computing hashes of questionable media and comparing them to reference hashes of high-quality sources. Moreover, metadata is studied to detect abnormalities in the metadata of media files that are inherent to the image such as timestamps, camera settings, and timestamps which are comparable to deepfake manipulation. Moreover, during the development process, the various ways you can integrate watermarks (which are small, indistinguishable electronic footprints) are also explored. Despite its potential for distortion of its image, these watermarks work as reliable determinants of accuracy and can be helpful for establishing whether any changes or manipulations have been made illegally. The above techniques provide a broad range of measures for estimating and eliminating the spread of deepfakes on different media platforms through the use of a centralized Deep Fake Lab. The effectiveness of the proposed framework with regard to identifying various forms of altered or harmful media is then examined in light of detection accuracy, resistance to adversarial attacks, and computational complexity through extensive testing utilizing diverse datasets of legitimate or altered media. Some of the key findings of the research provide an opportunity for greater support in ensuring that audiovisual information remains intact in a world that is increasingly in a digital age as well as giving rise to improved technologies for the development of deepfake detection.

Table of Contents

List of Figures

Chapter 1

Introduction

With time deepfake technology will become advanced and one will not be able to differentiate between real media and the alternative one manipulated. This project is able to solve the said problem by introducing frame by frame hashing technique, triple check of image using metadata along with image embedding using watermark detection technique that could work on all the three mediums i.e in picture form, video form and audio form. The core of the project is an algorithm for the intelligent hashing of visuals, which is specifically designed for popular picture, video, and audio formats. This program is very detailed on how it goes through audio or video clip in comparison to the source for some minor anomaly on its side. With this metric the system of possible deepfakes can be detected. Metadata verification checks the information contained in the media files such as its alteration history, the time and date shot a picture or video was taken and the type of camera used. The authenticity of media is measured in part through the deviations detected through the metadata inspection. Furthermore, it is possible to add watermarking for the video in order to achieve greater confidence. Digital watermarks can be defined as a label that is inserted into a material during the manufacturing process in a way that is not visible to human sight. These water marks are very useful for authenticity as it highlights areas where changes/alterations can easily be detected if they are made illegally.

1.1 Overview

This project aims to add a new dimension to the existing protection solutions for images, videos, and audio through combining some commonly-used anti-deepfake algorithms, such as hashing, metadata verification, and watermarking algorithms, and proposing a new solution that can be applied to data in different formats through a unified algorithm design and flow. The frame-by-frame hashing method is one of the most laborious and exact ways of comparing every frame of media with the original in search of differences that may point toward the existence of deepfakes. The use of watermarking has the added aspect of authentication and verification may be performed to evaluate the consistency of the information that has been embedded. When they are combined, these methods offer extensive detection efficiency, real-time decision-making, and straightforward integration into the current production chain systems to ensure that the falsified information that is effectively contained in deepfakes does not spread to various social media platforms or media formats.

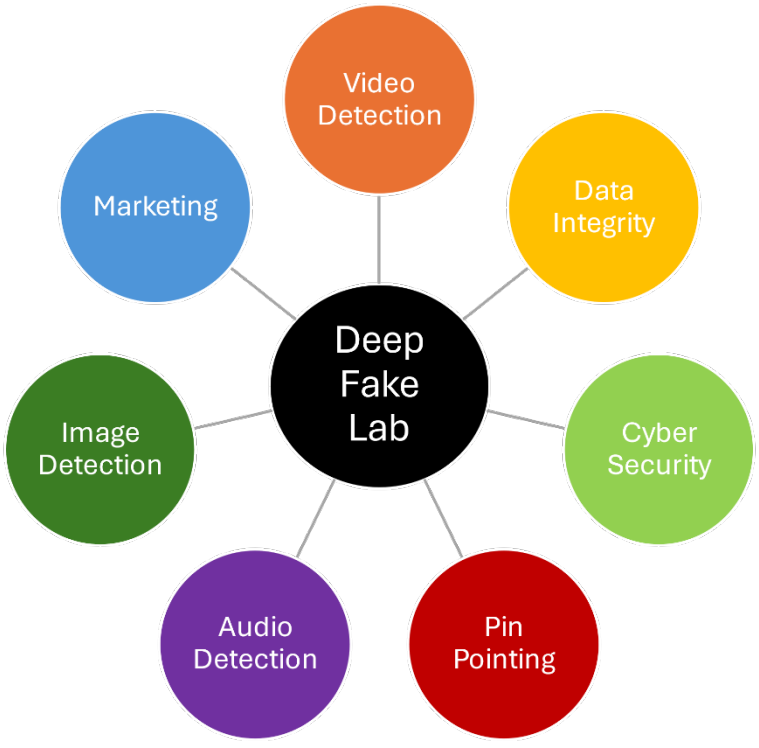


Figure 1: Overview of Deep Fake Lab

1.2 Problem Statement

Deepfakes have become one of the biggest challenges in our modern society with the advanced deepfake engineering making it possible to create the materials that can spread the misinformation, as well as cause a great amount of harm to the society, to the individuals, as well as the organization around the world. The risk of deceptive content has no been higher before because with Deepfakes, it is possible to create realistic content using these advancements in artificial intelligence and machine learning algorithms. The problem is that the use of deepfakes may mislead viewers or other consumers of the image and the information presented and influence their point of view as well as spread fake news and tarnish the reputation of another person by changing the timbre of his voice or simply through the use of a face on fake content. Moreover, it is also because such content can easily spread to a wider range in a very short time through social media and other media platforms. Preventive measures are thus urgently required to detect and mitigate deepfake images and ensure that people do not fall prey to the manipulation of information on digital media. This phenomenon is particularly difficult to combat and it is our project's goal to develop software for deepfake detection based on targeted approaches, like watermark embedding and metadata verification, to ensure that progress is made in a way that can effectively stop the spread of deepfakes across different media formats.

1.3 Proposed Solution

The Deep Fake Lab application has been designed to ensure that algorithms and technologies used to work with hashing, integrity verification, and watermarking can be used to identify or prevent deepfakes. I have outlined an idea that is based on an overall solution that must provide software to combat deepfake content across many forms of media. Some of the core components of this framework are to hash the media content using a cryptographic digest to identify unique identifiers and for comparison and detection of changes that may indicate tampering. A metadata verification layer can look for inconsistencies in metadata such as a block of edits or timestamp dates. Moreover, the

application works with metadata, enabling its consumers to use digital signatures to insert watermarks into media assets during their creation and, thus, guarantee the authenticity of the information. Real-time analysis is possible in this case since deepfake tracking modalities can be used to identify potential violations and address them as soon as they occur. A clear functional design allows for easy uploading and validation of media content and analyze them for deepfakes; ensuring and ability to integrate other services and flexibility for the developer to grow and shift focus if necessary. It works to improve the chances of recognizing deepfake manipulation to stop allegations of altered content being distributed across the net and protecting the integrity of digital media.

1.4 Working Principle

The Deep Fake Lab application employs hashing algorithms, metadata verification, and watermarking to avoid content containing deepfakes. The hashing algorithms generate signatures for media files that are then used to ensure that when there is any change in the file there is disruption in the signature. The metadata verification component works with the examination of editing histories and time stamps of the metadata embedded in the images in order to detect inconsistencies that would point to possible manipulation. Further, metadata is employed to generate watermarks, during the creation process of a media asset the application stores digital signatures in them. This approach reduces the effort needed in ensuring authentic media quality and also sustaining the quality of digital media.The list of modules is as under:

- Hashing Module
- Frame-by-Frame Hashing and Verification Module
- Metadata Verification Module
- Watermark Embedding and Detection Module
- Metadata Extraction
- Audio Hashing and Verification Module

- User Interface Module
- PDF Generation Module

1.4.1 Hashing Module:

- Produces the digits of fingerprints for multimedia content by applying cryptographic hash functions such as SHA-256. These are hashes or unique digital signifiers that correspond to content details and data for multimedia files.
- Does hashing for each segment of multimedia files including image frames. But the longitudinal diction is segmentation provides for identification of alterations and tampering at the finest level.
- Facilitates quick determination of whether files that are carried out are suitable by comparing the computed hashes with the reference hashes of original sources that are used for the production of original media files. Differences in the computed and reference hash values can be an indication of changes or an attempt at modifications to the media.
- Supports an effective and consistent proof-of-origin capability through the provision of identifiers for the recorded multimedia. The computed hashes are used to act as digital signatures to enable a user to authenticate the veracity of an original content and identify any alterations. This verification protocol adds to the increase in the level of trust, and assurance in relation to the authenticity of multimedia content in the area of countering deep fakes.

1.4.2 Frame-by-Frame Hashing and Verification Module:

- It computes and saves SHA-256 sums of the frames in a video. The hashes are an attempt at a cryptographic signature for the video frames that are contained in a separate text file related to the video file. This ensures that each frame is tamper proof and can be traced and self checked.
- This function takes the already stored hash values and compares it with the hash values obtained from the frames of the video and gives result in case of alteration or tampering. If any inconsistency reared its head then the system identifies the time of the changes based on the frame rate information and can thus identify the precise changed frames of video.
- This function checks if the stored hashes derived from the frames match the hashes of the frames from the given video or not to find any changes in the frame or not. The problem helps detect the exact time of post-processing by recording the frame rate, thus identify frames of manipulation.
- This function hashes the one-time stored hash values and compares the newly computed hash values with the video frames to ensure any alteration or tampering with the video file. If any error is detected it identifies the exact instant of the frame-rate based on the frame-rate information, which helps to find out the frames that is/are tampered.
- It defines two additional functions which compare the hashes stored in a file with the newly calculated hashes for the video frames in order to check for any alterations or manipulation. It provides accurate identification as it is possible to locate the tampered frame exactly if a difference is present and it can pinpoint the exact time of the change based on frame rate-related information.

1.4.3 Metadata Verification Module:

- This module analyses systemically the hidden tags that are contained in multimedia files. Metadata also contains supremely important information such as time stamps camera models and editing histories which are often necessarily always associated with the authenticity of content. To determine whether there has been any manipulation of such information, the module scans the metadata of the

information and scans for any conspicuous modifications that might indicate manipulation.

- On close inspection one can deduce that the module is driven by algorithms designed to identify gaps or anomalies in the metadata. An example of such anomalies may be discrepancies between date and time stamps, non-homogeneity of the camera or non-standard edit histories. Any shifts from the expected behaviors also indicate such variations in the multimedia contents that they are applied to.
- That is why the metadata that is used to authenticate the multimedia turns out to be an integral part of this module. The measures also serve the purpose of falsification through validation by verifying that the metadata meets the delineations of fake content from real content. It raises concerns and encourages the usage of the content integrity inquiry towards differences and exceptions.
- Through detailed analysis of embedded metadata, the module is involved in the enhancement of reliability of processes for proving the product origin of multi-media content. The module ensures the overall integrity of the system by providing a process to authenticate the metadata, i.e, ensuring that only those sources of information which are considered to be authentic can be deemed as trustworthy.

1.4.4 Watermark Embedding and Detection Module:

- This module adds the methods for watermarked embedding in multimedia data. It uses hashed metadata for creating each watermark, so that it correlates with the content. In addition to these watermarks, unique identifiers or authentication codes are occasionally stamped on the content in order to validate its origins. The module installs and implements appropriate software to crest watermarks directly into the multimedia files for it to carry markers of its authenticity.
- One of such algorithms is used to put watermarks in the multimedia content and the other one will be used to reveal the hidden markers that were embedded in the content. These detection algorithms scan the content to determine whether there are markers and if what is being embedded is legit. If the content is tampered

with, it will leave a pattern or signature that is unique to the particular watermark and the module will be able to detect it.

- This module enables the detection and embedding of images for the authentication of multimedia content at a higher level. It adds invisible, but strong indicators of content authenticity to determine whether it has been changed by third parties when used by users. This would be in the form of the incorporation of these digital watermarks which would be used to ensure that the users do not get manipulated multimedia contents by other users and this would supplement the whole process.
- Watermarks as an additional feature allow identifying and securing multi-media information from unauthorized alteration. Watermarks serve as unique identifying labels that remain unaltered by efforts to manipulate the content through distortion. Hence, the module helps in curbing deceptive manipulation since multimedia content may have been manipulated as deepfake.

1.4.5 Metadata Extraction:

- This module is chiefly involved in obtaining more details about multimedia metadata from videos and other associated multimodal files like images and audios. The extracted metadata contains fields like codec information, resolution, frame rate, bit rate, sample rate and creation date of the media which adequately depicts the content of the media files.
- The module guarantees the proper and accurate retrieval of the metadata using the tools and libraries like ExifTool, FFmpeg, and for photos and audio, PIL for Python libraries and pydub. This avoids one from having to undertake any intensive processing operation to capture the pertinent metadata.
- Hash Generator module is activated following the extracted metadata. It also also used by the Watermark Embedding module to generate watermarks on the base of the hashed metadata thus working within the framework of Deep Fake Lab.
- The Metadata Extractor lacks just basic attributes; it also work to retrieve some other more complex attributes such as history of editing and the programs used to create the image. This analysis gives a more thorough examination of the the

media file and also provides more information about the file as it relates to authenti

1.4.6 Audio Hashing and Verification Module:

- This module is tasked with creating a hash value using alpha numeric characters from SHA-256 and inserting it into the metadata of an audio file. To accomplish this, hash is generated from the metadata of the audio file making each file to have a unique digital versions. The module employs the Mutagen library to extract metadata tags from the audio file and to create the hash value and save it as a custom ID3 tag to allow for change of the audio file by the user without the hash changing.
- The module located the metadata in both the original and the hashed audio file, created a SHA-256 hash from the metadata of the original file, and checked this hash against the hash that was provided for the original file. But with the help of Mutagen library, the module can perform deep analysis of the audio file metadata like mime type, sample rate, bitrate, channels, duration and much more.

1.4.7 User Interface Module:

- This module has been built with very friendly and interactive interface that makes it easy to use. Guarantees that it is simple but at the same time it does not have complicated technical instructions for the ones who have much knowledge to the ones who have on information at all.
- Input interface which is used for uploading the material for analysis and supports several formats from images to video or audio files. The system can be accessed by the customer through their devices and can upload information.
- It has controls to help it initiate hashing and integrity operations.The user can choose the desired features to run particular operations – for example, generating hashes or checking the integrity – and then launch the process with a single click. The status and results of these operations are printed in a section by section output for the user's reference.
- The interface offers an in-depth view of the analysis in a relatively defined manner. It is also possible to see a summary of the results in the reports that

include information about metadata extraction, hash comparisons, and results showing whether watermarks were detected or not.

1.4.8 PDF Generation Module:

- This module provides intelligent statistical reports of the results derived from analysis of multimedia content. It can take results produced by various modules, such as metadata checking or watermarks presence, and generate a report for evaluation and reference.
- The generated reports contain comprehensive information on structural and behavioral authentication of multimedia content. These include information on metadata abnormalities, discovered watermarks and other relevant information in order to ease referring and analysis of the information by the users.
- The module saves a lot of time that is usually used to create and save the PDF report. Once the analysis is done to the various media sources the users are asked to create the report. After that, the data is compiled for the module and the PDF created in the designated directory. Users are notified when the report has been generated and then given the path to where it can be found.
- The module provides an option for generating PDF document files for distribution and sharing of reports. The ability to share reports with coworkers, clients, or third-party agencies for review, both for its performance, as well as archiving makes communication and decision-making regarding multimedia content analysis much easier.

1.5 Objectives

1.5.1 General Objectives:

To create an application that will eliminate and/or reduce any potential deepfake movies populating the web and other platforms. Another purpose is to supply a credible and useful deepfake-exploitation system that aims to prevent false information from being disseminated, especially in critical areas such as politics, health, and national security. Our project aims to reduce the generation and use of deepfakes and to contribute to a culture where online video content is used responsibly.

1.5.2 Academic Objectives:

- To help promote the field of computer science and artificial intelligence by devising a new algorithm that will be employed to deepfake content on various websites by the site's administrators / moderators.
- In order to understand and to try and assess the efficacy of frame-by-frame hashing as a primary means of detecting deepfakes as well as how it compares to other existing methods of counter-deepfake technologies.
- To present and share the findings and results of the study in a form of a research paper or academic journal that may be helpful for other researchers or like-minded academics who are studying the issue of deepfakes.
- The project is aimed at producing hashes based on metadata and media files to create 'fingerprints' of the files that can help identify whether changes are made to them. hashing algorithm is the most effective way of maintaining the integrity of an exchanged information in a confidential and efficient manner. Using these hashes and comparing them while embedding them in deepfake artefacts automatically shows differences that pinpoint manipulation in the images. This objective will improve the security and integrity of media content thereby making it much harder for people to manipulate content seamlessly.
- Employ strong testing and validation processes to determine the software solution's proficiency. Confirming the accuracy of what the system has picked up

and comparing it to the actual annotations for a system's reliability and trustworthiness.

- As we are already gaining expertise in the using of machine learning to solve real-world problems they will provide practicing for students or researchers to demonstrate their competence in using this tool and put them in a position where they can apply such knowledge in their future careers or academic work.

1.6 Scope

One of the central projects includes the development programming that would allow for the accurate detection of deepfakes and accurate specification of manipulated movies on various Internet resources. This involves the use of advanced techniques such as metadata examination, and also the embedding of watermarks and generating of hashes. The software will therefore have to function on many operating systems and web platforms to enhance the focusing on the analysis of the various multimedia content that is in form of pictures, videos and even audio files. The proposed method will focus on achieving high detection accurateness, resistance to intentional attacks, and real-time execution. Although potential ethical concerns related to the use of the technology will be addressed at various stages of the project up through the actual implementation and adoption of the technology, this process will be facilitated further by ensuring that any documentation provided with the initiative and a strong user support system.

1.7 Deliverables

1.7.1 Software Requirement Specification:

This article aims to present a general and detailed description of deep fakes. It will outline the system's features and functionality, its interfaces, what it will do, how it will do it, what has to be done to meet the requirements, and how the system will react to external stimuli. This material is intended for both official users and regular users.

1.7.2 Software Architecture Document:

The general architecture of the system is covered in this article, along with the introduction of different parts and subsystems. It is primarily supported by a system architecture diagram, which summarizes the high-level software components that perform the essential functions required to keep the system operating and gives an insider's view of the system.

1.7.3 Software Design Document:

The design document summarizes 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 these needs into practice.

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 Evaluation Reports:

Comprehensive reports that provide a detailed summary of the functionality, precision, and efficiency of the software solution after it has undergone rigorous testing and assessment. These papers contain in-depth analyses of the system's capabilities, including evaluations of its advantages, disadvantages, and possible improvement areas.

1.7.6 Software Testing Document:

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

1.7.7 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 is committed to reducing violence and achieving peace and justice through strong institutions. This is a particularly pertinent project under SDG 16 as it addresses the growing threat of deepfake videos that could be used to create conflict within society and cause political instability. The invention of a piece of software that is meant to be used to authenticating and to provide a barrier against deepfakes in a way endorses the main goal of strengthening of institutions and access to justice for all.

Deepfake videos are a cause for concern because they can be used to mislead an individual, organizations, and whole communities about the news and trust in the process. This project which seeks to develop an algorithmic software that will help in the identification of deepfakes does have a role in promoting and upholding peace and justice by reducing the posting of harmful contents and ensuring that information is not compromised.

This project also actively contributes to SDG 16 by ensuring that institutions are accountable and that information disseminated over the Internet is accurate and inclusive. This project contributes significantly to the creation of a trusted environment for online discussions since it introduces mechanisms to authenticate the videos uploaded to the platform. In general, this project’s pledge to stop such deepfake manipulation demonstrates its contribution towards SDG 16 that seeks to promote peace, justice, and strong institutions.

1.9 Structure of Thesis

Chapter 2 includes the literature review and background and analysis on which this thesis is based on.

Chapter 3 contains the design and development of the project.

Chapter 4 provides in-depth evaluation and analysis of the code.

Chapter 5 has conclusion of the project.

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

Chapter 2

Literature Review

A new product is introduced by changing and enhancing the attributes of similar earlier released products. A literature review is an important step to convert an idea into a new product. Similarly, an extensive examination of all related projects is necessary for the development of a product and its replacement in the logistics system. We divided our research into the following themes:

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

2.1 Industrial background

The digitization of the content creation process and the easy access to advanced AI software have changed the media industry. This section gives a brief introduction to the industrial background of the deepfakes technology and its impact on several sectors.

Deep fake detection using hashing meta data authentication and watermarking has been highlighted as one of the key areas of research and development in recent years as the technology to generate realistic fake videos has become widely available and usable. Deep fake technology poses a serious threat in numerous ways ranging from sharing false information to influencing public opinion or even engaging in fraudulent activities. The use of hashing algorithms for deep fake detection is ideal because it enables one to compare large quantities of data and even identify minor changes in pictures and videos easily and quickly. This can be used to recognize and stop deep fake content.

Industry's deep fake detection is one of the key areas that many companies are paying attention to at the moment due to the vast application of the technology in the media and entertainment industry or when it deals with security and surveillance. For example, social media platforms like Facebook and Twitter are making huge investments in the rise of deep fake detection to eliminate fake and insidious information that misleads their users. Likewise, DFD is being employed by security agencies and law enforcement bodies to detect suspicious activity and used against fraudsters.

Deepfake technology is an advanced technology that also leaves a lot of room for creating truly awesome advertising campaigns and personalized communications as it allows to make almost completely realistic photos and videos to promote the brand and communicate directly with the client. Marketers are seeing the opportunities that deepfakes present for creating product advertisements with the imitation of the exact people that consumers prefer in real life, this way leveraging the trust consumers place in such personalities for their brand authenticity and engagement. However, issues concerning ethics have arisen – specifically, issues relating to the disclosure and information authenticity of advertisements, so that contesters have seen the necessity of setting rules for the use of deep fakes in advertising.

Although the technology has various positive impacts, the issue of deepfake technology is a threat to journalism and news media because it eliminates objectivity and truth and allows information or images to be altered to deceive people. This means that journalists and the media organizations are now caught with a challenge when it comes to ethical standards, fact check mechanisms and the integrity of information they pass on to the public. There are plans to establish ways and means through which news journalists and other media personnel in news channels could effectively identify and demystify news deepfakes in news channels in order to protect the soul of news channels and media houses.

Cybersecurity and digital forensics issues that exist because of deepfake technology. These attacks can be used by criminals to identity theft or fraud and of surveillance or other process related to espionage and data security. Security specialists and the forensics community are already seeking ways to authenticate and verify content in ways that would best counter deepfake threats, including the involvement of government bodies and the need for action from all members of the industry.

2.2 Current approaches and their limitations.

Some of the existing deep fake detection methods can be categorized as image processing methods, machine learning methods and blockchain methods. Nevertheless, each of these perspectives has its own shortcomings.

2.2.1 Deepware Scan:

Deepware Scan is a deepfake detection tool that uses machine learning algorithms to identify features that are common in deepfake content. It aims to automatically detect inconsistencies in the content, e. g. , fake facial expressions or light/shading effects, to identify the potential deepfakes. But its efficiency is limited only to simple manipulations and relatively primitive detection methods.

2.2.2 Microsoft Video Authenticator:

Microsoft's Video Authenticator is a tool that works by detecting deepfakes based on different indicators and signals in videos. It uses sophisticated machine-learning techniques to analyze the video and determine whether content is genuine on the basis of elements like facial movements, voice tone, and the presence of image glitches. It provides automatic detection features but has been questioned about its effectiveness and privacy, even involving the use of facial recognition.

2.2.3 Deepfake Detection Challenge:

The DFDC-community is a collaborative effort which aims to encourage advances in the area of deepfake detection. It is a challenge run by some of the industry giants, and academic institutions that gives a dataset that comprises of authentic as well as manipulated content to the researchers and developers for them to develop the detection algorithms and then evaluate them. Each participant develops an A. I. in order to produce a solution that would be able to precisely identify deepfakes and to help innovate the process of recognizing deepfake technology.

2.2.4 Third-party Verification Services:

Deepfake detection is among the many services provided by third-party verification services for media authentication and integrity verification. These services use intelligent algorithms and human professionals to scan images and videos to detect any sign of deepfake creation. They may also serve as an extra piece of evidence when checking authenticity of media content, but their efficiency may depend on the quality of the algorithms used as well as the amount of human control involved.

2.2.5 Open-source Frameworks:

DeepFaceLab and Faceswap are two examples of open-source frameworks that allow for the creation and detection of deepfake content. These frameworks feature various capabilities such as facial landmark detection, image manipulation, and deepfake detection. While they are effective in allowing users to learn how to create deepfakes, their efficiency as effective countermeasures against deepfakes may also depend on factors such as user expertise and the sophistication of deepfake creation techniques. Image watermarking and digital signature verification also are ineffective due to the availability of advanced tools for creation of deep fake videos. Convolutional neural networks and generative adversarial networks have already been used in deep fake detection. But these methods need the training of large datasets of both real and fake images, which may be costly and time-consuming.

2.3 Research Papers

2.3.1 Fake-image detection with Robust Hashing:

A good analysis on the identification of manipulated photographs with robust hashing algorithm can be obtained from the paper entitled ‘Fake-image detection with Robust Hashing’ by Miki Tanaka and Hitoshi Kiya[1]. This research examines the issues occurring due to the extensive use of fake images, specially

those produced by deepfake technologies, and proposes the hashing method as a possible solution for amendments and the assurance of picture authenticity.

Because different algorithms have different abilities to locate bogus or manipulated pictures to back up your project's objectives, the writers provide comprehensive analysis of several strong hashing algorithms. The articles in the study focus on the importance of reliable detection methods that promote a l stop the spread of misinformation or fraudulent messages, which is a key topic for you.

The authors present the results of both experiments and tests which demonstrate that different types of resilient hashing algorithms are able to establish fraud in photos submitted for registration under various manipulations and transformation, from pixel scanning to compression artifacts. Presenting robust hashing as an empirical proof that it is indeed adequate and efficient to serve deepfake detection opens the possibility to enhance the reliability and accuracy of your specific detection methods.

It has also covered the limitations as well as the future scopes of the techniques that can be followed for fake image detection and then using the same to help you with the procedure of your project and in turn provide a vision of the new directions you can pursue for the culturing of the existing techniques. Overall, the data presented in this study can be considered useful in contributions to the field of deepfake detection and how to continue developing your research in this area.

2.3.2 Proactive Deepfake Defence via Identity Watermarking

The authors of the presented study recommend to resort to identity watermarking as a way to prevent deepfake technology from misrepresentation. This method is to embed digital signatures into multimedia media such that they function as labels for the origin of the media and make it easy to trace back to original and trusted sources [2]. The idea of the authors' work of inserting identity watermark

into multimedia files is to serve as a preemptive action against the dissemination of deepfakes.

This essay critically focuses on investigating the efficacy and benefits of identifying the potential value of the identity watermarking approach where if implemented, it will help reduce the risks of the deepfake manipulation, particularly its dissemination as lies and threats to one's reputation. According, the authors show in the empirical review and experimentation section their method in identifying and detecting deepfake material in various media types including photos and videos.

The conclusions made in the paper are directly in line with the vision of the project results, especially in the domain of detecting deep fakes. Instead you can help your project identify manipulated multimedia information more effectively and legitimately by proactively utilizing identity watermarking as a defense mechanism. On the other hand, students who take part in this study could contribute to developing new ways and methods to put a stop to deepfakes' dissemination through mass media.

2.3.3 Video Tampering Detection Using Difference-Hashing Algorithm

The authors of the presented study recommend to resort to identity watermarking as a way to prevent deepfake technology from misrepresentation. This method is to embed digital signatures into multimedia media such that they function as labels for the origin of the media and make it easy to trace back to original and trusted sources [3]. The idea of the authors' work of inserting identity watermark into multimedia files is to serve as a preemptive action against the dissemination of deepfakes.

This essay critically focuses on investigating the efficacy and benefits of identifying the potential value of the identity watermarking approach where if implemented, it will help reduce the risks of the deepfake manipulation,

particularly its dissemination as lies and threats to one's reputation. According, the authors show in the empirical review and experimentation section their method in identifying and detecting deepfake material in various media types including photos and videos.

The conclusions made in the paper are directly in line with the vision of the project results, especially in the domain of detecting deep fakes. Instead you can help your project identify manipulated multimedia information more effectively and legitimately by proactively utilizing identity watermarking as a defense mechanism. On the other hand, students who take part in this study could contribute to developing new ways and methods to put a stop to deepfakes' dissemination through mass media.

2.3.4 Detecting GAN-Generated Fake Images via Metadata Analysis

This paper presents a new method to detect counter-GANs photography that is a growing problem in counterfeit works using Generative Adversarial Networks[4].

The recommended action is to take the photographs and deconstruct their metadata fields like the camera model, creation date, GPS coordinates, etc. The method is aimed at searching for signs of fraud such as anomalies or inconsistencies that are characteristic for GAN generated images in metadata features that are considered a reliable means of verification of authenticity.

The authors demonstrate the effectiveness of their proposed approach for detecting and filtering GAN-originated fake images through a comprehensive experimental evaluation of the proposed algorithm performance. The results demonstrate that metadata analysis can be a strategic and timely addition to the toolkit for identifying deepfakes in order to proactively stop the distribution of manipulated media.

The information obtained from this research paper is very relevant to your deepfake detection project and might offer a better insight in which direction to incorporate metadata verification techniques into your detection pipeline. With the assistance of metadata analysis together with other possible methods of detection, your project will find a deeper degree of accuracy and reliability in identifying and eliminating problems corresponding to deepfakes produced by GANs..

2.3.5 A System for Mitigating the Problem of Deepfake News Videos Using Watermarking

These authors offer a process that prevents the propagation of fake news videos that have been edited using the deepfake technology with the help of watermarking methods. It overviews how watermarking is a process of embedding certain digital signatures or indicators that can be used to acknowledge whether media files are legitimate and consistent or not[5].

The proposed strategy aims at increasing the chance of tracking media content and its origin in an effort to reduce the rate of deepfake news videos. Content makers could generate a provable endorsement between the authenticity of a video and its source by embedding unique identifiers that are put into a video at the source.

The suppression of deepfakes envisaged by the authors' watermarking-based strategy does indeed work as the authors demonstrate through the empirical analysis and experimentation for deepfake news video.It clearly shows that watermarking is a viable tool as a potential safeguard against the propagation of misinformation.

The observations shed great insights in the form of recommendations for the your deepfake detection project – especially when it comes to using watermarking to mitigate negative effects of using manipulated media content. One of the ways that you may enhance the legitimacy and reliability of multimedia content is through the integration of watermarking techniques as part

of your detection systems. This will enable you to find a solution to the issues raised by such deepfake videos in the internet.

2.3.6 Learning to Detect Fake Face Images in the Wild

As the use of changed images gains popularity, especially deepfake images, it becomes necessary to find a solution to the problem and the research will try to solve the issue by providing the practical method of doing so using the machine learning approach[6].

The authors discuss in great depth the process of creating a machine learning model to differentiate these two types of photos based on their facial manipulation. Expectably, the suggested method demonstrates remarkable performance in the task of detecting fake face images for a wide range of scenarios and conditions, following extensive testing and evaluation.

Furthermore, the report helps in ascertaining the baseline of subtle details regarding malicious actor use of facial picture editing techniques. The study also explains the challenges that are associated with pinpointing these modifications while drawing the reader's attention to the fact that having strong detection mechanisms is crucial based on the details that have been noted and artifacts in the fake face images.

The conclusion drawn from this work shows that the research results are critical and useful for the science of deepfake detection and present one of the means for the future limitation of manipulated videos sharing. Scientists and professionals working on AI systems may strive to design better and more effective detection systems to prevent the harmful effects of deepfake pictures of people in cyberspace with the help of the concepts and methods used in this work.

2.3.7 Deepfake Video Detection using Image Processing and Hashing Tools

The current study aims to describe the use of the image processing and hashing techniques to evaluate the content of the image and the video for manipulation. The authors of the work talks about different algorithms and methodologies that can identify deepfake by using various sign of creation such as visual artifacts and inconsistencies synthesised during the process [7].

The authors of the article managed to develop an effective method to combat manipulated content by applying image processing and hashing algorithms. DeepFake detectors use visual forensic to identify the visual distortions caused by manipulations such as swaps in an image. Hashing tools create digital fingerprints for media files so that their integrity could easily be determined by comparing and matching their content's unique code. The inclusion of two detection methods helps improve the accuracy, offering an effective framework for the deepfake detection.

The study is extremely relevant as it focuses on the issue of misinformation through fake or manipulated digital content and aims to provide methods for reliable and accurate identification of the altered content that is published digitally. The outlined integrative model makes a wide-ranging contribution to the research on digital media integrity and provides ways how to address the problem of deepfake technology.

2.3.8 A Survey of Deepfake Detection Techniques: A Comprehensive Review

The authors of the current survey article provide a comprehensive overview of the recent research in the area of deepfake detection addressing all the possible techniques and approaches to the issue [8]. They organize and discuss the

various methods in a systematic way which includes the image method, video method, audio method, and various detection technologies as well.

The aim of this comprehensive piece of work is to elucidate the benefits and the disadvantages of each detection approach as well as the current status of the DeepFake technology detection research. I must add that the writers have defined keen perspectives on their expected future developments in the discipline as they discuss facts and possible future courses.

The survey paper is relevant to the present study according to the given requirements since its conclusions offer a general picture of the types of detection techniques and their habitat while focusing on deepfake detection in particular. With the aid of the information and skills described in this paper, you will be able to develop strong detection mechanisms that will be helpful in halting the spread of deep fakes to different platforms for dissemination.

2.3.9 Detection of Deepfake Video using Hybrid Cryptographic Techniques

This project provides a research into the use of hybrid cryptography techniques for deep fake video detection. The authors' aim is to propose a more viable solution for the detection of maliciously-altered video content by utilizing various cryptography algorithms in building the detection system[10]. The investigation also discusses the constantly escalating impact of deepfakes and the significance of advanced tools for addressing this issue.

The approach used in the paper encompasses the use of cryptographic hash and encryption to enhance the fault detection systems across electronics. They make use of hash and then encrypt the resultant data through the cryptographic algorithms so as to ensure them the integrity of the data. It has many benefits as it makes any of change or modification made to the video content easily identified. The paper also includes some information about the

implementation details and time complexity of the proposed method, and provides evidence why it will be feasible for real-time applications.

Pandey, Gupta, and Jain did an experiment to examine the efficacy of the proposed hybrid cryptosystem with the help of datasets of videos with both the original and manipulated video clips. This implies that the early detection rate compared to the classical approaches was improved. Hybrid approach was able to distinguish deepfake videos also when the video is manipulated to the advanced level of modification. The authors conclude that their method presents a secure and effective method for countering deepfake and that by tackling this issue they are thus able to suggest a promising direction in the way towards the future of media tampering prevention. The current problem shows the need for more advanced cryptography to address the problem brought by deep fake technologies.

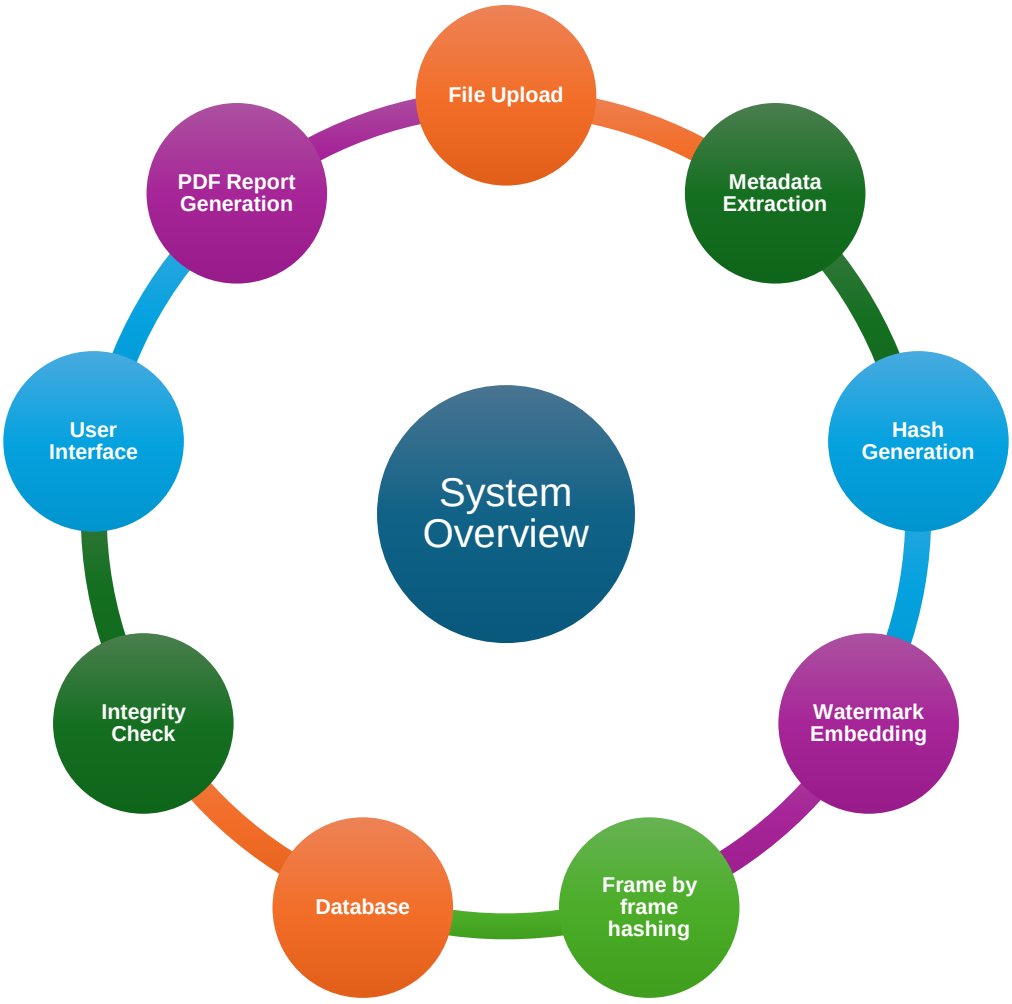
Chapter 3

Design and Development

3.1 System Overview

The suggested deepfake detection identification system is a global system which has ways to detect deepfake content for photos, videos, and sounds. It consists of several interconnected elements which provide the basic level of security and ensure the authenticity of digital content. It has a few modules intended to deal with various aspects regarding the detection and verification of deepfake media.

Figure 2: System Overview



3.2 Architecture Design

The systems architecture design for the Deep Fake Lab is focused on ensuring that reliable and effective solutions for the detection and mitigation of deepfake content in multimedia files should be possible. The design of this architecture has several modules that perform specific functions and contribute to efficient flow of data and the ability to perform such real-time processing and analysis tasks.

3.2.1 Components and Interactions

3.2.1.1 File Upload Module:

- **Function**: Allows upload of photos and video and audio recording for content analysis.
- **Implementation**: It has been developed using Tkinter for having the GUI interface.
- **Interactions**: Accepts data from the user and forwards it to the Metadata Extraction Module

3.2.1.2 Metadata Extraction Module:

- **Function**: Extracts detailed metadata from the uploaded multimedia files.
- **Implementation**: Utilizes libraries such as PIL for images, FFmpeg for videos, and pydub for audio.
- **Interactions**: Extracts metadata and sends it to the Hash Generation Module.

3.2.1.3 Hash Generation Module:

- **Function:** creates stand-alone crypto hashes (e. g. the same exhaustive set of hashes (e. g. using SHA-256), which are utilized as digital fingerprints for verification.
- **Implementation:** Uses hashlib for construct secure hashes**.**
- **Interactions:** Receives metadata from the Metadata Extraction Module and sends hashes to the Watermark Embedding Module.

3.2.1.4 Watermark Embedding Module:

- **Function**: Adds digital fingerprints to the media files as a part of the generated hashes to verify the content's integrity.
- **Implementation**: Provides features for encoding without altering the genuineness of media.

- **Interactions**: Shapes the unique hashes from the Hash Generation Module and stamps the media files and sends it to the Real-time Processing Module.

3.2.1.5 Real-time Processing Module:

- **Function**: It never stops scanning the media files for any evidence of tampering and automatically detecting deepfakes as soon as they appear.
- **Implementation**: Implements real-time processing algorithms to analyze media integrity.
- **Interactions**: Monitors watermarked files and sends data to the Integrity Check Module for verification.

3.2.1.6 Integrity Check Module:

- **Function:** Validates the integrity of the media files by comparing their hashes with the previous stored ones**.**
- **Implementation**: Uses hashing and comparison algorithms to detect discrepancies.
- **Interactions**: Receives data from the Real-time Processing Module and updates the User Interface Module with integrity check results.

3.2.1.7 User Interface Module:

- **Function**: Is an easy to use software for uploading, analyzing, and validating multimedia files with instant feedback and comprehensive analysis reports.
- **Implementation**: Developed using Tkinter for the graphical user interface.

- **Interactions**: Shows feedback and detailed analysis results to the user and interacts with other modules in real-time.

3.2.1.8 PDF Report Generation Module:

- **Function:** Produces the final PDF reports of the analysis with metadata, hash hashes, and integrity check report**.**
- **Implementation**: Uses FPDF or ReportLab for generating PDF reports.
- **Interactions**: Receives data from the User Interface Module and generates comprehensive reports for the user.

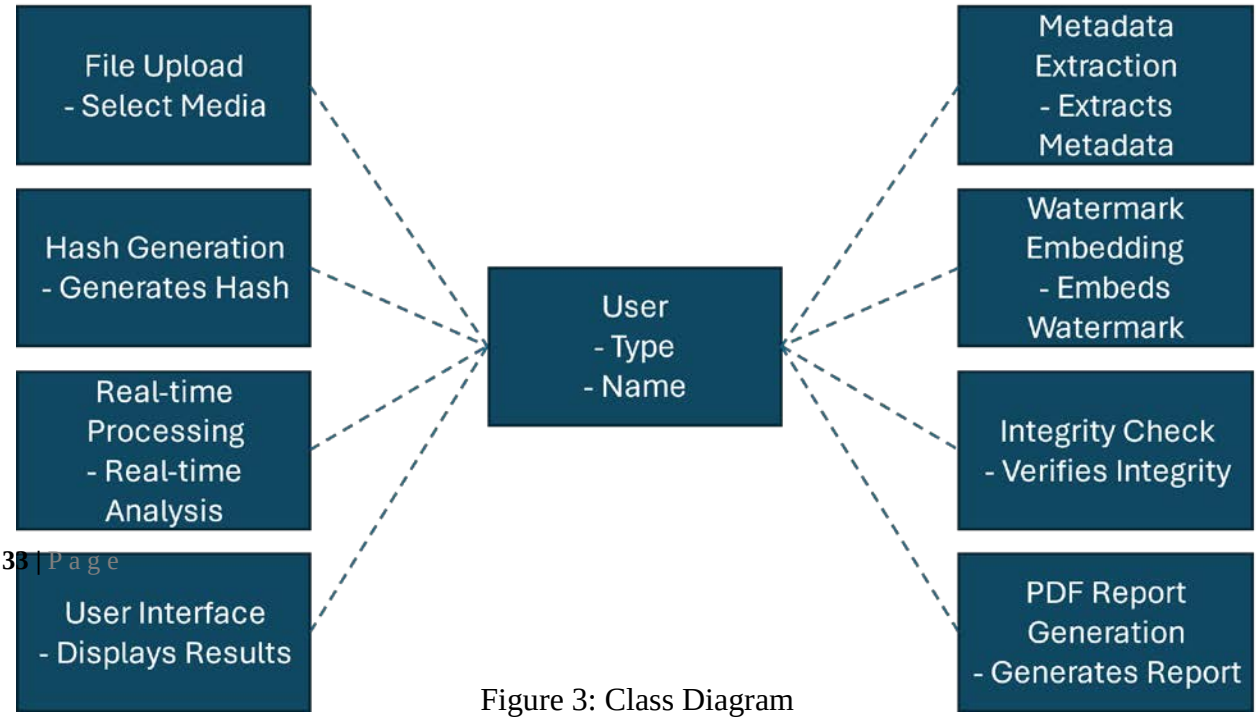


Figure 3: Class Diagram

3.3 Process Decomposition

Sequence as well as use case diagrams that break down the system into separate and united processes are used to explain the process decomposition. The use cases explain the series of activities a user performs in when working with Deep Fake Lab.

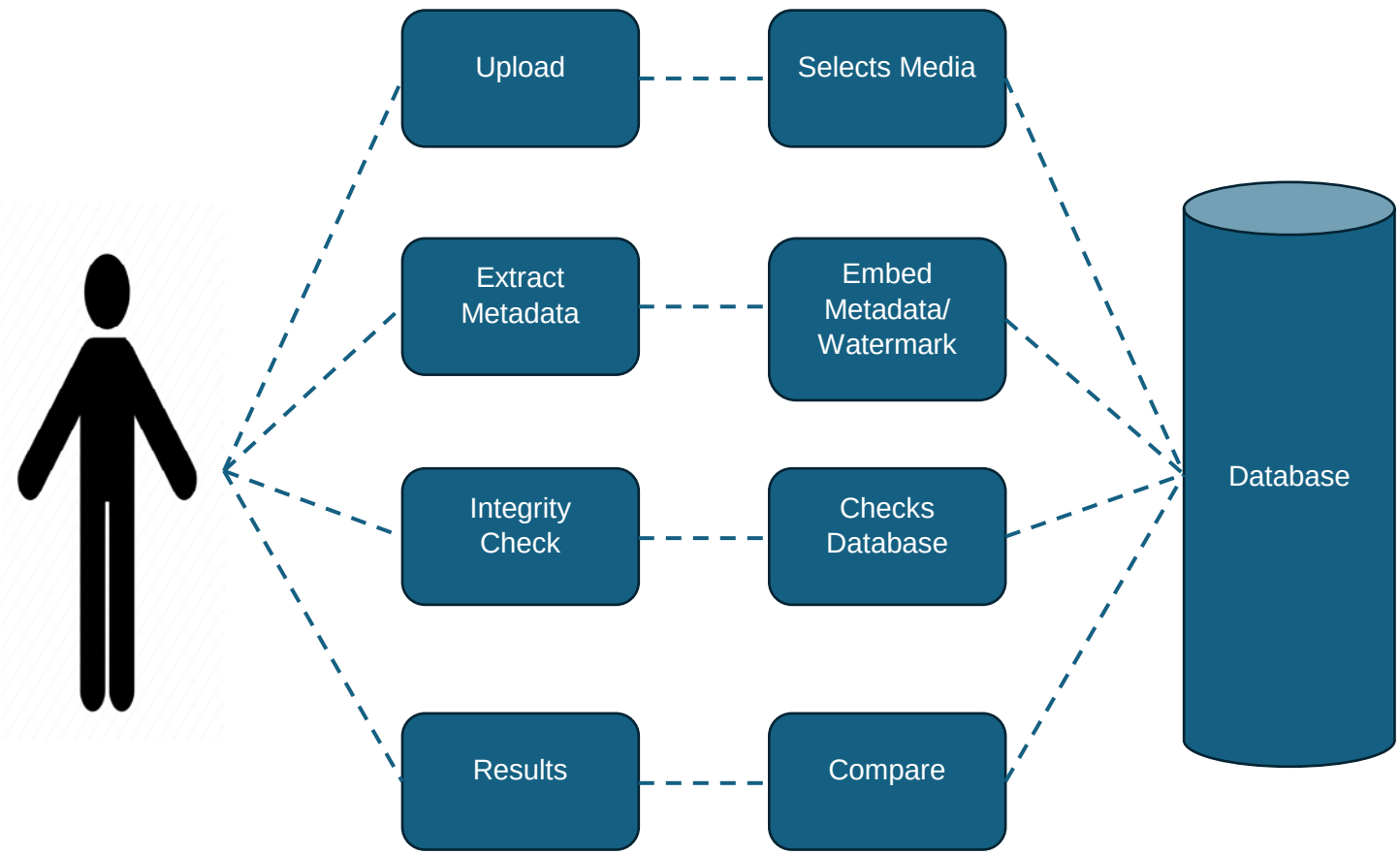


Figure 4: Use Case Diagram

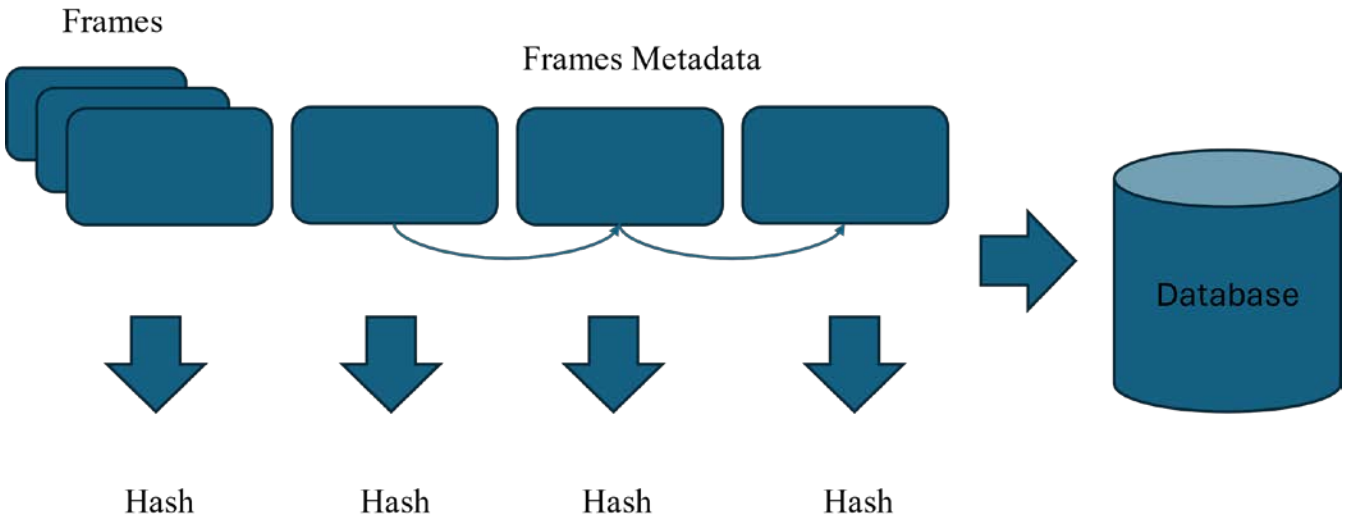


Figure 5: Prevention Flow Diagram for Video

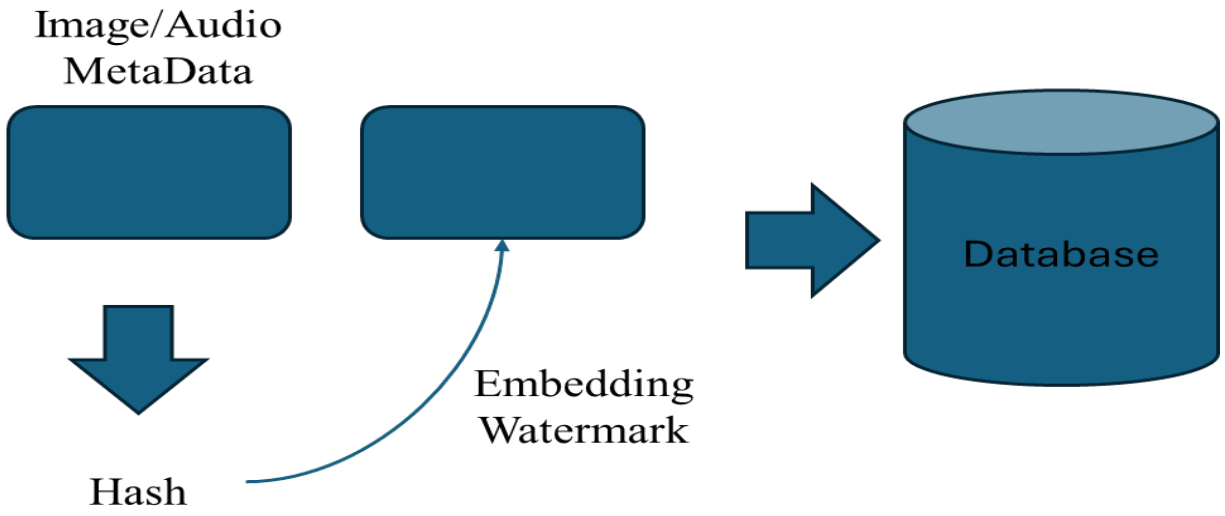


Figure 6: Prevention Flow Diagram For Image/Audio

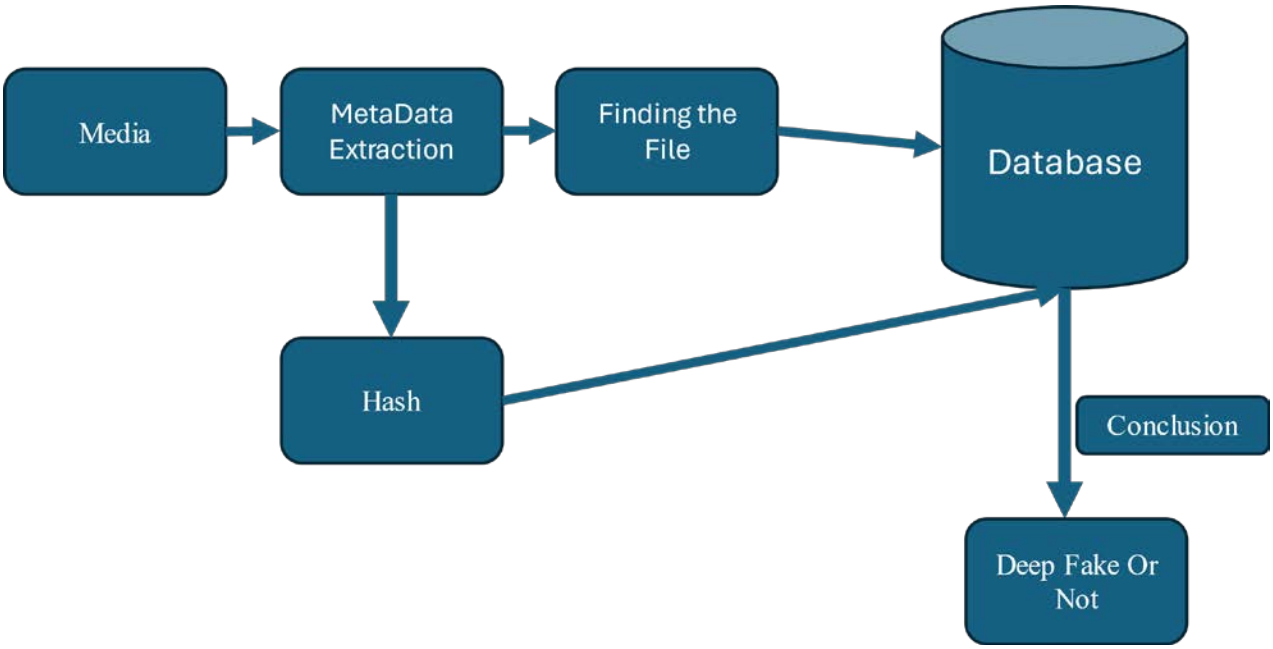


Figure 7: Detection Flow Diagram

3.4 Design Rationale

The focus when designing my project is to ensure that a hashing technique-based deep fake detection tool is built, with a separate originality check tool added as an option as well. The structure of the project is comprised of an application-oriented architecture with a front-end interface and several backend modules for deep fake verification and originality.

A decision of making use of application based architecture was made to make sure the users could have an easy access to the application on their local machines. This method ensures they can engage with the system without any internet or web servers. Using just an application that does not require any browser makes it easy to deliver the desired uniformity of experience no matter where the user is.

Python was chosen to be the primary programming language because of its powerful library support and its outlier user friendliness. Options like Tkinter for user interface, OpenCV for picture and video modeling, hashlib for developing hashes ensure that Python can transform to one of the very best languages for creating scalable and maintainable program. This helps one in the creation of a fast and effective, application that will also be user friendly.

Among the reasons why Python was the primary language used was its ability to use large library resources. The user interface library Tkinter, image/video processing library OpenCV, and hash-generation library hashlib make Python an excellent choice for constructing the application logics that are scalable and maintainable. It can be attained through use of a framework that supports rapid application delivery while adding value to the user experience.

It was a necessary addition to offer users an option to authenticate any picture or video as original. With the hash values we can quickly tell if any uploaded images or videos that an original can be compared to our database of known originals. This feature

goes a long way into boosting the security of analysis work because users know that they are dealing with authentic content.

The decisions on design were made with great emphasis on the scalability and extensibility of the solution. The above structure lends to the systems flexibility as it is expandable without affecting the existing operations of the system. The use of Python and hash algorithms guarantees that the system can work as more user base and data grows and also its reliability over time.

Based on the recommendations for design rationale for my project the design rationale for my project is to create a product which is easy to use and provide an alternative solution for deep fake detection as well as incorporation an originality check feature via the use of hashing techniques. The selection was based upon the simple and time effective application of chosen technologies or methodologies permitting to achieve well scalable and efficient solution.

Chapter 4

System Implementation

The system implementation stage outlines the development and deployment of Deep Fake Lab as an application. The following section provides the description of the development environment the technologies and libraries used implementation details of each module.

4.1 Development Environment

4.1.1 Hardware Requirements:

- **Development Machine:** Intel Core i7, 16GB RAM, 512GB SSD.
- **Minimum Requirements:** Dual-core processor, 4GB RAM, 100GB storage.
- **Recommended Requirements:** Quad-core processor, 8GB RAM, 256GB storage.

4.1.2 Software Requirements:

- **Operating System:** Windows 10, macOS, or Linux.
- **Development Tools:** Visual Studio Code, Python 3.x.
- **Version Control:** Git.

4.2 Libraries and Dependencies

4.2.1 Python Libraries:

- **Tkinter:** For the graphical user interface.
- **OpenCV:** For video and image processing.
- **hashlib:** For generating hashes.
- **Mutagen:** For audio file metadata extraction.

- **FPDF:** For generating PDF reports.

4.3 Integration and Testing

4.3.1 Integration:

- There was an attempt to make the different modules work together where data was exchanged between the modules. For example, the information retrieved in the Metadata Extraction Module is sent to the Hash Generation Module to calculate the hash based on the information.
- Challenges faced during integration included ensuring compatibility between different libraries and managing dependencies. These were resolved by thorough testing and version control.

4.3.2 Testing:

- Unit testing was conducted for each module to ensure individual functionalities worked as expected.
- Integration testing ensured that the modules worked together without issues.
- User acceptance testing involved real users testing the application to ensure it met their needs.
- Example test cases and results were conducted to track the testing process and identify any bugs.

4.4 Graphical User Interface (GUI)

It was at least an attempt to integrate the various modules and collaborate in transferring data between the modules. For instance, information obtained in the Metadata Extraction module is passed to the Hash Generation Module for the generation of hash based on this information.

4.4.1 Key Components of GUI:

4.4.1.1 Main Window:

- There is a Main Window in the Application with which users of the system can interact.
- Title: "File Uploader"

4.4.1.2 Background Video Playback:

- A video plays in the background to provide a dynamic and engaging user experience.
- The video playback is handled by the imageio library.

4.4.1.3 Heading Label:

- An image serves as the heading label, positioned at the top center of the window.
- The image is loaded using the PIL library.

4.4.1.4 Upload Button:

- A button with a play icon allows users to upload files.
- Clicking this button opens a file dialog for selecting the file to be analyzed.
- The button is styled with a background color and an image icon.

4.4.1.5 Output Label:

- Displays the path of the selected file.
- Positioned below the upload button.

4.4.1.6 Mini Terminal Box:

- A text widget serves as a mini terminal to display real-time feedback and status messages.
- Provides information about the ongoing processes and results.
- Positioned centrally with a label indicating "Output."

4.4.2 Functionality:

4.4.2.1 File Upload and Media Type Detection:

- When a user clicks the upload button, a file dialog opens to allow file selection.
- The selected file's path is displayed in the output label.
- The system determines the media type based on the file extension and displays it in the media type label.

4.4.2.2 Process Selection:

- Users can select either the integrity check or hashing function by checking the respective checkboxes.
- The selected process type is used to determine the appropriate processing steps.

4.4.2.3 Start Processing:

- When the start button is clicked, the application processes the uploaded file according to the selected function (hashing or integrity check).
- There is a mini box with a terminal where we can observe the data processing in real-time, namely hash calculation, watermark embedding, and integrity verification.

4.4.2.4 PDF Report Generation:

- Users after processing are asked to make a PDF summary.
- If the user chooses to generate the report, the system collates all the analysis and generates a professional and comprehensive report in the PDF format, which it then saves to a chosen location.

4.4.2.5 Real-time Feedback:

- The mini terminal box shows the progress stage of the contributions submission, processing, as well as the results of the contributions.
- The users should be informed about each step and any possible problem that may occur throughout the processing.

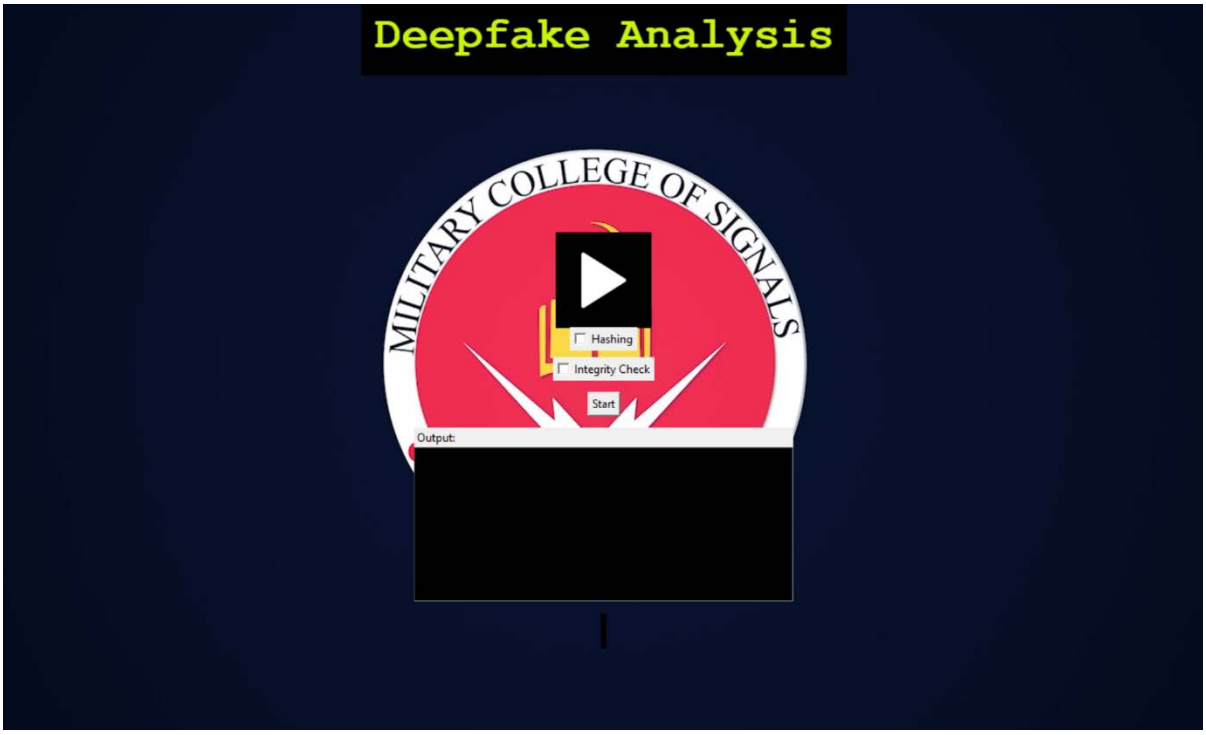


Figure 8: Main Window

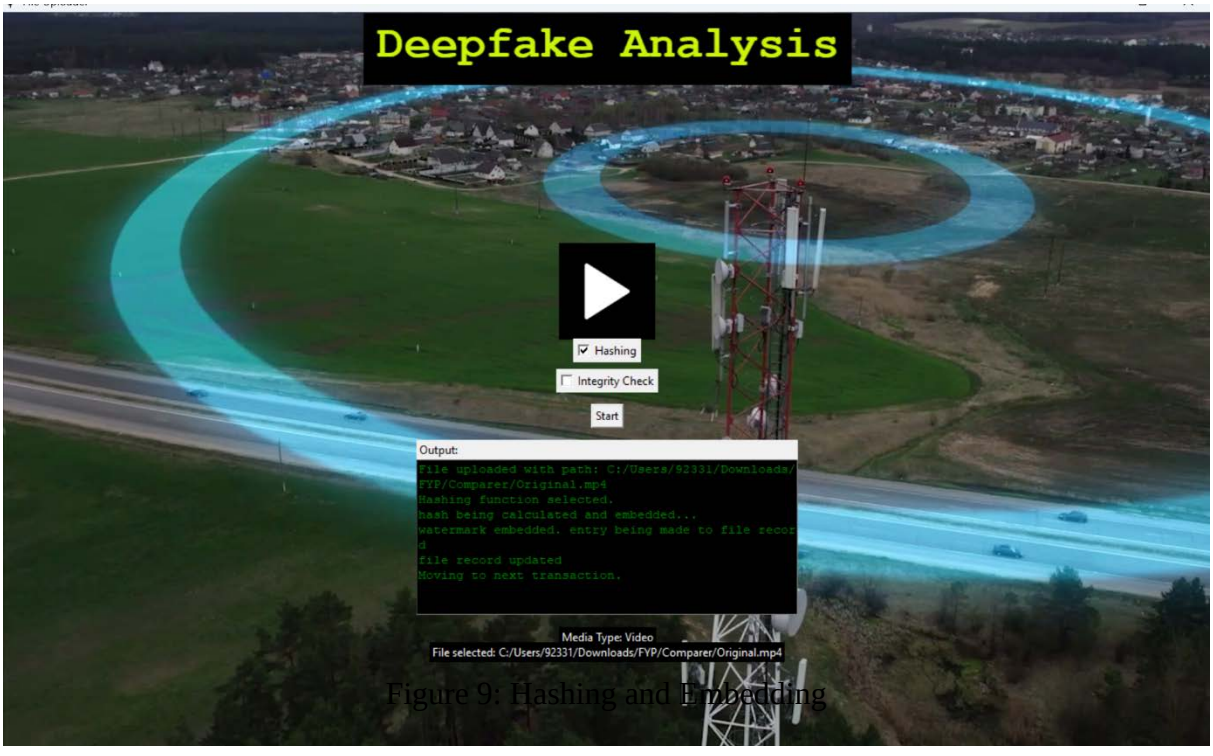


Figure 9: Hashing and Embedding

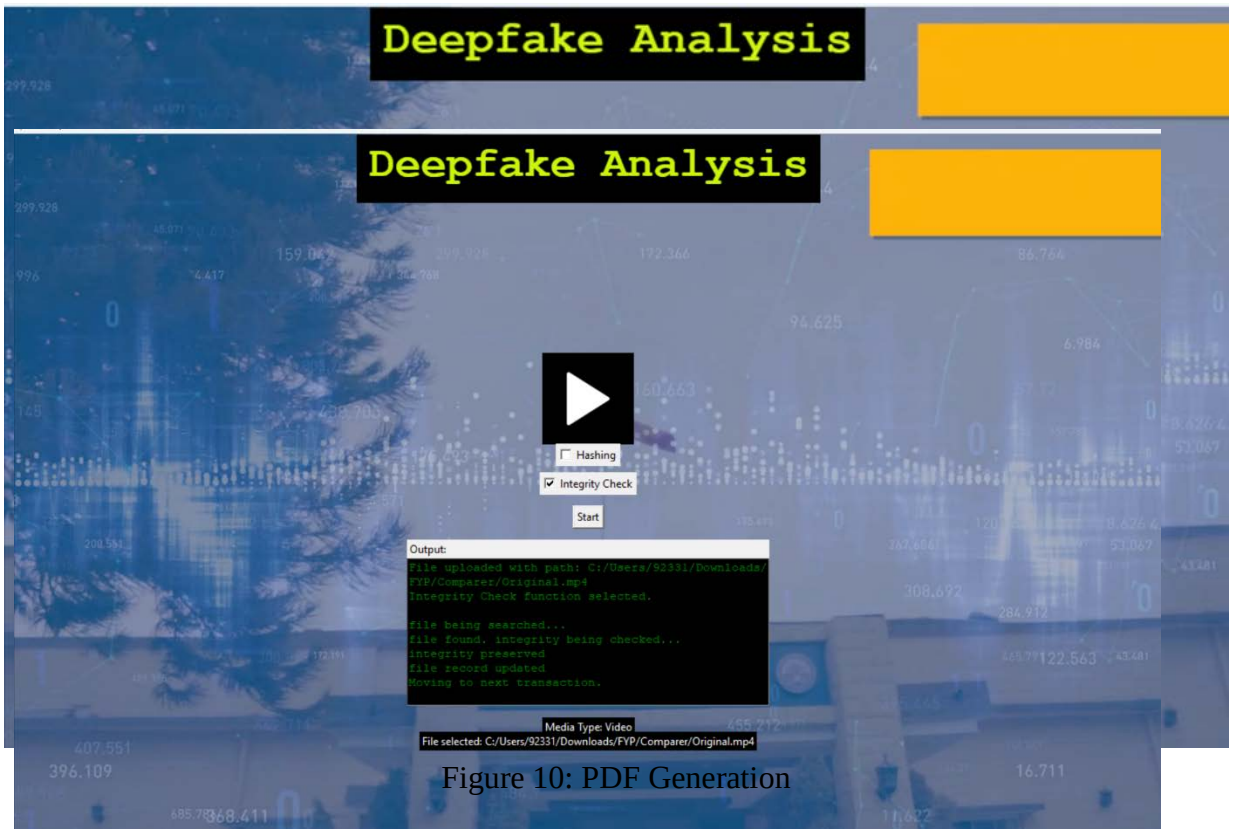


Figure 10: PDF Generation

Figure 11: Integrity Check

DEEPFAKE REPORT

Date of Generation
2024-05-20

Group Members
-GC Fahad Asif
-GC Ibad ur Rehman
-GC Ahmed Abdullah
-GC Ali Hassan Awan

Sr #	File_name	Hash_file	Last_integrity_check	Integrity_status
1	C:/deepfake_lab_fiaa/DSC_0556.JPG	C:/deepfake_lab_fiaa/DSC_0556.JPGupd1716152803.985421.jpg	2024-05-20 9:26:21	Fake detected
2	C:/deepfake_lab_fiaa/countdown_test.mp3	C:/deepfake_lab_fiaa/countdown_test.mp3_hashed.mp3	2024-05-20 9:15:19	Preserved
3	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	2024-05-20 9:47:43	Preserved
4	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
5	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
6	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
7	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
8	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
9	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
10	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
11	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked
12	C:/deepfake_lab_fiaa/test_video.mp4	C:/deepfake_lab_fiaa/test_video.mp4hashes.txt	none	not checked

Figure 12: PDF Report

Chapter 5

Conclusion

The paper outlines the Deep Fake Lab project in detail, explaining how it gives a multi-faceted framework to address deepfake content in different media formats. The use of metadata extraction mechanisms, cryptographic hashing algorithms, and the embedding of watermarks into digital media are used to enforce its integrity and authenticity. And the real-time processing capabilities allow effective prevention and timely identification of deepfake content as well as a tree reliable instrument in the areas of digital content integrity.

Upload and verification program makes the system user friendly and their media documents of different sort could be easily analyzed and verified by users with different levels of technical background. The PDF reports with the detailed description of the analysis are received by the user to make sure that no strange manipulations have been made and the result could be trusted.

With the use of these advanced techniques and at the same time designing the framework to be the scalable Deep Fake Lab model has been able to overcome some of the challenges hat traditional deepfake detection methods pose. Its module structure facilitates constant change and makes it possible for the system to always be one step ahead of the latest deep-fake trends.

Deep Fake Lab therefore helps immensely in the fight against the digital media integrity by offering a simple but efficient tool for dealing with deepfake content. This comprehensive approach effective in preventing false information from spreading as well as ensures that the concept of authenticity as in digital media among users are reinforced so that future digital environment is protected from false information.

Overall, the Deep Fake Lab project is an important initiative and would be valuable contribution to the study of digital media integrity as it provides an effective method to tell if the content being consumed is a fake or not in a short and cost – effective way. This strategy not only ensures that false details cannot be disseminated but also reinforces the need for the information to be genuine within the digital sphere in order to promote a safer and more dependable digital society.

Chapter 6

Future Work

The Deep Fake Lab project represents a promising approach for countering deepfake content and the risks it poses, but some improvements and further efforts are necessary to make it more effective and expand its functionality. Other milestones that this project have to be achieved in order to be commercialized are the following:

6.1 Access to real life logistics marketplace:

The primary objective of this project is to develop such a product that is also capable of making the process easier for everyone to use who is involved in this business. It is strategically important to have direct access to some logistical market.

6.2 Future Improvements:

6.2.1 Expanding to Real-time Streaming Media:

Extending capabilities to address real-time detection in the streaming media is critical to the approach for effectively dealing with threats as soon as they appear. This would mean improving the real-time processing module to effectively handle live streaming of videos and audio and is trained to address deepfakes as they enter the stream.

6.2.2 Improving User Interface and Experience:

While the current interface is designed to be user-friendly, there is always room for improvement. Future enhancements could include more intuitive navigation, better visualization of analysis results, and additional user guidance features. These would make the system more user friendly and even accessible to those who are less professionally or technically inclined.

6.2.3 Integration with Social Media Platforms:

Integrating the deepfake detection framework with popular social media platforms could significantly enhance its impact. This would allow for automatic detection and flagging of deepfake content on platforms where such content is most likely to spread rapidly. Collaborations with social media companies could facilitate the seamless integration of the detection system into their existing infrastructures.

6.2.4 Developing Mobile Applications:

Creating mobile applications for both Android and iOS platforms would increase the accessibility of the deepfake detection framework. Mobile apps would allow users to verify the authenticity of media content on the go, making it easier to combat the spread of deepfakes in real-time.

6.2.5 Addition of User Feedback:

The addition of the users' feedback can strengthen the deep fake detection module. For instance, users can report fake videos which failed to be identified as such and that data can be fed back to the algorithms to improve their accuracy.

6.2.6 Extending to Other Media Types:

While the current framework focuses on images, videos, and audio, future work could explore extending detection capabilities to other media types such as text and synthetic voices. This would provide a more comprehensive solution to the deepfake problem.

References and Work Cited

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

[2] Yuan Zhao, Bo Liu, Ming Ding, Baoping Liu, Tianqing Zhu, Xin Yu, “Proactive Deepfake Defence via Identity Watermarking” University of Technology Sydney, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023, (pp. 4602-4611)

[3] 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, Journal of Physics: Conference Series, 2021

[4] Zhang, S. Zheng, Y. Yuan, and Q. Zhao, “Detecting GAN-Generated Fake Images via Metadata Analysis”, in the IEEE Transactions on Information Forensics and Security, vol. 15, (pp. 666-678), 2020

[5] Adnan Alattar, Ravi Sharma, and John Scriven, “A System for Mitigating the Problem of Deepfake News Videos Using Watermarking”, Digimarc Corporation, 9405 SW Gemini Drive, Beaverton, OR, USA 97008-7192.

[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] R. Sharma, A. Jain, P. Kumar, S. Agrawal, and V. Srivastava, “Deepfake Video Detection using Image Processing and Hashing Tools,” Academia.edu, 2023.

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

[9] 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).

Appendix-A (Code)

PYTHON CODE USED IN PROJECT

Audio Analysis:

```
from mutagen import File
from mutagen.mp3 import MP3
from mutagen.id3 import ID3, TXXX
import hashlib
import shutil
def embed_hash(audio_file_path):
    hashed_audio_file_path = audio_file_path+"_hashed.mp3"
    shutil.copy(audio_file_path, hashed_audio_file_path)
    metadata=""
    # Load the audio file using Mutagen
    audio_metadata = File(hashed_audio_file_path)

    audio_file = MP3(hashed_audio_file_path, ID3=ID3)



    # Print audio metadata from Mutagen (e.g., tags)
    if audio_metadata:
        metadata=str(audio_metadata.mime) + str(audio_metadata.info.length) +
str(audio_metadata.info.bitrate) + str(audio_metadata.info.channels) +
str(audio_metadata.info.sample_rate)



        # Tags may include title, artist, album, etc.
        for tag in audio_metadata:
            print(f" {tag}-{audio_metadata[tag]}")
            metadata+=str(tag)
            metadata+=str(audio_metadata[tag])
    else:
        print("No metadata found using Mutagen")

    print(metadata)
    hash= hashlib.sha256(metadata.encode('utf-8')).hexdigest()
```

```
    print(hash)
    tag_id = 'hash'  # Custom tag ID
    tag_value = hash  # Custom tag value

    if audio_file.tags is None:
        # If no tag exists, create a new ID3 tag

        audio_file.tags = ID3()
    # Create a TXXX frame with the custom tag ID and value
    custom_tag = TXXX(encoding=3, desc=tag_id, text=tag_value)
    audio_file.tags.add(custom_tag)

    # Save the modified metadata
    audio_file.save()
    return hashed_audio_file_path,hash
def extract_compare_metadata(audio_file_path,hashed_audio_file_path):

    metadata=""
    # Load the audio file using Mutagen
    audio_metadata = File(hashed_audio_file_path)

    stored_hash=""
    # Print audio metadata from Mutagen (e.g., tags)
    if audio_metadata:
        print("Metadata from Mutagen:")
        print(f"File type: {audio_metadata.mime}")
        print(f"Sample rate: {audio_metadata.info.sample_rate} Hz")
        print(f"Bit rate: {audio_metadata.info.bitrate} bps")
        print(f"Channels: {audio_metadata.info.channels}")
        print(f"Duration: {audio_metadata.info.length} seconds")

        # Tags may include title, artist, album, etc.
        for tag in audio_metadata:
            print(f" {tag}-{audio_metadata[tag]}")
            metadata+=str(tag)
            metadata+=str(audio_metadata[tag])
            if "hash" in tag:
                stored_hash=audio_metadata[tag]
                print  (stored_hash)
    else:
        print("No metadata found using Mutagen")
```

```
    audio_metadata = File(audio_file_path)
    if audio_metadata:
        metadata=str(audio_metadata.mime) + str(audio_metadata.info.length) +
str(audio_metadata.info.bitrate) + str(audio_metadata.info.channels) +
str(audio_metadata.info.sample_rate)
        for tag in audio_metadata:
            print(f" {tag}-{audio_metadata[tag]}")
            metadata+=str(tag)
            metadata+=str(audio_metadata[tag])


    print(metadata)
    hash= hashlib.sha256(metadata.encode('utf-8')).hexdigest()
    print(hash)
    flag = True
    if hash == stored_hash:
        print( "integrity of audio file preserved. no modification detected.")
    else:
        flag = False
        print( "modification detected. audio has been changed.")
    return flag
```

Embed Check:

```
import piexif

import hashlib
from PIL import Image
from datetime import datetime

def embed_hash_in_image(image_path):
    img=Image.open(image_path)
    exif_dict = piexif.load(image_path)
    metadata=""

    # Print out the EXIF data in an organized mannner
    for ifd_name in exif_dict:
        for tag in exif_dict[ifd_name]:
            try:
                tag_name = piexif.TAGS[ifd_name][tag]["name"]
```

```
                tag_value = exif_dict[ifd_name][tag]
                # Decode tag values if necessary
                if isinstance(tag_value, bytes):
                    try:
                        tag_value = tag_value.decode('utf-8')
                    except UnicodeDecodeError:
                        pass

                metadata=metadata+f"{tag}: {tag_name} - {tag_value} "

            except KeyError:
                # Handle missing tag names
                print(f"{tag}: .....")

    hash = hashlib.sha256(metadata.encode('utf-8')).hexdigest()
    hash_tag_id = 37510  # tag ID for User Note. this tag has been selected for storing hash of image
metadata


    # Encode the value as UTF-8 bytes
    encoded_hash = hash.encode('utf-8')
    if "Exif" not in exif_dict:
        exif_dict["Exif"] = {}

    # adding hash to metadata
    exif_dict["Exif"][hash_tag_id] = encoded_hash
    exif_bytes = piexif.dump(exif_dict)
    time=datetime.now()
    timestamp=time.timestamp()
    file_path= image_path+"upd"+str(timestamp)+".jpg"
    img.save(file_path, exif=exif_bytes)
    print("hash embedded with tag ID 37510 as User comment in image at location:\n",file_path)
    return file_path,hash   # returns file path of image with embedded hash along with the hash

def hash_comparator(image_path,hashed_file_path):
    exif_dict = piexif.load(image_path)
    metadata = ""
    flag = True
    # traverse through the original metadata of raw file
    for ifd_name in exif_dict:
        for tag in exif_dict[ifd_name]:
            try:
```

```
                tag_name = piexif.TAGS[ifd_name][tag]["name"]
                tag_value = exif_dict[ifd_name][tag]
                # Decode tag values if necessary (e.g. user comments)
                if isinstance(tag_value, bytes):
                    try:
                        tag_value = tag_value.decode('utf-8')
                    except UnicodeDecodeError:
                        pass

                metadata = metadata + f"{tag}: {tag_name} - {tag_value} "

            except KeyError:
                # Handle missing tag names
                print(f"{tag}: .....")

    hash = hashlib.sha256(metadata.encode('utf-8')).hexdigest()
    hash_tag_id = 37510  # tag ID for User Note. this tag has been selected for storing hash of image metadata


    if "Exif" not in exif_dict:
        exif_dict["Exif"] = {}


    exif_dict = piexif.load(hashed_file_path)

    if "Exif" not in exif_dict:
        exif_dict["Exif"] = {}

    # extracting stored hash from the updated image corresponding to the raw image
    stored_hash=exif_dict["Exif"][hash_tag_id]

    if stored_hash==hash.encode("utf-8"):
        flag=True
    else:
        flag=False

    return flag
```

Upload:

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import os
import imageio
import video_hash_check
import audio_analysis
import image_embed_check
import excel_sheet_handling

file_record = r"C:\deepfake_lab_fiaa\file_record.xlsx"
filepath = ""
media_type = ""
global done

pdf_generated = False  # Flag to track whether PDF has been generated

def process(file_path, media_type, type):
    global pdf_generated
    if type == 'hash':
        if media_type == "Image":
            display("hash being calculated and embedded...\n")
            hashed_img_path,img_hash=image_embed_check.embed_hash_in_image(file_path)
            display("watermark embedded. entry being made to file record\n")
            excel_sheet_handling.entry(file_record, file_path, hashed_img_path)
            display("file record updated\n")
        elif media_type == "Video":
            display("hash being calculated and embedded...\n")
            hashed_vid_path = video_hash_check.embed_vid_hash_fbf(file_path)
            display("watermark embedded. entry being made to file record\n")
            excel_sheet_handling.entry(file_record, file_path, hashed_vid_path)
            display("file record updated\n")
        elif media_type == "Audio":
            display("hash being calculated and embedded...\n")
            hashed_aud_path , aud_hash= audio_analysis.embed_hash(file_path)
            display("watermark embedded. entry being made to file record\n")
            excel_sheet_handling.entry(file_record, file_path, hashed_aud_path)
            display("file record updated\n")
        else:
            display("filetype unknown\n")
```

```
elif type == 'integrity':
    display("file being searched...\n")
    if media_type == "Image":
        found,flag,hashed_file,coords=excel_sheet_handling.search_file(file_path,file_record)
        if flag:
            display("file found. integrity being checked...\n")
            integrity=image_embed_check.hash_comparator(file_path,hashed_file)
            if integrity:
                display("integrity preserved\n")
            else:
                display("deepfake detected\n")
            excel_sheet_handling.integrity_update(file_record,file_path,integrity)
            display("file record updated\n")
        else:
            display("file not found in record\n")
    elif media_type == "Video":
        found, flag, hashed_file, coords = excel_sheet_handling.search_file(file_path, file_record)
        if flag:
            integrity,time = video_hash_check.verify_frame_hashes(file_path,hashed_file)
            display("file found. integrity being checked...\n")
            if integrity:
                display("integrity preserved\n")
            else:
                display("deepfake detected at time"+str(time)+"\n")
            excel_sheet_handling.integrity_update(file_record,file_path,integrity)
            display("file record updated\n")
        else:
            display("file not found in record\n")
    elif media_type == "Audio":
        found, flag, hashed_file, coords = excel_sheet_handling.search_file(file_path, file_record)
        if flag:
            display("file found. integrity being checked...\n")
            integrity = audio_analysis.extract_compare_metadata(file_path,hashed_file)
            if integrity:
                display("integrity preserved\n")
            else:
                display("deepfake detected\n")
            excel_sheet_handling.integrity_update(file_record, file_path, integrity)
            display("file record updated\n")
        else:
            display("file not found in record\n")
    else:
```

```
            display("unknown media type\n")


    def open_file_dialog():
        global filepath, media_type
        filepath = filedialog.askopenfilename(title="Select a file")
        if filepath:
            output_label.config(text="File selected: " + filepath)
            # Determine the type of media
            media_type = determine_media_type(filepath)
            media_type_label.config(text="Media Type: " + media_type)
            response = "File uploaded with path: {}\n".format(filepath)
            terminal_text.insert(tk.END, response)


    def determine_media_type(filepath):
        _, file_extension = os.path.splitext(filepath)
        file_extension = file_extension.lower()

        if file_extension in ('.jpg', '.jpeg', '.png', '.gif', '.bmp'):
            return "Image"
        elif file_extension in ('.mp4', '.avi', '.mkv', '.mov', '.wmv'):
            return "Video"
        elif file_extension in ('.mp3', '.wav', '.flac', '.aac'):
            return "Audio"
        else:
            return "Unknown"

    def play_video():
        global filepath
        video_path = r"C:\deepfake_lab_fiaa\gui_elements\Changed Voice over.mp4" # Update with
    correct file path
        reader = imageio.get_reader(video_path)

        def update_frame():
            try:
                frame = reader.get_next_data()
                frame = Image.fromarray(frame)
                photo = ImageTk.PhotoImage(frame)
                background_label.config(image=photo)
                background_label.image = photo
                root.after(10, update_frame)  # Update every 10 milliseconds (increase frame rate)
```

```
            except StopIteration:
                reader.close()

        update_frame()

    def integrity_check(filepath, media_type):
        # Implement Integrity Check function here
        response = "Integrity Check performed for file: {}\n".format(filepath)
        terminal_text.insert(tk.END, response)
        integrity_flag = True  # Example flag for integrity check
        return integrity_flag

    def hashing(filepath, media_type):
        # Implement Hashing function here
        response = "Hashing performed for file: {}\n".format(filepath)
        terminal_text.insert(tk.END, response)
        hashing_flag = False  # Example flag for hashing
        return hashing_flag

    def generate_pdf():
        global pdf_generated

        # Logic to generate PDF
        report=excel_sheet_handling.generate_report(file_record)
        messagebox.showinfo("PDF Generation", "PDF generated successfully at location"+report)
        pdf_generated = True


    def start_processing():
        global filepath, media_type, done
        type=''
        if filepath and media_type:
            if integrity_check_var.get() == 1:
                response = "Integrity Check function selected.\n"
                terminal_text.insert(tk.END, response+"\n")
                type='integrity'
            if hashing_var.get() == 1:
                response = "Hashing function selected.\n"
                terminal_text.insert(tk.END, response)
                type='hash'
            process(filepath, media_type, type)
```

```
        # Check if PDF generation is required
        generate_pdf_prompt()

def generate_pdf_prompt():
    response = messagebox.askyesno("Application Prompt", "Do you want to generate PDF?")
    if response == True:
        generate_pdf()
    else:
        continue_without_pdf()


def continue_without_pdf():
    # Logic to continue without generating PDF
    response = "Moving to next transaction.\n"
    terminal_text.insert(tk.END, response)


def display(response):
    terminal_text.insert(tk.END, response)


root = tk.Tk()
root.title("File Uploader")

# Play video in the background
background_label = tk.Label(root)
background_label.place(relx=0.5, rely=0.5, anchor="center")

# Load the image for the heading
heading_img = Image.open(r"C:\deepfake_lab_fiaa\gui_elements\output-onlinepngtools.png")
heading_photo = ImageTk.PhotoImage(heading_img)

# Heading label
heading_label = tk.Label(root, image=heading_photo, bg="black")
heading_label.photo = heading_photo  # Keep a reference
heading_label.place(relx=0.5, rely=0.05, anchor="center")

# Load the image for the upload button
upload_img = Image.open(r"C:\deepfake_lab_fiaa\gui_elements\—Pngtree—vector play button
icon_4258876.png")  # Update with correct file path
upload_img = upload_img.resize((100, 100))  # Resize the image
upload_photo = ImageTk.PhotoImage(upload_img)
```

```
# Upload button
upload_button = tk.Button(root, image=upload_photo, command=open_file_dialog, bd=0)
upload_button.configure(bg='black')  # Set background to white
upload_button.place(relx=0.5, rely=0.38, anchor="center")

# other GUI elements
output_label = tk.Label(root, text="", fg="white", bg="black")
output_label.place(relx=0.5, rely=0.87, anchor="center")

media_type_label = tk.Label(root, text="", fg="white", bg="black")
media_type_label.place(relx=0.5, rely=0.85, anchor="center")

# Checkboxes for Integrity Check
integrity_check_var = tk.IntVar()
integrity_check_checkbox = tk.Checkbutton(root, text="Integrity Check",
variable=integrity_check_var, onvalue=1, offvalue=0)
integrity_check_checkbox.place(relx=0.5, rely=0.5, anchor="center")

# Checkboxes for Hashing
hashing_var = tk.IntVar()
hashing_checkbox = tk.Checkbutton(root, text="Hashing", variable=hashing_var, onvalue=1,
offvalue=0)
hashing_checkbox.place(relx=0.5, rely=0.46, anchor="center")

# Start button
start_button = tk.Button(root, text="Start", command=start_processing)
start_button.place(relx=0.5, rely=0.55, anchor="center")

# Mini terminal box
terminal_frame = tk.Frame(root)
terminal_frame.place(relx=0.5, rely=0.7, anchor="center")
terminal_label = tk.Label(terminal_frame, text="Output:")
terminal_label.grid(row=0, column=0, sticky="w")
terminal_text = tk.Text(terminal_frame, height=10, width=50, bg="black", fg="green")
terminal_text.grid(row=1, column=0, sticky="ew")

# Make the heading label draggable
root.photo_label = heading_label

root.after(10, play_video)  # Start video playback after 10 milliseconds
root.mainloop()
```

Video Hash Check:

```
import cv2
import hashlib

def embed_vid_hash_fbf(video_path):
    hash_file_path = video_path + "hashes.txt"

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

    if not cap.isOpened():  # Check if the video file was opened successfully
        print("Could not open video file")
    else:

        with open(hash_file_path, "a") as hash_file:    # Open a file to write frame hashes to a txt
document
            frame_number = 0
            while True:

                ret, frame = cap.read() # Read the next frame
                if not ret:
                    break

                frame_hash = hashlib.sha256(frame.tobytes()).hexdigest()    # Calculate the hash of the
frame data
                hash_file.write(f"{frame_number},{frame_hash}\n")   # Save the hash and frame number
to the file
                frame_number += 1   # Increment the frame counter
            cap.release()   # Release the video capture object when done
        hash_file.close()


    return hash_file_path


def verify_frame_hashes(video_path, hash_file_path):

    flag=True   # set flag to identify integrity of video
```

```
cap = cv2.VideoCapture(video_path)
fps= cap.get(cv2.CAP_PROP_FPS)  # get the fps for pinpointing time of alteration

stored_hashes = {}
with open(hash_file_path, "r") as hash_file:    # Read stored frame hashes
   for line in hash_file:
      frame_number, frame_hash = line.strip().split(",")
      stored_hashes[int(frame_number)] = frame_hash
time = 957810
# Check each frame hash against the stored hashes
frame_number = 0
while True:
   # Read the next frame from the video
   ret, frame = cap.read()

   # If the frame was not read successfully, break the loop
   if not ret:
      break

   # Calculate the hash of the frame data
   current_frame_hash = hashlib.sha256(frame.tobytes()).hexdigest()

   # Check the hash against the stored hash
   if frame_number in stored_hashes and current_frame_hash != stored_hashes[frame_number]:
      print(f"Frame {frame_number} has been altered. \n")
      time=int((frame_number+1)/fps)
      print(f"Alteration occured at time={time}s\n")
      flag=False
   # Increment the frame counter
   frame_number += 1

cap.release()

if flag==True:
   print("The video has not been modified")
else:
   print(f"The video is altered at time={time}s")
return flag,time
```

Excel Sheet Handling:

```
#from openpyxl import Workbook
from openpyxl import load_workbook
import win32com.client
import datetime



def entry(file_record,file_path,hashed_file_path):
    wb = load_workbook(file_record)
    # Select the worksheet
    ws = wb.active
    # Find the last row with data
    last_row = ws.max_row
    sr_no=str(last_row)

    data=[]
    data.append(sr_no)
    data.append(file_path)
    data.append(hashed_file_path)
    data.append("none")
    data.append("not checked")
    # Append data to the next available row
    ws.append(data)
    # Save the workbook
    wb.save(file_record)

def search_file(file_name,file_record):
    wb = load_workbook(file_record)
    ws = wb.active
    flag=False
    hashed_file=''
    found = []
    coords=[]
    # Iterate over all cells and search for the keyword
    for row in ws.iter_rows():
        for cell in row:
            if cell.value == file_name:
                flag=True
                #print("found")
                for cel in row:
                    found.append(cel.value)
```

```
                coords.append(cel)
    if flag==True:
        hashed_file=found[2]
    return found,flag,hashed_file,coords


def integrity_update(file_record,file_path,flag):
    #check the file path in excel sheet

    found,f_flag,hashed_file,coords=search_file(file_path,file_record)
    print (coords)
    wb = load_workbook(file_record)
    ws = wb.active

    if f_flag == True:
        ws[coords[3].coordinate]=datetime.datetime.now()    # datetime.datetime.now() will give
timestamp
        if flag == True:    # the integrity flag taken from comparison procs beforehand will determine
integrity status
            ws[coords[4].coordinate]="Preserved"
        else:
            ws[coords[4].coordinate]="Fake detected"

        wb.save(file_record)    # all info written to SPECIFIC CELLS


def generate_report(file_record):
    report = file_record + "_report.pdf"
    excel = win32com.client.Dispatch("Excel.Application")
    # Open the Excel file
    workbook = excel.Workbooks.Open(file_record)
    # Access the first sheet (you can modify this according to your sheet selection)
    sheet = workbook.Sheets(1)
    sheet.PageSetup.Orientation = 2
    sheet.PageSetup.LeftMargin = 20  # Left margin: 0.5 inch
    sheet.PageSetup.RightMargin = 20  # Right margin: 0.5 inch
    sheet.PageSetup.TopMargin = 90  # Top margin: 0.5 inch
    sheet.PageSetup.BottomMargin = 20  # Bottom margin: 0.5 inch
    sheet.UsedRange.Borders.Weight = 1.5  # Weight 2 represents thick borders
    sheet.PageSetup.CenterHeader = "&\"Arial,Bold\"&44DEEPFAKE REPORT"  # Add a centered
header with the title "Title"
    # Export the sheet as PDF
```

```
sheet.ExportAsFixedFormat(0, report)
# Close Excel
workbook.Close(False)
excel.Quit()
return report
```