BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous Institute under VTU, Belagavi | Approved by AICTE,New Delhi Recognized by Govt. of Karnataka

NACC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belgavi)
"Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,
Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

A Project Report On

AI-POWERED DEEPFAKE DETECTION: LIVENESS DETECTION

A dissertation submitted to the Department of CSE (Data Science) of Visvesvaraya Technological University in partial fulfillment for the Degree of Bachelor of Engineering award.

Project Associates:

CHETANA HK	**3BR22CD007**
K SHAHSIKALA	**3BR22CD023**
MOUNIKA M	**3BR22CD039**
PUSHPITHA J R	**3BR22CD046**

Under the Guidance of

Dr. Jagadish R M

Professor

Dept of CSE (DATA SCIENCE)

BITM, Ballari.

Visvesvaraya Technological University

Belagavi, Karnataka

2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous Institute under VTU, Belagavi | Approved by AICTE,New Delhi Recognized by Govt. of Karnataka

NACC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belgavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,
Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the project work entitled "**AI-POWERED DEEPFAKE DETECTION: LIVENESS DETECTION**" is a Bonafide work carried out by Chetana HK(3BR22CD007), K Shashikala(3BR22CD023), Mounika M (3BR22CD039), Pushpitha J R (3BR22CD046) in partial fulfillment for the award of degree of **Bachelor Degree in CSE (DATA SCIENCE)** in the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2025-2026. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The project has been approved as it satisfies the academic requirements in respect of major project phase-2 work prescribed for a Bachelor of Engineering Degree.

Signature of Project guide	Signature of HOD	Signature of Principal
Dr. Jagadish R M	Dr. Aradhana D	Dr. Yadavalli Basavaraj

External Viva

Name of the Examiner	Signature with Date
1.	
2.	

ABSTRACT

Advances in deep learning have made it easier to create realistic visual content, popularly known as deepfakes. While such tools find applications in creative media, the serious risks of identity theft, misinformation, and digital manipulation also come with them. This work presents a lightweight detection framework that identifies manipulated visual content and verifies whether the source is a real live person. The system proposed here is empowered with both MobileNet and custom CNN models to analyze facial behavior, expression dynamics, and minute texture variations that distinguish genuine recordings from spoofing attempts created using printed images, masks, or replayed clips. For real-time processing of both images and videos, a web-based interface is developed using Flask. Experimental evaluations demonstrate accuracy close to 90%, thus extending the applicability of the proposed solution to secure authentication environments and digital forensics.

ACKNOWLEDGEMENT

We express our warm and profound sense of gratitude to all the eminent faculties who inspired, guided and supported us in accomplishing our project work.

We are deeply indebted to **Dr. Jagadish R M**, Department of CSE (Data Science), our guide on this project & coordinators **Dr. Aradhana D,** on this project, for consistently providing us with the required guidance to help us in the timely and successful completion of this project. Despite her hectic schedules in the Department, she was always available to share her deep insights, wide knowledge, and extensive experience with us.

We are much obliged to our honorable Head of the Department **Dr. Aradhana D** for understanding the capability in us to complete the project successfully and, we are thankful to her for encouraging us and being a motivation and inspiration for us.

We extend our warm thanks to our Principal **Dr. Yadavalli Basavaraj** for giving us moral support and providing us all kinds of facilities.

We also thank all the faculty members of our department for being with us when needed. Last but not least we also thank our beloved friends who were there with us all the time, supporting, providing assistance, and giving us enough strength to successfully complete our project, proving the popular saying, "A Friend in need is a Friend indeed" to be true.

NAME	**USN**
CHETANA HK	**3BR22CD007**
K SHAHSIKALA	**3BR22CD023**
MOUNIKA M	**3BR22CD039**
PUSHPITHA J R	**3BR22CD046**

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous Institute under VTU, Belagavi | Approved by AICTE,New Delhi Recognized by Govt. of Karnataka

NACC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belgavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,
Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

CSE (DATA SCIENCE)

DEPARTMENT VISION AND MISSION

VISION:

To Transform into a hub of excellence in Data Analytics, cultivating adept professionals in the fields of data analytics and research.

MISSION:

M1: To Empower students with innovative and cognitive skills to excel in the field of Data Science.

M2: To Provide exceptional infrastructure, facilities, and ambiance for the nurturing of young professional.

M3: To Cultivate knowledge through an industry-friendly environment, enabling excellence and research in a data-driven world.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background of Project

The rapid growth of digital media and social networking platforms has made images and videos a primary source of communication and information sharing. With advancements in Artificial Intelligence (AI) and deep learning, the creation of highly realistic synthetic media, known as *deepfakes*, has become easier and more accessible. Deepfakes use advanced neural networks to manipulate or generate fake images and videos that appear authentic, making it difficult for humans to distinguish between real and fake content.

While these technologies have positive applications in entertainment, education, and virtual communication, their misuse has raised serious concerns. Deepfakes are increasingly used for spreading misinformation, identity fraud, political manipulation, cybercrime, and reputational damage. Traditional security systems and manual verification methods are no longer sufficient to detect such sophisticated manipulations, especially when attackers use high-quality fake videos or replay attacks.

To address these challenges, Artificial Intelligence and Computer Vision techniques have emerged as effective solutions for detecting deepfake content. However, detecting deepfakes alone is not enough, as attackers can still bypass systems using static images, recorded videos, or masks. This limitation highlights the need for **liveness detection**, which ensures that the input comes from a real, live human being rather than a spoofed source.

This project proposes an **AI-Powered Deepfake Detection with Liveness Detection system** that combines deep learning models and facial analysis techniques to identify manipulated media and verify real-time human presence. The system analyzes facial features, eye blinking patterns, head movements, texture inconsistencies, and visual artifacts to accurately classify input as real or fake. It supports both image and video inputs, making it suitable for real-world security applications.

The system is designed using a modular and scalable architecture consisting of:

- **Frontend:** HTML and CSS for a simple and user-friendly interface.
- **Backend:** Python with Flask for secure request handling and processing.
- **AI Engine:** Deep learning models using TensorFlow and OpenCV for deepfake and liveness detection.
- **Storage & Processing:** Structured datasets and temporary storage for uploaded media.

This well-structured and efficient design ensures accuracy, reliability, and ease of use. As digital media continues to dominate communication and authentication systems, AI-powered deepfake and liveness detection tools play a crucial role in enhancing digital security, preserving media authenticity, and building trust in modern information systems.

1.2 Problem Context

The digital world increasingly relies on images and videos for communication, authentication, and information sharing. Social media platforms, online meetings, surveillance systems, and digital identity verification generate and consume massive volumes of visual data every day. However, with the rapid advancement of Artificial Intelligence, creating highly realistic fake images and videos—known as deepfakes—has become easier, faster, and more convincing. Identifying whether visual content is real or manipulated has therefore become a major challenge.

Existing security systems and media verification tools are often limited in their ability to detect sophisticated deepfakes. Many current detection methods focus only on identifying manipulated visuals but fail to verify whether the input is coming from a real, live human being. This limitation allows attackers to bypass systems using printed photos, recorded videos, or AI-generated content, making traditional verification techniques unreliable and error-prone.

Most available deepfake detection tools also lack real-time analysis and user-friendly interaction. They do not provide detailed confidence scores, visual explanations, or easy-to-use interfaces for non-technical users. Furthermore, many systems do not store previous analysis results or reports, making it difficult for users to track, compare, or review past detections. This reduces their usefulness in professional environments such as banking, law enforcement, remote authentication, and digital forensics.

Users and organizations require a smarter system that not only detects deepfake content but also ensures **liveness detection**, confirming the presence of a real human through facial movements, eye blinking, and behavioral patterns. Such a system should support both images and videos, deliver fast and accurate results, and provide clear feedback that can be easily understood by users.

Considering these needs and existing gaps, this project aims to develop an **AI-Powered Deepfake Detection with Liveness Detection system** that is accurate, interactive, and reliable. By combining deep learning models with facial and behavioral analysis, the proposed system enhances digital security, reduces identity fraud, and improves trust in visual media. This solution offers a modern and efficient approach to tackling deepfake threats in today's increasingly digital environment.

Existing tools for detecting manipulated images and videos are often unable to accurately identify advanced deepfake content or verify whether the input comes from a real, live human being. Many current systems rely only on basic visual analysis, which fails to capture subtle facial inconsistencies, behavioral patterns, and temporal cues introduced by sophisticated deepfake generation techniques. As a result, important details are missed, leading to incorrect classification and security vulnerabilities.
Most available deepfake detection tools do not include liveness detection mechanisms, allowing attackers to bypass systems using static images, recorded videos, or replay attacks. In addition, these tools often lack real-time interaction, clear confidence reporting, and user-friendly interfaces, making them difficult for non-technical users to understand and trust. Support for both image and video inputs is also limited in many existing solutions.
To address these limitations, this project aims to develop an **AI-powered deepfake detection system with liveness detection** that accurately identifies manipulated media while ensuring real-time human presence. The proposed system uses deep learning and computer vision techniques to analyze facial features, eye movements, texture patterns, and motion behavior. By providing reliable detection, fast results, and an easy-to-use

interface, the system enhances digital security and helps prevent identity spoofing and misinformation.

1.3 Problem Statement

To develop an AI-powered system capable of detecting deepfake content and verifying real- time human presence through liveness detection, to enhance digital security, ensure media authenticity, and prevent identity spoofing attacks.

1.4 Motivation of the Project

In recent years, the rapid growth of digital media platforms, online communication, and remote authentication systems has created a strong need for intelligent solutions that can verify the authenticity of images and videos. People increasingly rely on visual content for information sharing, identity verification, and decision-making, but the rise of advanced deepfake technology has made it difficult to trust what is real. Understanding whether visual content is genuine or manipulated has become a major challenge, especially when fake media is created using sophisticated AI techniques.

Existing deepfake detection tools often provide limited accuracy and fail to ensure that the input comes from a real, live human being. Many systems do not analyze facial behavior, eye movements, or motion patterns in real time, which makes them vulnerable to spoofing attacks using photos or recorded videos. These limitations highlight the need for a system that preserves visual authenticity, detects manipulation, and verifies human presence simultaneously.

This project is developed to address these gaps by proposing an **AI-powered deepfake detection system with liveness detection**. By combining deep learning models with computer vision techniques such as facial analysis, texture inspection, and motion tracking, the system provides a more accurate, secure, and user-friendly solution. This topic is chosen because it addresses a critical real-world problem and has wide applications in digital security, social media, online authentication, and cybercrime prevention, making it highly relevant in today’s technology-driven society.

1.5 Objectives of the Project

The main objective of this project is to develop an **AI-powered deepfake detection system with integrated liveness detection** that enhances digital security by accurately identifying manipulated images and videos while verifying real-time human presence. The specific objectives of the project include:

- **To design and implement an intelligent deepfake detection system** capable of identifying synthetically generated or manipulated images and videos without losing critical visual details.
- **To incorporate liveness detection mechanisms** that analyze facial movements, eye blinking, head motion, and texture patterns to ensure the presence of a real, live human.
- **To enable both image and video-based detection** for comprehensive analysis of different types of deepfake and spoofing attacks.
- **To provide accurate classification and confidence scores** that clearly indicate whether the input media is real or fake, enhancing user trust and transparency.
- **To develop a secure and scalable web-based platform** using modern frontend and backend technologies for efficient media handling and real-time analysis.
- **To store detection reports and analysis history** allowing users to retrieve previous results and monitor deepfake trends over time.
- **To improve the reliability of digital authentication systems** by preventing identity spoofing, replay attacks, and AI-generated media fraud.

1.6. Scope of the Project

This project focuses on the design and implementation of an **AI-powered deepfake detection system with liveness detection** to help users identify manipulated images and videos efficiently. The system analyzes visual content to detect deepfake artifacts and verifies real-time human presence by examining facial movements, eye blinking, head motion, and texture inconsistencies. It combines computer vision and deep learning techniques to provide accurate and reliable detection results through a web-based application.

The system integrates AI and image-processing modules with a **Python (Flask) backend**, a simple and user-friendly **HTML/CSS frontend**, and structured storage for uploaded media and detection reports. Users can upload images or videos, receive instant classification results (Real or Fake), view confidence scores, and access previously analyzed files for quick reference. The project emphasizes fast response time, clear output presentation, and ease of use for both technical and non-technical users.

The project also includes designing an intuitive interface, delivering near real-time detection feedback, and maintaining a history of detection results for better tracking and analysis. It aims to enhance digital trust and security by preventing identity spoofing and misuse of AI-generated media.

CHAPTER 2

LITERATURE SURVEY

In paper **[1]**, titled **“Deepfake Detection Using Adversarial Neural Networks”** (2025), the authors propose an adversarial learning-based framework to identify forged facial images and videos by detecting inconsistencies in facial textures and spatial features. The study utilizes adversarial neural networks to improve robustness against manipulation techniques and demonstrates improved detection accuracy compared to traditional CNN-based approaches. Experimental results show effective identification of common deepfake artifacts under controlled conditions. However, the model performance decreases significantly when tested on highly realistic and unseen deepfake samples. Additionally, the system does not incorporate liveness detection mechanisms, making it vulnerable to replay and spoofing attacks. These limitations motivate the need for a combined deepfake and liveness detection approach as proposed in this project.

In paper **[2]**, titled **“Identifying Deepfake Cyber Attacks: Challenges and Countermeasures”** (2025), the authors analyze the growing threat of deepfake-based cybercrime and discuss AI-driven forensic and biometric verification techniques to counter such attacks. The paper highlights the importance of integrating facial analysis, behavioral cues, and biometric validation for improved security. While the study provides a comprehensive overview of existing countermeasures, it remains largely theoretical and lacks extensive real-world implementation or performance evaluation. The absence of a practical, real-time detection framework and liveness verification highlights a gap that the proposed system aims to address.

In paper **[3]**, titled **“A Survey: Deepfake and Current Technologies for Solutions”** (2024), the authors present a detailed survey of deepfake generation methods and detection techniques, including CNN-based models, watermarking strategies, and forensic analysis. The study compares multiple detection approaches and identifies challenges such as dataset bias and generalization issues. Although the survey offers valuable insights, it reports poor performance of existing models on unseen deepfake types and does not focus on liveness detection as a defense mechanism. This gap emphasizes the importance of integrating behavioral and motion-based analysis, which is a key feature of the proposed project.

In paper **[4]**, titled **“Deepfake Detection Using CNN and Temporal Analysis”** (2023), the authors investigate the use of convolutional neural networks combined with temporal feature analysis to identify inconsistencies across video frames. The model captures unnatural facial movements, blinking patterns, and frame-level artifacts to improve detection accuracy. Experimental results show improved performance over single-frame models. However, the approach is computationally expensive and not suitable for real-time deployment. Moreover, it lacks explicit liveness verification to ensure the presence of a real human. These limitations highlight the need for a lightweight and real-time system, as implemented in this project.

In paper **[5]**, titled **“Face Liveness Detection for Anti-Spoofing Systems”** (2021), the authors explore various liveness detection techniques such as eye-blink detection, head movement analysis, and texture-based classification to prevent spoofing attacks. The study demonstrates that liveness detection significantly improves security in facial authentication systems. However, the methods are tested mainly against static image attacks and do not address AI-generated deepfake videos. The lack of integration between deepfake detection and liveness verification underlines the novelty of this project, which combines both techniques into a unified AI-powered system.

In paper **[6]**, titled **“FaceForensics++: Learning to Detect Manipulated Facial Images”** (2019), the authors introduce a large-scale benchmark dataset containing various types of facial manipulations such as face swapping, expression reenactment, and identity replacement. The study evaluates multiple convolutional neural network architectures to detect manipulated images and videos. The results demonstrate that deep learning models can effectively identify deepfake artifacts when trained on large, diverse datasets. However, the authors report a significant drop in detection accuracy when models are tested on unseen datasets or heavily compressed videos. The work also does not include liveness detection to differentiate between live subjects and replayed or synthetic media. These limitations support the need for a system like the proposed project, which combines robust detection with liveness verification.

In paper **[7]**, titled **"Detecting Deepfakes via Eye Blinking Patterns"** (2018), the authors propose a novel liveness-based approach that analyzes abnormal eye-blinking behavior in deepfake videos. The method uses facial landmark detection and temporal analysis to identify unrealistic blink rates, which are often missing in AI-generated videos. Experimental results show promising performance in identifying early-generation deepfakes. However, the technique becomes less effective when deepfake models learn to mimic realistic blinking patterns. Additionally, the approach alone cannot handle image-based attacks or advanced video manipulations. This highlights the importance of integrating eye-blink analysis with broader deepfake detection models, as done in the proposed system.

In paper **[8]**, titled **"Deepfake Video Detection Using CNN-LSTM Networks"** (2022), the authors present a hybrid deep learning architecture combining convolutional neural networks for spatial feature extraction and long short-term memory (LSTM) networks for temporal sequence analysis. The model captures frame-level artifacts as well as temporal inconsistencies across video sequences. The experimental evaluation shows improved detection accuracy compared to frame-based CNN models. However, the system requires high computational resources and long video sequences, making it unsuitable for real-time applications. The model also lacks an explicit liveness detection mechanism. These drawbacks motivate the design of a lightweight, real-time deepfake and liveness detection system as proposed in this project.

CHAPTER 3

EXISTING SYSTEM AND PROPOSED SYSTEM

3.1 System Analysis

3.1.1 Introduction

System analysis is the structured process of studying and evaluating existing media verification and security systems to identify their limitations, operational challenges, and areas that require improvement. It involves examining how current deepfake detection tools function, assessing their accuracy and response time, understanding how spoofing attacks occur, and analyzing the shortcomings that prevent these systems from providing reliable and secure performance. Through this analytical approach, the root causes of inaccurate detection, delayed response, and lack of real-time verification are clearly identified.

The purpose of system analysis is to ensure that the proposed solution effectively addresses real-world security requirements. By understanding user expectations, attack scenarios, system constraints, and operational environments, system analysis helps define clear objectives and functional requirements for the new system. It also determines the scope, feasibility, and limitations of the project, ensuring that the final solution is accurate, efficient, and practical for real-time deployment.

In the context of this project, system analysis focuses on the shortcomings of existing deepfake detection tools, such as their inability to verify real-time human presence, lack of liveness detection, poor generalization across different deepfake techniques, and limited user interaction. Many existing systems detect manipulated content only at a surface level and fail to analyze facial behavior, motion patterns, or spoofing attempts. These limitations strongly motivate the development of an **AI-Powered Deepfake Detection with Liveness Detection system** that integrates deep learning, computer vision, and behavioral analysis into a single, reliable solution.

3.1.2 Existing System

The existing systems used for deepfake detection mainly rely on basic image or video analysis techniques that focus on identifying visual artifacts in manipulated media. While these systems can detect some forms of deepfake content, they often struggle with highly realistic or newly generated deepfakes. Most existing tools analyze individual frames without considering temporal consistency, facial behavior, or motion patterns, which reduces detection accuracy.

Current systems generally do not include liveness detection mechanisms. As a result, attackers can bypass these systems using printed images, pre-recorded videos, or replay attacks. Without verifying real-time human presence, existing solutions remain vulnerable to spoofing and identity fraud. Additionally, many systems are not optimized for real-time processing, making them unsuitable for practical security applications.

Another major limitation of existing systems is the lack of user-friendly interfaces and clear result interpretation. Many tools do not provide confidence scores, visual feedback, or easy upload mechanisms for users. They also lack proper storage and report management, preventing users from reviewing previous detection results. Secure user authentication and history tracking are often missing.

Due to these limitations—such as low generalization capability, absence of liveness detection, lack of real-time analysis, and poor usability—existing deepfake detection systems are not fully reliable for real-world deployment. This creates a strong need for a smarter and more comprehensive system that combines deepfake detection with liveness verification and user-friendly interaction.

3.1.3 Proposed System

The proposed system introduces an **AI-Powered Deepfake Detection with Liveness Detection** solution to overcome the limitations of existing media verification tools. Instead of relying only on basic visual inspection, the system uses advanced deep learning and computer vision techniques to detect manipulated images and videos while verifying real-time human presence. It analyzes facial features, texture inconsistencies, eye blinking patterns, head movements, and temporal behavior to accurately determine whether the input media is real or fake.

The system supports both image and video inputs, enabling comprehensive detection across different types of deepfake and spoofing attacks. Liveness detection plays a crucial role by ensuring that the input originates from a live human and not from static images, recorded videos, or replay attacks. The system provides clear classification results along with confidence scores, improving transparency and user trust.

The proposed solution is implemented using a modular web-based architecture. A **user-friendly frontend** built with HTML and CSS allows users to upload media files and view results easily. The **backend**, developed using Python and Flask, handles request processing, face detection, and communication with the deep learning models. The **AI engine**, built using TensorFlow and OpenCV, performs deepfake detection and liveness analysis. Uploaded media files and detection results are securely stored, enabling users to access previous reports when needed.

This architecture ensures fast response time, reliable performance, and scalability for future enhancements. By integrating deepfake detection, liveness verification, real-time processing, and an intuitive interface into a single workflow, the proposed system simplifies media verification and strengthens digital security. It helps users make faster and more informed decisions in applications such as online authentication, surveillance, social media monitoring, and cybercrime prevention.

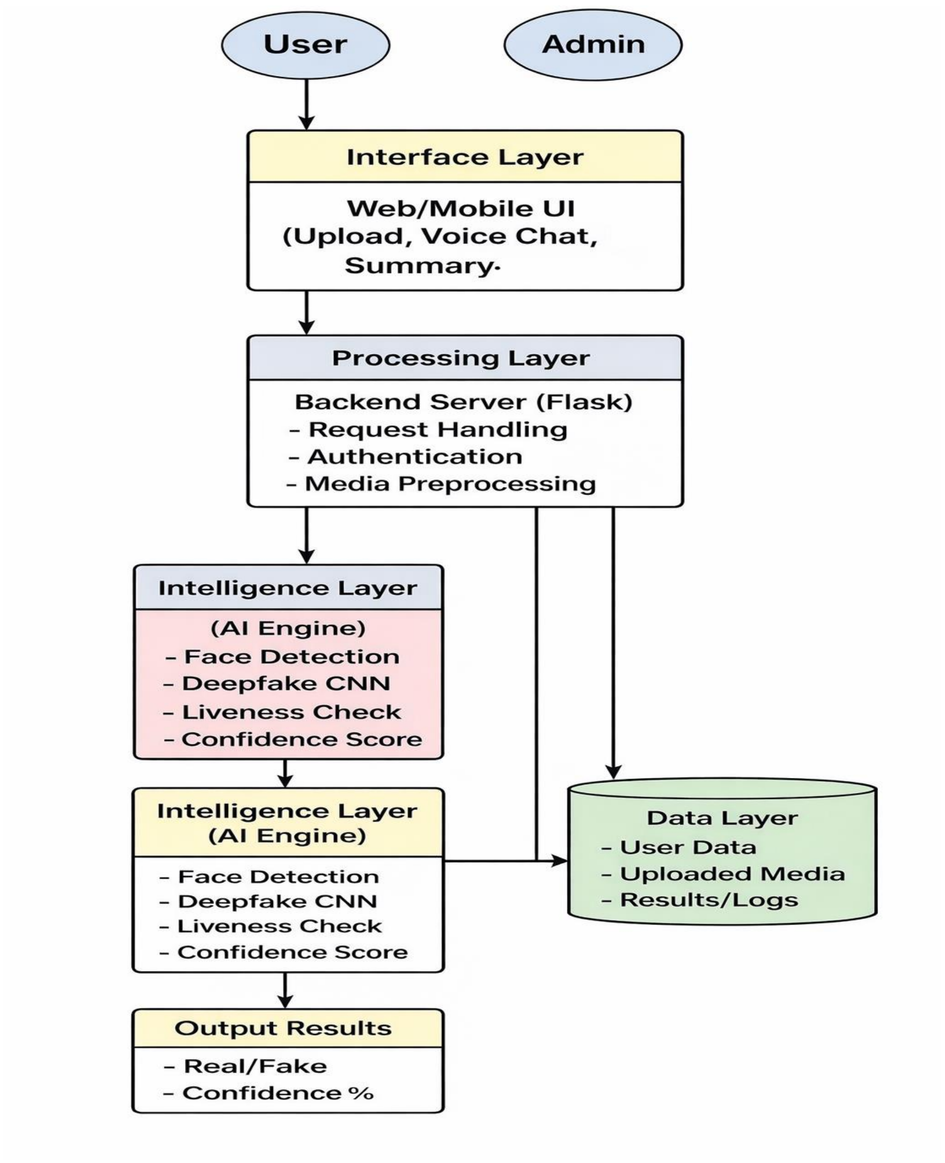


Figure 3.1.3.1 Block Diagram of Proposed System

CHAPTER 4

FEASIBILITY STUDY

4.1 Feasibility Study

The proposed **AI-Powered Deepfake Detection with Liveness Detection system** demonstrates high feasibility in terms of technical, operational, economic, and time aspects. The system integrates deep learning and computer vision techniques with a scalable web-based architecture to provide accurate and reliable detection of manipulated images and videos. It addresses real-world security challenges such as identity spoofing and media manipulation, making it suitable for applications in authentication, surveillance, and digital content verification.

4.1.1 Technical Feasibility

The proposed system is technically feasible as it is developed using well-established and widely supported technologies. The frontend is implemented using HTML and CSS to provide a simple and responsive user interface. The backend is built using Python and Flask, which efficiently handles user requests, file uploads, and communication with AI models. The intelligence layer uses TensorFlow and OpenCV to perform face detection, deepfake classification, and liveness verification using proven deep learning techniques.

All required tools and libraries are open-source and compatible with standard computing systems. The system does not require specialized hardware and can operate on machines with basic processing power. The technical skills required—such as Python programming, deep learning model integration, and basic web development—are well within the scope of undergraduate-level projects. Hence, the system can be developed and deployed without major technical challenges.

4.1.2 Economic Feasibility

The project is economically feasible as it relies entirely on open-source software and free development tools. Technologies such as Python, Flask, TensorFlow, OpenCV, and standard web technologies eliminate the need for costly licensed software. No specialized or high-end hardware is required; a standard computer with internet access is sufficient for development and testing.

Storage and deployment costs are minimal, as lightweight hosting or local servers can be used for academic purposes. Since the system automates the detection of deepfake and spoofing attacks, it reduces the need for manual verification, thereby saving time and operational costs. Overall, the system offers high value with minimal financial investment, making it cost-effective and suitable for academic and small-scale organizational use.

4.1.3 Operational Feasibility

The system is operationally feasible and user-friendly. Users can easily upload images or videos through the web interface and receive detection results in real time. The system provides clear output indicating whether the media is real or fake along with a confidence score, making it easy to interpret even for non-technical users.

Liveness detection enhances system reliability by preventing spoofing attacks using photos or recorded videos. Detection results and uploaded media can be stored securely, allowing users to review previous analyses. The system is suitable for use in educational institutions, security applications, and digital platforms where media authenticity is critical. Its simple workflow and automated processing make it practical and effective in real-world scenarios.

4.1.4 Time Feasibility

The project is feasible within the given academic timeline as the development process is divided into manageable phases such as requirement analysis, system design, model integration, frontend and backend development, testing, and deployment. Each phase can be completed sequentially without excessive complexity.

Pre-trained models and existing datasets reduce the time required for model training and testing. Modern development frameworks and libraries speed up implementation and debugging. With proper planning and teamwork, the complete system can be developed, tested, and documented within the allotted project duration. Therefore, the project is highly feasible in terms of time constraints.

4.2 Methodology

4.2.1 Introduction

Methodology refers to the structured approach and overall plan used to design, develop, and implement a system after completing system analysis. It provides a systematic framework that ensures each stage of development follows a logical sequence and contributes effectively toward achieving the project objectives. A well-defined methodology helps maintain clarity, consistency, and efficiency throughout the project, ensuring that the system is implemented in a reliable, scalable, and organized manner.

The purpose of adopting a methodology in this project is to clearly define the technical approach, specify the development model followed, and describe the workflow involved in building the proposed AI-powered system. It outlines the processes involved in media input handling, facial preprocessing, deepfake detection, liveness verification, result generation, and storage. By establishing clear steps from image and video acquisition to deep learning inference and output visualization, the methodology ensures a structured and repeatable development process.

In this project, the methodology explains how deepfake detection and liveness verification are integrated into a single intelligent system. It describes the tools, algorithms, and workflow used—from user interface design and backend processing to AI model execution and secure data handling. This methodology acts as a blueprint that transforms project requirements into a fully functional and deployable AI-based deepfake detection system.

4.2.2 Project Approach

The project follows a structured and systematic development approach to ensure successful implementation of the proposed AI-Powered Deepfake Detection with Liveness Detection system. The development process is divided into the following phases:

- **Requirement Gathering:** This phase identifies both functional and non-functional requirements such as image and video upload support, accurate deepfake detection, liveness verification, real-time result generation, system security, and user-friendly interaction. Existing deepfake detection and anti-spoofing systems are analyzed to understand their limitations and identify improvement opportunities.

- **System Design:** In this phase, the overall system architecture is designed, including the frontend interface, backend services, AI processing modules, and data storage components. Block diagrams, workflow diagrams, and data flow models are created to illustrate how media files are processed through face detection, deepfake classification, liveness analysis, and result display.

- **Implementation:** The frontend is developed using HTML and CSS to provide a simple and responsive user interface. Backend services are implemented using Python and Flask to handle file uploads, preprocessing, and communication with the AI models. Deep learning models built with TensorFlow and OpenCV are integrated for deepfake detection and liveness verification. Media files and detection results are securely stored for future reference.

- **Testing:** The system is tested for functionality, accuracy, usability, and performance. Various image and video samples are used to evaluate deepfake detection accuracy and liveness verification reliability. Errors are identified and corrected, and model performance is validated under different conditions such as lighting variations and video quality.

- **Evaluation:** The final system is evaluated based on detection accuracy, response time, reliability, and user satisfaction. The results are compared with existing deepfake detection approaches to validate improvements in security, efficiency, and real-time performance.

- This well-defined project approach ensures that development is carried out in an organized, efficient, and technically sound manner, resulting in a stable and effective deepfake detection system suitable for real-world security applications.

4.2.3 Development Model

For the systematic development of this project, the **Incremental Development Model** has been adopted. The Incremental Model is a structured approach in which the system is developed and delivered in small, functional modules (increments), with each increment adding new features to the existing system. This model is well suited for the proposed project because it involves multiple interrelated components such as media upload, deepfake detection, liveness verification, result visualization, and data storage.

The Incremental Model begins with **requirement gathering**, where the limitations of existing deepfake detection systems—such as lack of liveness verification, poor generalization, and limited real-time capability—are clearly identified. This is followed by **system design**, where the overall architecture, block diagrams, and data flow between frontend, backend, and AI modules are defined.

In the **implementation phase**, the system is developed increment by increment. The first increment focuses on media upload and preprocessing. The second increment integrates face detection and deepfake classification using deep learning models. The third increment introduces liveness detection techniques such as eye blink and head movement analysis. Subsequent increments handle result generation, confidence scoring, and storage of detection reports.

After each increment, **testing and validation** are performed to ensure correct functionality and accuracy before moving to the next stage. The final increment integrates all modules into a complete system, followed by overall evaluation.

The structured and flexible nature of the Incremental Development Model reduces development risk, allows early detection of errors, and supports future enhancements such as audio deepfake detection or mobile deployment. Hence, this model is highly suitable for developing a reliable and scalable AI-powered deepfake detection system.

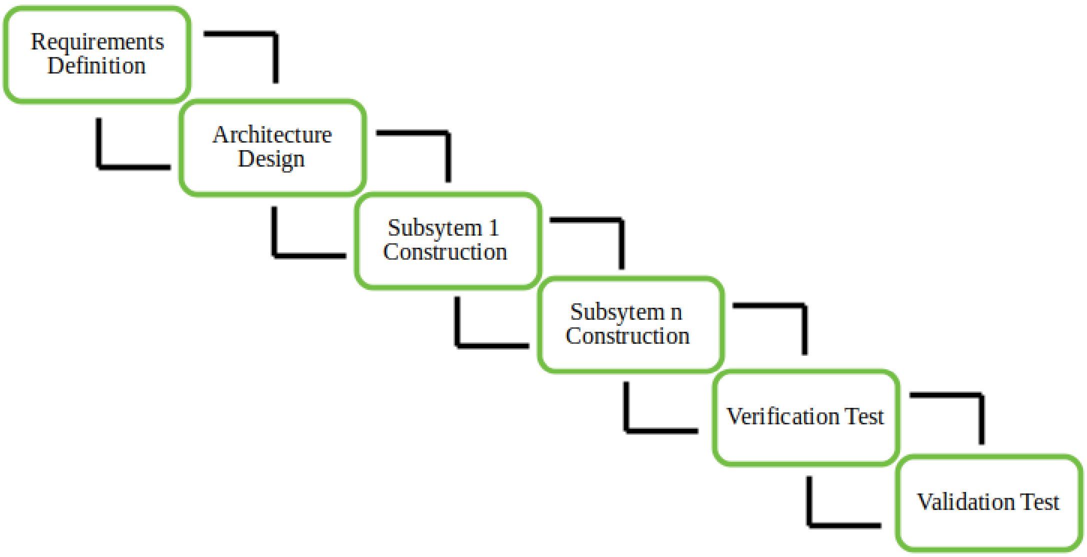


Figure 4.2.3.1 Incremental Model

4.2.4 System Workflow

- **Start & initialize:** The system initializes required services, loads the AI models, and prepares the backend for processing user requests. services.
- **Media Upload:** The user uploads an image or video through the web-based interface.
- **Preprocessing:** The uploaded media is validated, resized, and prepared for analysis. For videos, frames are extracted at regular intervals.
- **Face Detection:** Facial regions are detected and cropped from images or video frames using computer vision techniques.
- **Deepfake Detection Execution:** The cropped facial data is passed to the deep learning model to detect manipulation and classify the content as real or fake.
- **Liveness Detection:** The system verifies real-time human presence by analyzing eye blinking patterns, head movements, and facial motion across frames.
- **Decision Making**
  Results from deepfake detection and liveness verification are combined to determine the final authenticity status.
- **Result Generation**
  The system generates the final output indicating **Real or Fake**, along with a confidence score.
- **Display Output**
  The detection result and confidence percentage are displayed to the user through the web interface.
- **Secure Storage**
  Uploaded media files and detection results are stored securely for future reference and analysis.
- **User Interaction Loop**
  The user can upload new media files or review previous detection results.
- **End**
  The session ends when the user exits the application.

4.2.5 Techniques and Algorithms

To develop the proposed **AI-Powered Deepfake Detection with Liveness Detection system**, several deep learning, computer vision, and biometric analysis techniques are employed. These techniques ensure accurate identification of manipulated media and reliable verification of real-time human presence. The following techniques and algorithms form the core of the system's functionality and performance

1. **Core model components & algorithms:**
    - **Convolutional Neural Networks (CNNs):** CNNs are used as the primary deep learning model for detecting deepfake content in images and video frames. These models learn spatial features such as texture inconsistencies, unnatural facial patterns, and visual artifacts introduced during manipulation.
    - **Transfer Learning (MobileNet/CNN Backbone):** Pre-trained CNN models are fine-tuned on deepfake datasets to improve accuracy while reducing training time and computational cost.
    - **Binary Classification Algorithm:** The system classifies input media into Real or Fake based on the probability score generated by the model.
2. **Liveness Detection Techniques:**
    - **Eye Blink Detection (EAR – Eye Aspect Ratio):** Facial landmark detection is used to analyze natural eye blinking patterns, helping identify whether the subject is a live human.
    - **Head Movement Analysis:** Tracks head motion across video frames to ensure real-time interaction and prevent spoofing through static images or recorded videos.
    - **Facial Landmark Tracking:** Detects irregular or unnatural facial movements that indicate deepfake or replay attacks.
3. **Preprocessing and Feature Extraction Techniques:**
    - **Face Detection and Cropping:** OpenCV-based face detection techniques are used to locate and crop facial regions for focused analysis.

- **Frame Extraction (for Videos):** Videos are broken into individual frames at fixed intervals to capture temporal inconsistencies.
- **Image Normalization and Resizing:** Facial images are resized (e.g., 224×224) and normalized to ensure consistent model input.

4. **Optimization and Performance Techniques.**
   - **Frame-wise Prediction Aggregation:** Predictions from multiple frames are combined to generate a final video-level decision.
   - **Lightweight Model Architecture:** MobileNet-based CNN ensures fast inference and real-time detection even on systems with limited hardware.
   - **Confidence Score Calculation:** Probability scores are generated to indicate detection certainty.
5. **Stabilization and Reliability Techniques.**
   - **Threshold-Based Decision Making:**
     Final classification is based on confidence thresholds to reduce false positives.
   - **Multi-Cue Verification:**
     Combines deepfake detection results with liveness detection outcomes for higher reliability.
   - **Secure File Handling:**
     Uploaded media files and detection results are safely managed by the backend.
6. **Data processing & workflow setup:**
   - **Media Upload and Validation:**
     Supports image and video file formats with validation checks.
   - **Sequential Processing Pipeline:**
     Upload → Preprocessing → Face Detection → Deepfake Detection → Liveness Verification → Result Generation.
   - **Real-Time Web Delivery:**
     Results are displayed instantly via the web interface.

7. **Evaluation and Validation Technique:**

    - **Accuracy Measurement:**
      Evaluates how correctly the system identifies real and fake inputs.
    - **Confusion Matrix Analysis:**
      Used to analyze false positives and false negatives.
    - **Performance Testing:**
      Measures system speed and responsiveness under different media sizes and conditions.

8. **Tools, frameworks & deployment considerations:**

    - **Python + Flask:** Backend processing and AI model integration
    - **TensorFlow & OpenCV:** Deep learning and image processing
    - **NumPy:** Numerical computation
    - **HTML & CSS:** Frontend interface
    - **Standard Hardware:** Runs efficiently on systems with integrated GPU

4.2.6 Tools and Technologies

The development of the proposed **AI-Powered Deepfake Detection with Liveness Detection system** requires a combination of programming languages, web technologies, deep learning frameworks, computer vision libraries, and system tools. The following tools and technologies are used for designing, implementing, testing, and deploying the system.

1. **Programming Languages**
   - **Python:** Python is the primary programming language used for implementing deep learning models, image and video preprocessing, face detection, deepfake classification, and liveness detection. It is also used for backend logic and AI inference.
   - **JavaScript:** JavaScript is used to enhance frontend interaction and support communication between the user interface and backend services.
2. **Web & Backend Frameworks**
   - **HTML & CSS:** Used to design a simple, responsive, and user-friendly web interface for uploading images/videos and displaying detection results.
   - **Flask (Python):** Acts as the backend framework that handles file uploads, preprocessing requests, model inference, and result generation. Flask enables lightweight and efficient communication between the frontend and AI models.
3. **AI and Computer Vision Libraries**
   - **TensorFlow / Keras:**
     Used for building and deploying deep learning models such as CNNs (e.g., MobileNet) for deepfake detection.
   - **OpenCV:**
     Used for image and video processing tasks such as face detection, frame extraction, resizing, and normalization.

- **NumPy:**
  Supports numerical operations, array manipulation, and data handling during preprocessing and model inference.
- **Dlib / MediaPipe (Optional):**
  Used for facial landmark detection to support eye blink analysis and head movement tracking for liveness detection.

4. **Development Tools & IDEs**
   - **Visual Studio Code:** Used as the primary Integrated Development Environment (IDE) for writing, debugging, and managing Python and web application code.
   - **Jupyter Notebook:** Used for experimentation, model testing, data analysis, and visualization of deepfake detection results.
5. **Storage and Data Management**
   - **Local Storage / File System:**
     Used to temporarily store uploaded images, videos, and extracted frames during processing.
   - **Structured Dataset Storage:**
     Public datasets such as FaceForensics++ and DFDC are used for training and evaluation purposes.
6. **Evaluation and User Experience Tools**
   - **Scikit-Learn:** Used to calculate evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
   - **Matplotlib:** Used for visualizing model performance, accuracy graphs, and detection statistics.
7. **Version Control and Documentation**
   - **Git & GitHub:** Used for version control, source code management, collaboration, and backup.

- **Markdown:** Used for maintaining documentation, project notes, and development logs.

8. **Hardware Requirements**
    - **Standard Computer / Laptop with Internet:** A system with Intel i5 processor, 8GB RAM, integrated GPU, and stable internet connection is sufficient for development and testing.
    - **Web Browser:** Google Chrome is used for accessing and testing the web application.

CHAPTER 5

PROJECT MANAGEMENT AND COST ESTIMATION

5.1 Project Management

Project management plays a crucial role in ensuring the successful development and timely completion of the AI-Powered Deepfake Detection: Liveness Detection system. The project is managed using a structured and systematic approach, where all activities such as requirement analysis, system design, model development, implementation, testing, and documentation are carefully planned and executed. The Incremental Development Model is followed to allow step-by-step development and continuous testing of individual modules such as media upload, deepfake detection, liveness verification, and result generation. Clear task allocation, regular progress monitoring, and proper coordination among team members help in minimizing risks and ensuring smooth project execution within the academic timeline.

The project schedule is organized into sequential phases, starting from requirement gathering and design, followed by model integration, frontend and backend development, testing, and final evaluation. Each phase is assigned a specific duration to ensure steady progress and timely completion. This planned workflow enables early identification of issues and supports effective debugging and optimization. Proper documentation is maintained throughout the project lifecycle to ensure clarity, maintainability, and ease of future enhancements.

5.2 Cost Estimation

Cost estimation for the project indicates that the overall expense is minimal, making it highly suitable for academic implementation. The system is developed using open-source technologies such as Python, Flask, TensorFlow, OpenCV, HTML, and CSS, which eliminates the need for expensive licensed software. Existing hardware resources such as a standard computer with an Intel i5 processor, 8 GB RAM, integrated GPU, and stable internet connection are sufficient for development and testing. No additional specialized hardware is required.

CHAPTER 6

SYSTEM REQUIREMENTS

6.1 Introduction

A Requirement Specification is a detailed and structured document that clearly defines what a system is expected to do, how it should behave, and the conditions under which it must operate. It translates user needs, project goals, and system expectations into well-defined functional and non-functional requirements that guide the entire development process. By specifying system inputs, outputs, processing steps, constraints, and performance expectations, the requirement specification acts as a foundational blueprint for developers, designers, and testers.

Requirement specification is essential to ensure that all stakeholders have a shared understanding of the system's purpose and capabilities. It reduces ambiguity, minimizes development errors, and provides measurable criteria for validation and testing. In AI-based security systems such as deepfake detection, a clear requirement specification is critical to ensure accuracy, reliability, and real-time performance throughout the project lifecycle.

6.2 Functional Requirements

Functional requirements define the essential operations, behaviors, and features that the proposed system must perform to achieve its objectives.

1. **Media Upload and Preprocessing**
    - The system must allow users to upload image and video files for analysis.
    - The system must validate file formats and sizes before processing.
    - The system must preprocess uploaded media by resizing, normalizing, and extracting frames (for videos).
2. **Face Detection Module**
    - The system must detect human faces from uploaded images or video frames.
    - The system must crop and isolate facial regions for focused analysis.
3. **Deepfake Detection Module**

- The system must analyze facial features to detect manipulated or synthetic media.
- The system must classify input media as **Real** or **Fake**.
- The system must generate a deepfake probability score.

4. Liveness Detection Module

- The system must verify real-time human presence.
- The system must analyze eye blinking patterns, head movements, and facial behavior.
- The system must prevent spoofing attacks using static images or recorded videos.

5. Result Generation

- The system must display the final classification (Real/Fake).
- The system must show confidence percentage and detection details.

6. User Interface

- The system must provide a simple and user-friendly interface.
- The system must allow users to upload media and view results easily.

7. Data Storage and Retrieval

- The system must store uploaded media and detection results securely.
- The system must allow users to review previous detection results.

6.3 Non-Functional Requirements

Non-functional requirements describe the quality attributes, constraints, and performance expectations of the system.

1. **Performance Requirements**
   - The system should process images and videos in real time or near real time.
   - The system should handle large media files without significant delay.

2. **Accuracy Requirements**
   - The system should accurately distinguish between real and fake media.
   - False positives and false negatives should be minimized.

3. **Reliability Requirements**

- The system should function consistently without crashes.
- Stored data must remain accessible without corruption.

4. **Scalability Requirements**
   - The system should support multiple users simultaneously.
   - The system should be extendable to support audio deepfake detection in the future.
5. **Usability Requirements**
   - The interface must be intuitive for both technical and non-technical users.
   - Results must be easy to understand and visually clear.
6. **Maintainability Requirements**
   - The codebase should be modular for easy updates.
   - Proper documentation should be maintained.
7. **Portability Requirements:**
   - The system should run on different browsers and operating systems.
   - The system should be accessible through a web-based platform.

6.4 Hardware Requirements

The proposed system requires standard computing hardware capable of handling deep learning inference and media processing.

1. **Development Environment**
   - **Laptop / Desktop Computer**
     - Minimum Processor: Intel i3 / AMD equivalent
     - Recommended Processor: Intel i5 or above
   - **RAM**
     - Minimum 8 GB.
     - Recommended 16 GB for faster model execution.
   - **Storage:** At least 1-2 GB free disk space.
2. **User Environment**
   - Any device capable of accessing a modern web browser.
   - Stable internet connection for web-based access.

- Hardware requirements fall within standard academic and professional device capabilities.

6.5 Software Requirements

Software requirements define the platforms, tools, and libraries required to develop and execute the system.

1. **Operating System**
    - Windows 10 / Windows 11, MacOS
2. **Programming Language**
    - Python 3.10 or above.
    - JavaScript
3. **Backend & AI Frameworks**
    - Flask – Backend framework for request handling and AI inference
    - TensorFlow / Keras – Deep learning model implementation
4. **Computer Vision Libraries**
    - OpenCV – Image and video processing
    - Dlib / MediaPipe – Facial landmark detection (for liveness analysis)

1. **Frontend Technologies**
    - HTML, CSS – User interface design
2. **Development Tools**
    - Visual Studio Code – Code development and debugging
    - Jupyter Notebook – Model testing and experimentation

7. **Web Browser**
    - Google Chrome (recommended)

CHAPTER 7

SYSTEM DESIGN

7.1 Introduction

System design is the process of defining the architecture, components, modules, interfaces, and data flow of a system before actual implementation. It explains how different system components interact with each other to achieve the desired functionality. A well-structured system design acts as a blueprint for developers, helping to reduce complexity, avoid errors, and ensure that the system meets all functional and non-functional requirements.

7.2 Data Flow Diagram

A Data Flow Diagram (DFD), is the highest-level representation of a system that shows the entire system as a single process. It illustrates how the system interacts with external entities such as users, administrators, or other systems. DFD focuses only on major data flows entering and leaving the system, without showing internal processes or details. Its purpose is to provide a simple and clear overview of the system boundaries, external communication, and primary inputs and outputs.

The Data Flow Diagram (DFD) represents as a single high-level process interacting with external entities such as the user and the database. It illustrates how images or videos provided by the user are processed by the system to determine authenticity and generate detection results. This diagram defines the system boundaries and provides an overall view of data flow without revealing internal processing details.

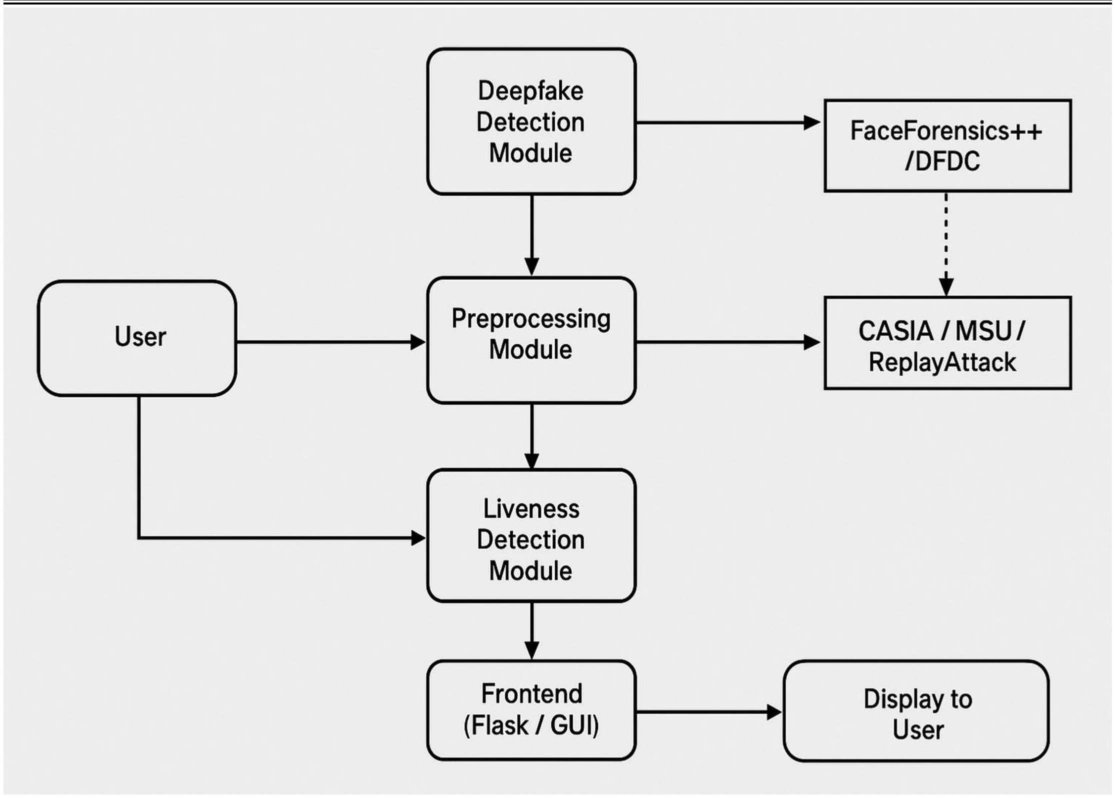


Figure 7.2.1 Data Flow Diagram

Main Elements

1. **User**: The user uploads image or video files for verification and receives the detection results, including real/fake classification and confidence score.
2. **Deepfake Detection Application**: This represents the core system that processes the uploaded media, performs deepfake detection, verifies liveness, and generates the final authenticity result.
3. **Data Flows:**

   - **Upload Image / Video →** Input data sent from the user to the deepfake detection application for analysis.
   - **Detection Result / Confidence Score →** Output sent from the application back to the user after processing.
   - **Store / Retrieve Detection Data →** Interaction between the application and the database for saving and fetching detection results.

- **Processed Media / Detection History →**
  Stored information that allows result review and system logging.

7.3 Flow Chart

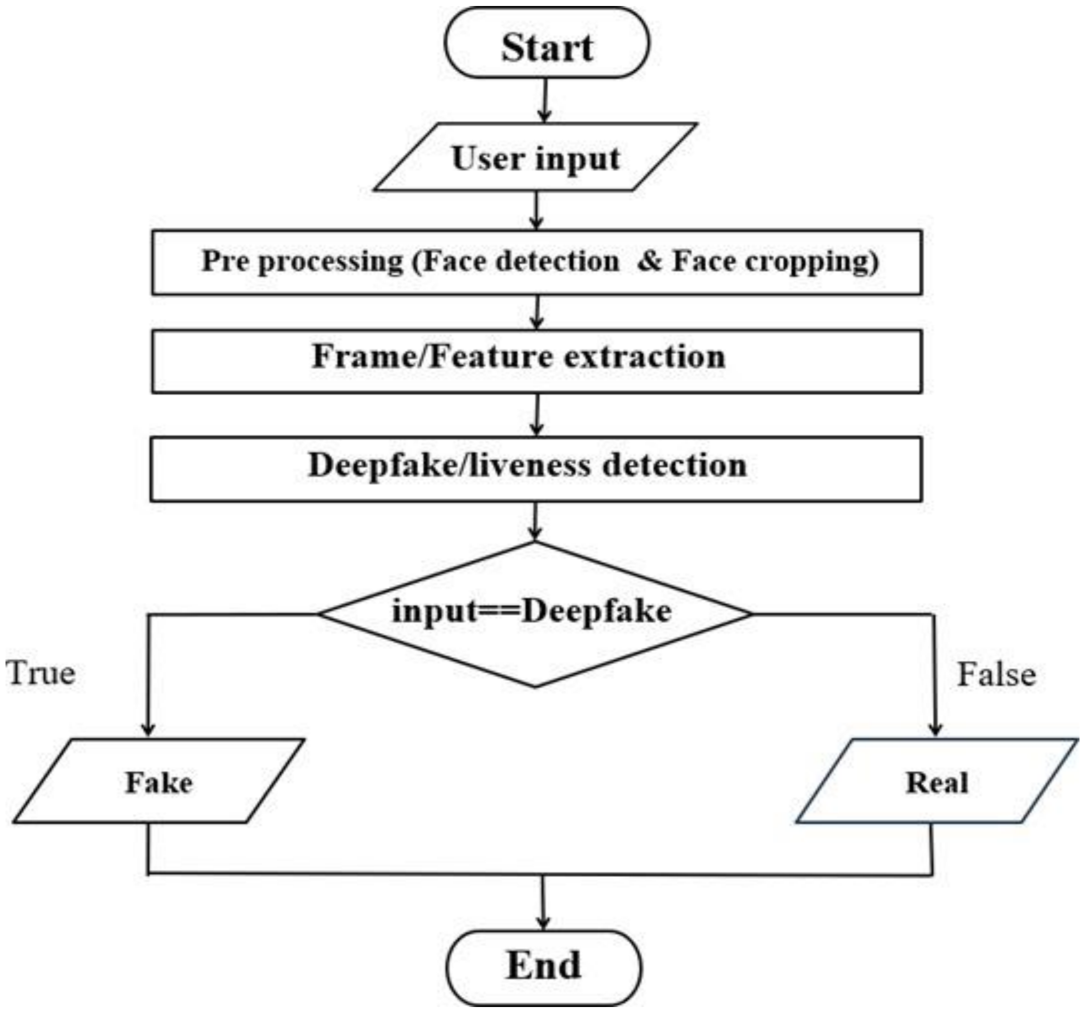

Start
User input
Pre processing (Face detection & Face cropping)
Frame/Feature extraction
Deepfake/liveness detection
input==Deepfake
True
False
Fake
Real
End

Fig 7.3.1 Flow Chart of Deepfake and liveness

1. Start:

This block represents the initialization of the system. At this stage, the deepfake and liveness detection framework is activated, and all required modules such as input handling, preprocessing, and detection models are prepared to operate.

2. User Input

In this step, the user provides input data to the system.

The input can be:

A static image (e.g., face photograph)

A video clip (e.g., selfie video or recorded footage)

A live camera feed

This input acts as the raw data on which deepfake or liveness analysis will be performed.

3. Pre-processing (Face Detection & Face Cropping)

Pre-processing is a crucial stage to ensure accurate analysis.

Face Detection: The system identifies the presence and location of a human face in the input using face detection algorithms.

Face Cropping: Once detected, the face region is cropped from the background to remove unnecessary information such as surroundings, objects, or other people.

Purpose:

- Reduces noise
- Focuses analysis only on facial regions
- Improves detection accuracy and efficiency

4. Frame / Feature Extraction

In this stage, meaningful data is extracted from the processed face.

For images: Facial features such as texture, edges, skin patterns, and landmarks are extracted.

For videos: The video is divided into multiple frames, and temporal features such as eye blinking, head movement, and facial expressions are analyzed.

Purpose:

- Converts raw face data into measurable features
- Helps the model understand visual and motion-based cues
- Enables detection of manipulation artifacts or spoofing attempts

5. Deepfake / Liveness Detection

This is the core processing stage of the system.

A trained deep learning or machine learning model analyzes the extracted frames or features.

The model checks for:

Artificial face generation artifacts

Inconsistent lighting or textures

Lack of natural facial movements (for liveness)

Replay, mask, or photo attacks

Outcome: The model predicts whether the input is genuine (live) or manipulated (deepfake/fake).

6. Decision Node: Input == Deepfake

This is a conditional decision block.

If True →The system classifies the input as Fake / Deepfake

(e.g., AI-generated video, spoofed image, replay attack)

If False →The system classifies the input as Real / Live

(e.g., genuine human face with natural behavior)

This decision is based on confidence scores or probability thresholds produced by the detection model.

7. End

The process terminates after classification.

The final result (Real or Fake) is returned to:

The user

A security system

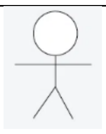
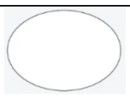
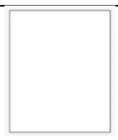
An authentication module

This output can be used for identity verification, fraud prevention, or access control systems.

7.4 Use Case Diagram

A Use Case is a functional description of how a user or external actor interacts with a system to achieve a specific goal. It outlines the sequence of actions, system responses, and interactions that take place during a particular task. Use cases help define system requirements from the user's perspective, ensuring that the system supports all necessary functionalities needed by its users.

Table 7.4.1 Symbols Representation for Use Case Diagram

Symbol	Name	Description
	**Actor**	Represents an external user interacting with the system
	**Use Case**	Represents a functional requirement or feature provided by the system.
	Association	Connects actors to use cases indicating interaction.
	**System Boundary**	Defines the scope of the system functions.

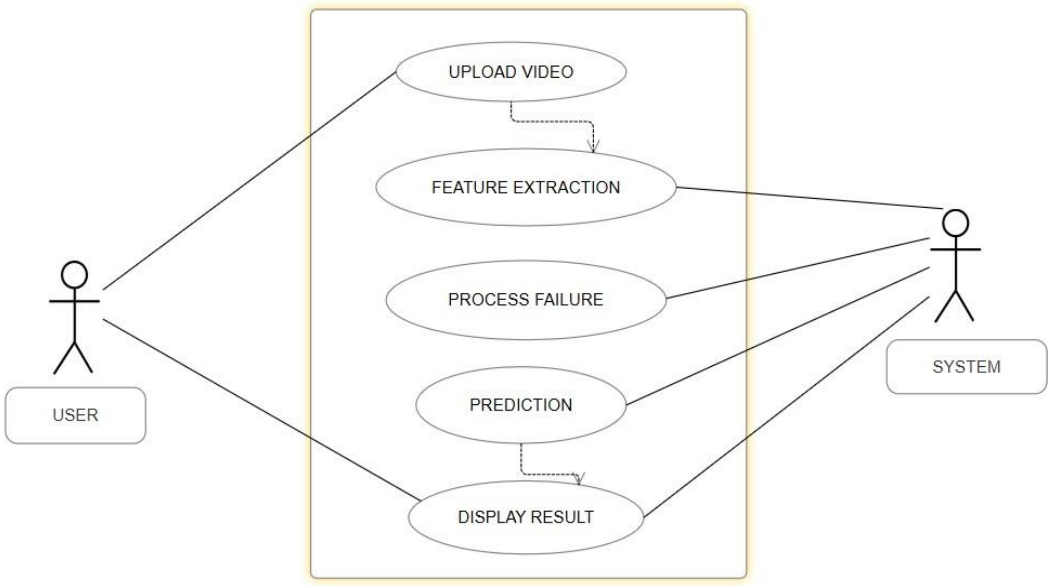


Figure 7.4.1 Use Case Diagram

The Use Case Diagram illustrates the interaction between the user and the AI-Powered Deepfake Detection: Liveness Detection system, highlighting the major functionalities involved in video analysis and result generation. The primary actor in the system is the User, who initiates the process by uploading a video for authenticity verification. The System acts as the backend processing unit that handles all internal operations such as feature extraction, prediction, and result generation.

The use case begins when the user uploads a video to the system through the user interface. Once the video is successfully uploaded, the system automatically performs feature extraction, where important visual and temporal features such as facial landmarks, eye movements, texture patterns, and frame-level inconsistencies are extracted from the video. These features are essential for identifying manipulated or synthetic content. If any issue occurs during feature extraction or processing—such as unsupported file formats or corrupted media—the system may trigger a process failure, which acts as an optional outcome handled internally by the system.

After successful feature extraction, the system proceeds to the prediction phase, where deep learning models analyze the extracted features to determine whether the video is real or fake. This stage performs deepfake detection and liveness verification to ensure real-time human presence. Once the prediction is completed, the system executes the display result use case, presenting the final outcome to the user. The result includes a clear classification such as “Real” or “Fake,” often accompanied by a confidence score to improve transparency and user understanding.

7.5 Sequence Diagram

The sequence diagram illustrates the step-by-step interaction between the User, Web Interface, Preprocessing module, and Detection Model in the AI-Powered Deepfake Detection: Liveness Detection system. It shows how a file uploaded by the user is processed and analyzed to generate the final authenticity result.

The interaction begins when the user initiates the process by uploading a file, which may be an image or a video, through the web interface. In response, the web interface displays the available upload options, allowing the user to select the desired file type. Once the user selects and confirms the file, the web interface forwards the uploaded file to the preprocessing component of the backend system.

The preprocessing module prepares the uploaded file for analysis by performing operations such as face detection, frame extraction (for videos), resizing, normalization, and feature extraction. After successful preprocessing, the processed data is sent to the deepfake detection model. The detection model evaluates the extracted features using deep learning techniques and generates a deepfake probability score or classification result indicating whether the input is real or fake.

The detection result is then sent back to the preprocessing module, which forwards the outcome to the web interface. Finally, the web interface displays the result to the user in a clear and understandable format, such as "Real" or "Fake," along with a confidence score if available. This sequential flow ensures efficient communication between system components and provides the user with accurate and timely deepfake detection results.

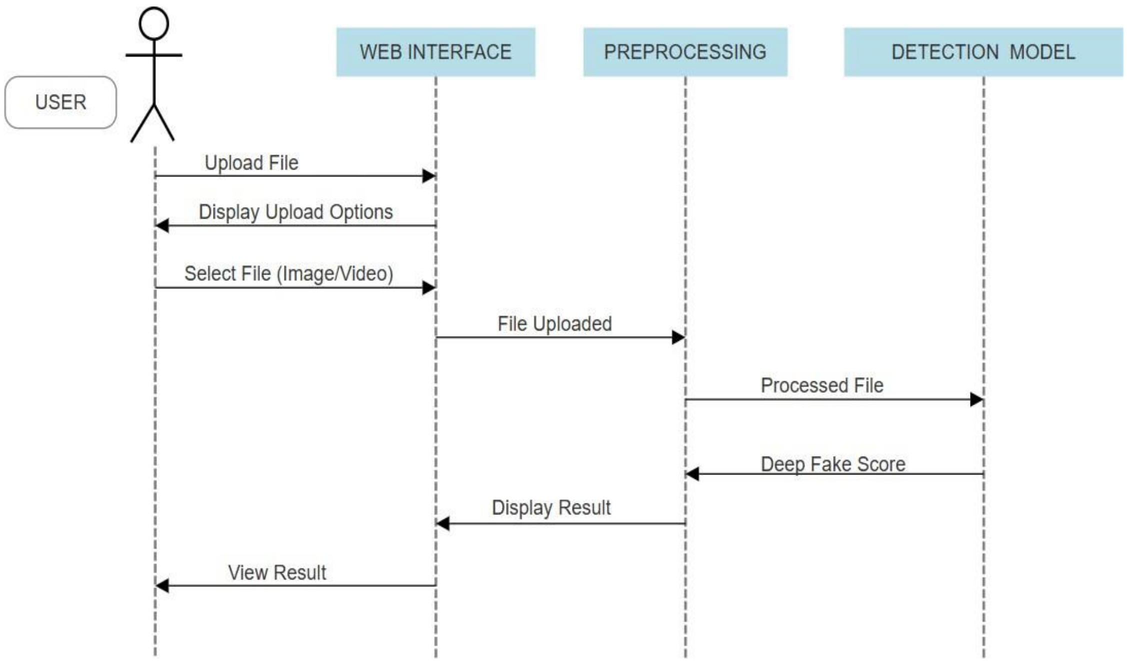


Figure 7.5.1 Sequence Diagram

CHAPTER 8

IMPLEMENTATION

8.1 Introduction

Implementation is the phase in which the system design and models are transformed into a fully functional software application. It involves coding the individual modules, integrating frontend and backend components, implementing deep learning models, and ensuring smooth communication between different system layers. This phase is crucial as it converts the theoretical architecture into a working system that users can interact with in real time.

8.2 Modules and Their Functionality

The proposed system is divided into several core modules, each responsible for a specific task in the deepfake and liveness detection workflow. These modules work together to process input media, analyze authenticity, and present results to the user.

1. **User Interface Module**: Provides a simple and intuitive web-based interface that allows users to upload images or videos and view detection results along with confidence scores.
2. **Media Upload Module**: Handles user-uploaded image and video files, validates file formats, and forwards the media to the backend for processing.
3. **Preprocessing Module**: PPerforms operations such as frame extraction (for videos), resizing, normalization, and face detection to prepare the input data for analysis.
4. **Deepfake Detection Module**: Uses Convolutional Neural Networks (CNNs) to analyze facial features, texture inconsistencies, and visual artifacts to detect manipulated media.
5. **Liveness Detection Module:** Verifies real-time human presence by analyzing eye blinking patterns, head movements, and facial motion to prevent spoofing attacks.
6. **Decision and Scoring Module**: Combines deepfake detection and liveness verification results to generate a final classification (Real or Fake)

along with a confidence score.

7. **Result Display Module**: Sends the detection result and confidence score to the frontend for clear visualization to the user.

8.3 Tools & Technologies Used

The following tools and technologies are used during the implementation phase to develop a robust and scalable system.

1. Frontend Technologies

- **HTML & CSS:** Used for creating a responsive and user-friendly interface.
- **JavaScript:** Enables interaction between the frontend and backend services.

2. Backend Technologies

- **Flask (Python):** Handles file uploads, preprocessing requests, and communication with AI models.

3. Programming Languages

- **Python:** Used for deep learning model implementation, image/video processing, and backend logic.
- **JavaScript:** Used for frontend interactions and API communication.

4. AI and Computer Vision Libraries

- **TensorFlow / Keras:** For building and deploying CNN-based deepfake detection models.
- **OpenCV:** For image and video preprocessing, face detection, and frame extraction.
- **NumPy:** For numerical computation and data handling.

5. Development Tools

- **Visual Studio Code:** Used for coding, debugging, and project management.
- **Jupyter Notebook:** Used for model experimentation and testing.

8.4 Techniques & Algorithms Used

The system uses several deep learning and computer vision techniques to detect deepfakes and verify liveness accurately:

- **Convolutional Neural Networks (CNNs):**
  CNNs are used to analyze spatial features in facial images and video frames to identify manipulation artifacts.
- **Transfer Learning:**
  Pre-trained CNN models such as MobileNet are fine-tuned to improve accuracy while reducing training time.
- **Frame-Based Video Analysis:**
  Videos are split into frames to detect temporal inconsistencies and unnatural facial movements.
- **Eye Blink Detection:**
  Facial landmark-based analysis is used to detect natural eye blinking patterns as part of liveness verification.
- **Head Movement Analysis:**
  Tracks facial motion across frames to ensure real-time human presence.
- **Threshold-Based Decision Algorithm:**
  Combines model predictions and liveness results to generate a final decision with confidence score.
- **Secure Data Handling:**
  Ensures safe processing and temporary storage of uploaded media during analysis.

CHAPTER 9

TESTING

9.1 Introduction

Testing is a critical phase in the software development lifecycle that involves executing the system to verify whether it functions correctly, meets the specified requirements, and operates without major errors or defects. Through systematic testing procedures, the developed software is evaluated to ensure that each module performs its intended function, integrated components work together as expected, and the overall system behaves reliably under various operating conditions.

Testing is especially important for AI-based systems such as deepfake detection, as it ensures the accuracy, stability, and robustness of the detection models and liveness verification mechanisms. It helps identify bugs, performance issues, incorrect classifications, or inconsistencies in model predictions at an early stage. Proper testing ensures that deepfake detection results are reliable, liveness checks function correctly, and the system produces consistent outputs across repeated executions.

9.2 Testing Strategies

9.2.1 Unit Testing

Unit Testing focuses on verifying the smallest functional components of the system, such as individual functions, classes, or modules, in isolation. Each unit is tested independently to ensure it performs the expected operation and handles inputs, outputs, and edge cases correctly without errors.

In this project, unit testing is applied to modules such as media upload validation, frame extraction, face detection, deepfake prediction logic, liveness detection functions (eye blink and head movement), and result formatting. Unit testing ensures that each core function works correctly before it is integrated with other modules. Early detection of errors through unit testing reduces debugging effort and improves overall system reliability.

9.2.2 Module Testing

Module Testing verifies the functionality of a complete module after all its internal units have been tested and integrated. Unlike unit testing, which focuses on individual functions, module testing evaluates whether all components within a module work together correctly as a cohesive unit.

For the proposed system, module testing is performed on modules such as the preprocessing module, deepfake detection module, liveness detection module, and result generation module. This testing ensures that internal data flow, parameter passing, and interactions between functions operate correctly. Module testing helps identify logical errors, incorrect data handling, or mismatches between internal components before full system integration.

9.2.3 Integration Testing

Integration Testing focuses on verifying the interaction between different modules once they are combined into larger subsystems. It ensures that data produced by one module is correctly consumed by another and that communication between modules is seamless.

In this project, integration testing verifies interactions between the frontend and backend, preprocessing and detection modules, deepfake detection and liveness verification components, and result generation and display modules. This testing helps identify issues such as incorrect data formats, broken API communication, timing problems, or mismatched assumptions between modules. Integration testing ensures smooth end-to-end execution of the deepfake detection workflow.

9.2.4 System Testing

System Testing evaluates the entire deepfake detection system as a fully integrated unit. After unit, module, and integration testing are completed, system testing verifies that the system meets all functional and non-functional requirements.

This includes testing real-time detection, accuracy of classification, system performance, usability, and reliability under different input conditions. System testing ensures that the system correctly processes images and videos, performs deepfake and liveness detection, displays accurate results with confidence scores, and remains stable during continuous usage. It serves as the final validation step before deployment or evaluation.

9.3. Test Cases

9.3.1. Introduction

Test cases are a structured set of conditions designed to evaluate whether specific functions or features of a system work correctly. Each test case defines the input data, the actions to be performed, and the expected output. Test cases provide a systematic and repeatable way to verify system behavior and ensure compliance with project requirements.

In this project, test cases are designed to validate functionalities such as media upload, preprocessing, deepfake detection, liveness verification, result generation, and user interaction. Executing these test cases helps identify errors, confirm correctness, and ensure consistent performance across different scenarios. Well-defined test cases reduce the risk of system failure in real-world use and ensure that the deepfake detection system meets both functional and quality expectations.

9.3.2 Unit Test Cases

Test Case ID	**Input**	**Description**	**Expected Output**	**Status**
UT-MU-01	Valid image file	Upload valid image	Image uploaded successfully	Pass
UT-MU-02	Valid video file	Upload valid video	Video uploaded successfully	Pass
UT-MU-03	Unsupported file type	Upload invalid format	Error message shown	Pass
UT-MU-04	Large file	Upload oversized file	Upload restricted message	Pass
UT-MU-05	No file selected	Submit without file	Warning displayed	Pass

Table 9.3.2.1 Unit Test Cases – Media Upload Unit

Test Case ID	**Input**	**Description**	**Expected Output**	**Status**
UT-FD-01	Image with single face	Detect face	Face detected	Pass
UT-FD-02	Image with multiple faces	Detect all faces	Faces detected	Pass
UT-FD-03	Image without face	Validate no-face case	Warning message shown	Pass
UT-FD-04	Blurred face image	Test detection accuracy	Face detected with warning	Pass
UT-FD-05	Low-light image	Detect face	Face detected	Pass

Table 9.3.2.2 Unit Test Cases – Face Detection Unit

Test Case ID	Input	Description	Expected Output	Status
UT-DD-01	Manipulated image	Detect deepfake	Classified as Fake	Pass
UT-DD-02	Genuine image	Detect real image	Classified as Real	Pass
UT-DD-03	Manipulated video	Detect deepfake video	Classified as Fake	Pass
UT-DD-04	Low-quality media	Test robustness	Correct classification	Pass
UT-DD-05	Repeated input	Test consistency	Same result produced	Pass

Table 9.3.2.3 Unit Test Cases – Deepfake Detection Unit

Test Case ID	Input	Description	Expected Output	Status
UT-LD-01	Video with eye blinking	Verify liveness	Liveness confirmed	Pass
UT-LD-02	Static image	Detect spoof	Liveness failed	Pass
UT-LD-03	Replay video	Detect replay attack	Liveness failed	Pass
UT-LD-04	Head movement video	Detect motion	Liveness confirmed	Pass
UT-LD-05	No facial motion	Test non-live input	Liveness failed	Pass

Table 9.3.2.4 Unit Test Cases – Liveness Detection Unit

Test Case ID	**Input**	**Description**	**Expected Output**	**Status**
UT-RD-01	Real media	Display real result	“Real” shown	Pass
UT-RD-02	Fake media	Display fake result	“Fake” shown	Pass
UT-RD-03	Detection confidence	Show confidence score	Percentage displayed	Pass
UT-RD-04	Error result	Display error message	Proper message shown	Pass
UT-RD-05	Multiple tests	Display updated results	Latest result shown	Pass

Table 9.3.2.5 Unit Test Cases – Result Display Unit

CHAPTER 10

RESULTS AND DISCUSSION

10.1 System Output Screenshots

Homepage of the website

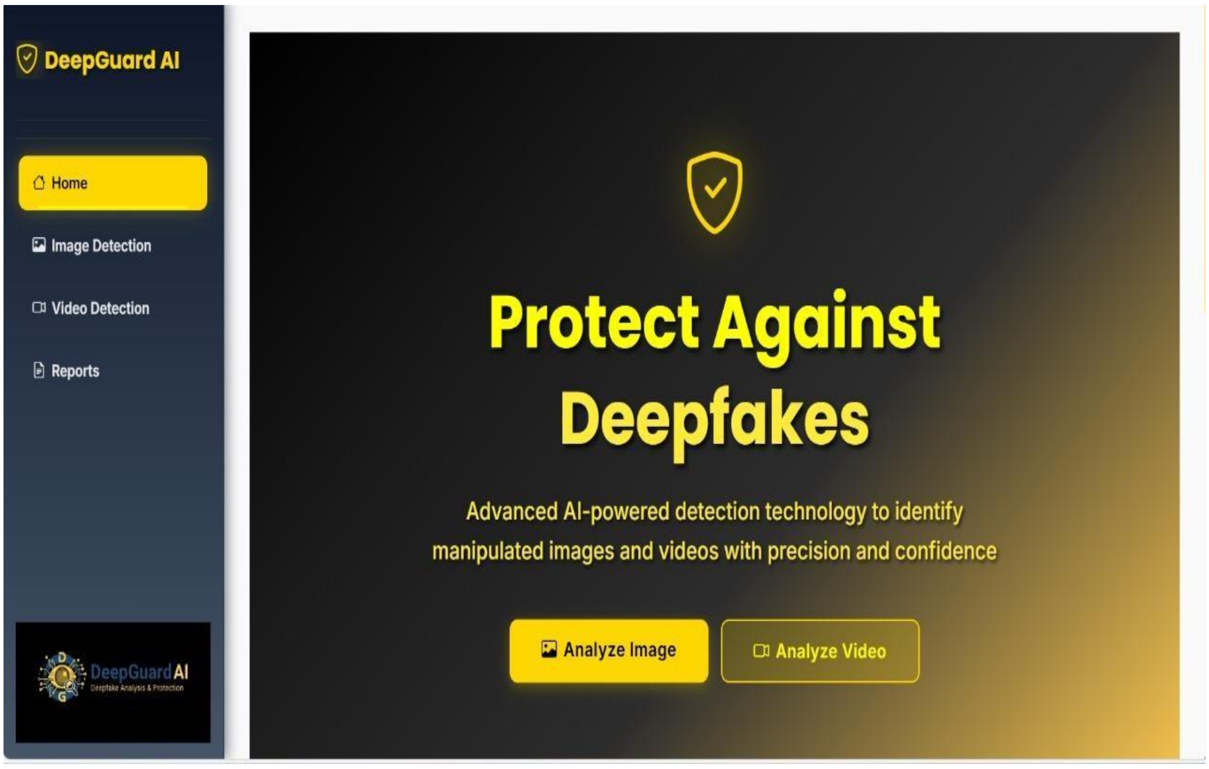

DeepGuard AI
Home
Image Detection
Video Detection
Reports
DeepGuard AI
Deepfake Analysis & Protection
Protect Against Deepfakes
Advanced AI-powered detection technology to identify manipulated images and videos with precision and confidence
Analyze Image
Analyze Video

Fig 10.1.1 Homepage interpretation of the project

DeepGuard AI is an intelligent deepfake detection platform designed to identify manipulated images and videos. It uses advanced AI models to analyze digital content with high accuracy. The interface provides options for image and video analysis through a clean, user-friendly dashboard. Users can quickly detect inconsistencies and verify media authenticity with confidence. The system enhances digital safety by protecting against deceptive and harmful deepfake content.

Image uploading unit

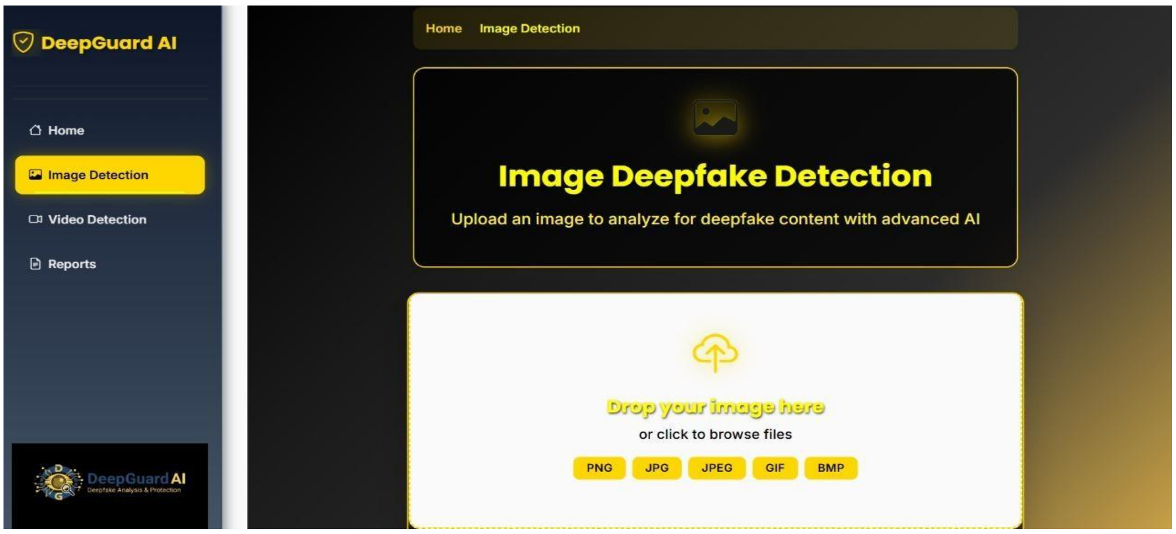


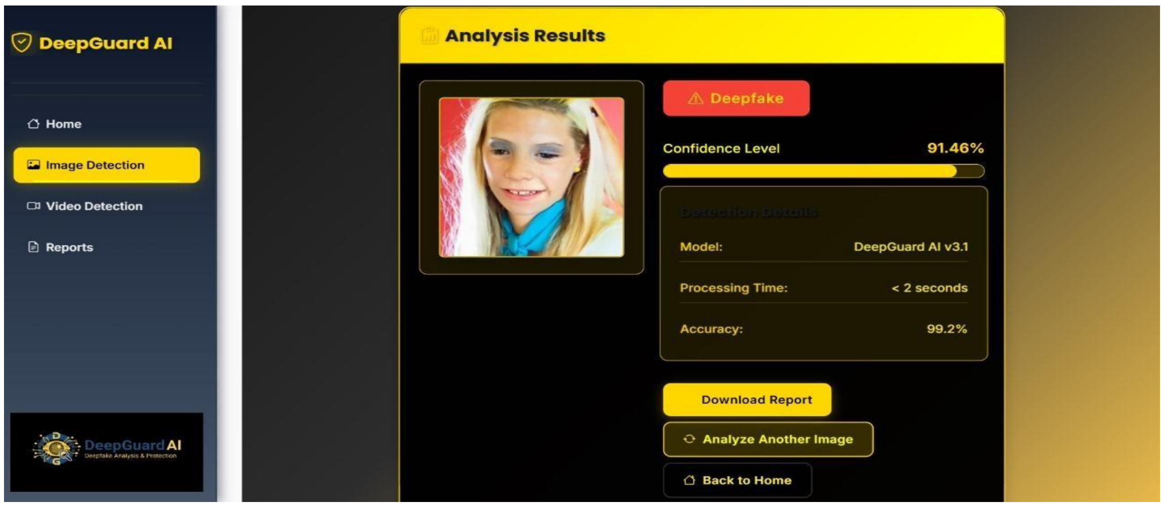
Image Deepfake Detection

Upload an image to analyze for deepfake content with advanced AI

Fig 10.1.2 Unit where the user can upload the spoofed picture

This screen shows the Image Deepfake Detection module of the DeepGuard AI system. This page enables fast and accurate deepfake identification through automated image analysis.

Image Analysis Result



Confidence Level 91.46%

Detection Details

Model: DeepGuard AI v3.1

Processing Time: < 2 seconds

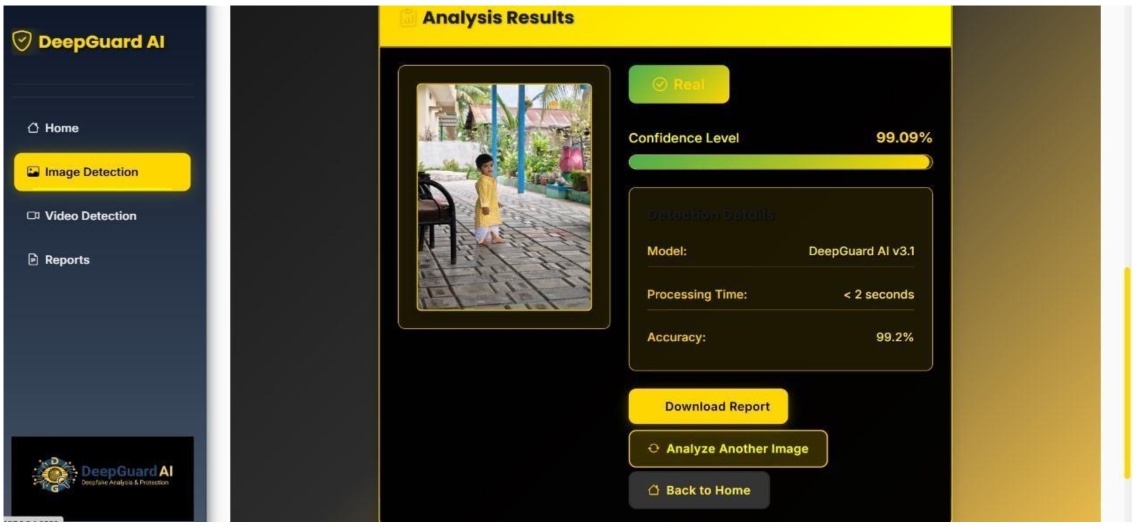
Accuracy: 99.2%

Fig 10.1.3 Fake image analysis result

The screen displays the Analysis Results page of DeepGuard AI after processing an uploaded image.

The system labels the image as a Deepfake with a confidence level of **91.46%**. Detection details such as the model version, processing time, and accuracy are clearly shown. Users can download a detailed report or choose to analyze another image.

The interface provides a clean, informative summary to help users understand the deepfake detection outcome.



Confidence Level 99.09%

Detection Details

Model: DeepGuard AI v3.1

Processing Time: < 2 seconds

Accuracy: 99.2%

Fig 10.1.4 Real image analysis result

The screen shows the Analysis Results page of DeepGuard AI for an uploaded image. The system classifies the image as Real with a high confidence level of **99.09%**. Details such as model version, processing time, and overall accuracy are displayed for transparency.

Users can download a detection report or proceed to analyze another image. This interface provides a clear and reliable summary confirming the authenticity of the analyzed media.

Video upload unit

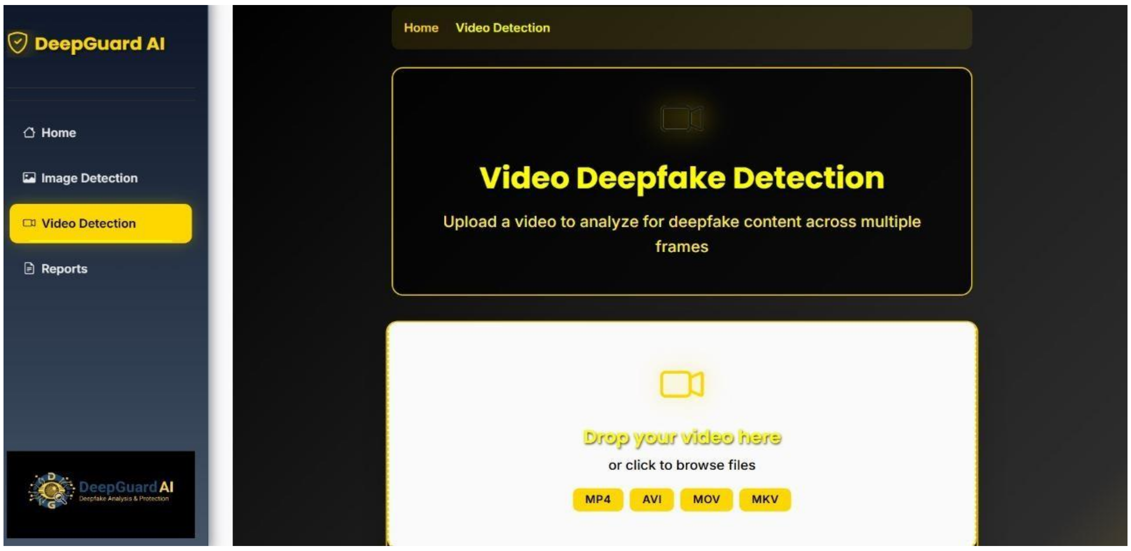


Fig 10.1.5 Unit where the user can upload the spoofed video

This screen displays the Video Deepfake Detection module of the DeepGuard AI system. Users can upload videos for analysis to detect manipulated content across multiple frames. A clean and intuitive layout guides users through the video detection workflow. This page enables efficient and accurate deepfake identification for video files.

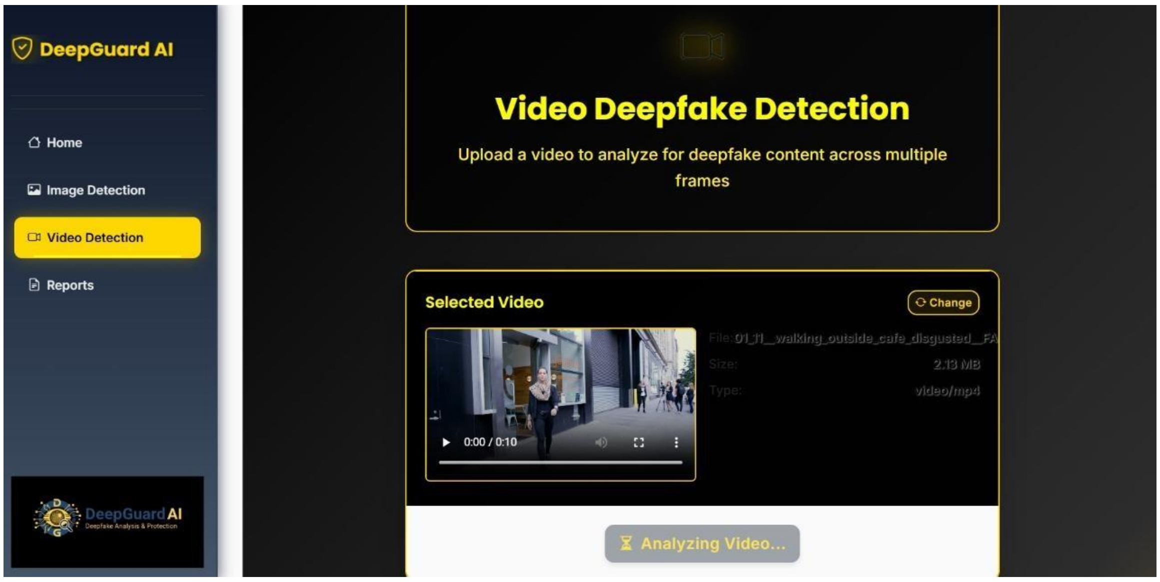
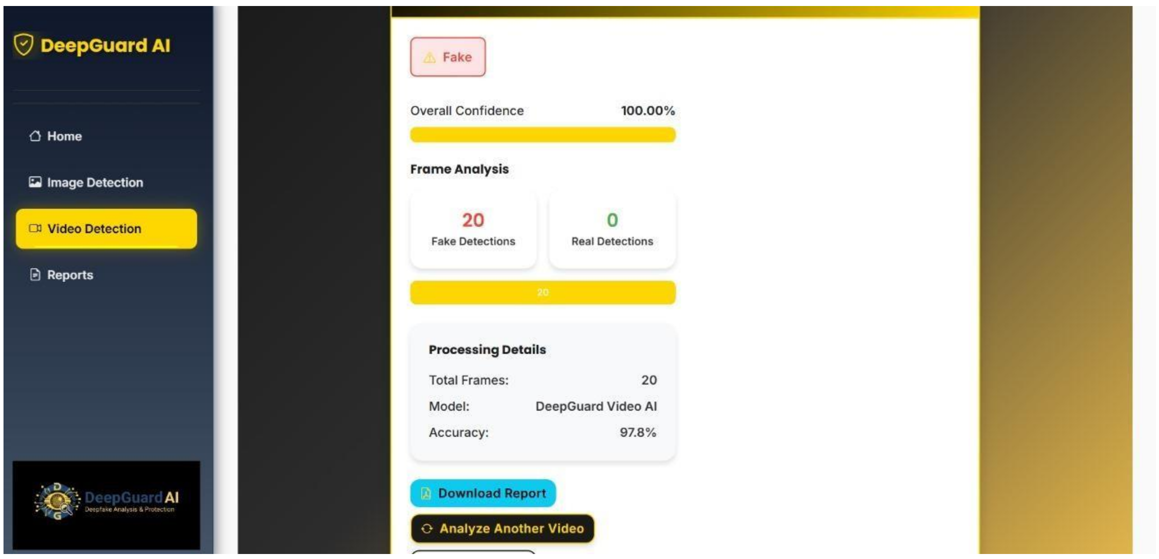


Fig 10.1.6 Video analysis part

This screen shows the Video Deepfake Detection process in DeepGuard AI after a video has been uploaded.

A preview of the selected video is displayed along with its file details for user confirmation. The system begins analyzing the video frame-by-frame to detect possible manipulations. A progress indicator labeled "Analyzing Video…" informs the user that detection is underway. The interface provides a smooth and guided workflow for deepfake analysis in video content.



Overall Confidence 100.00%

Frame Analysis

20 Fake Detections

0 Real Detections

20

Processing Details

Total Frames: 20

Model: DeepGuard Video AI

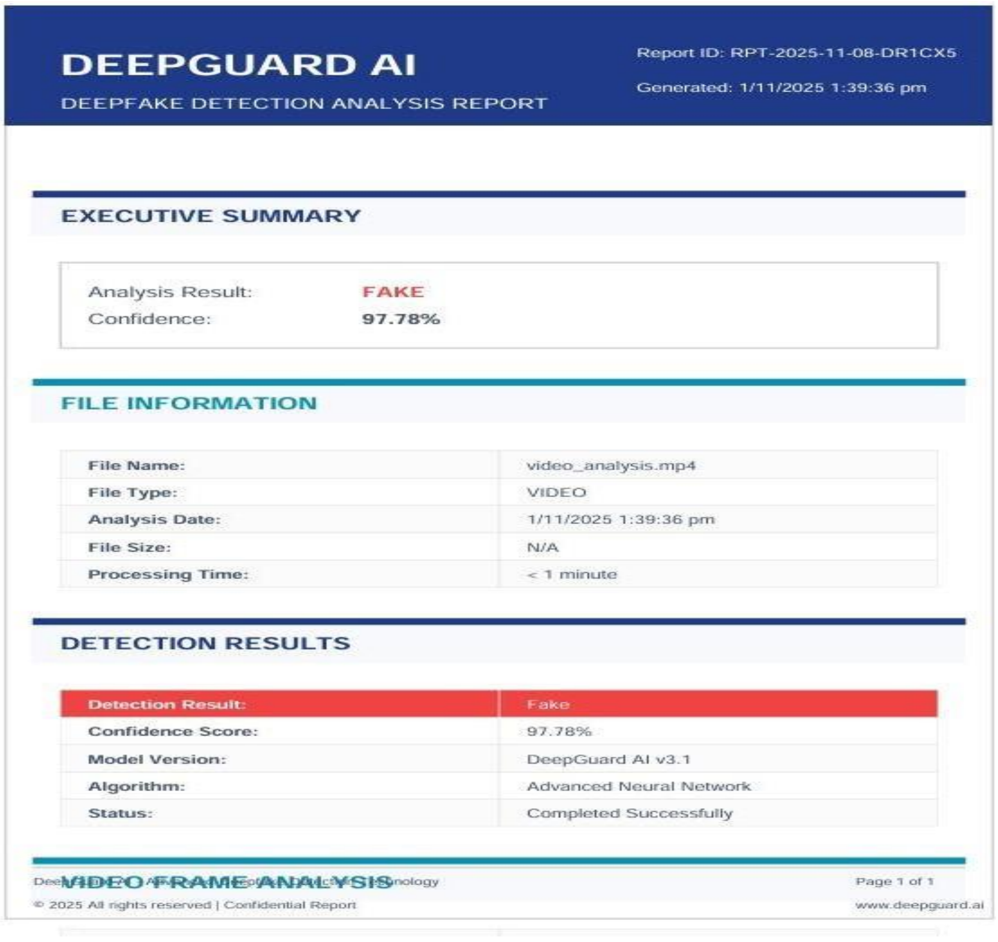
Accuracy: 97.8%

Fig 10.1.7 Fake video analysis result

This screen displays the video deepfake analysis results generated by DeepGuard AI. The system identifies the video as Fake with an overall confidence of 100%. Frame analysis shows 20 fake detections and **0** real frames, confirming complete manipulation. Processing details include total frames analyzed, model version, and detection accuracy of 97.8%.

Users can download the report or choose to analyze another video through the provided options.

Report



DEEPGUARD AI

DEEPFAKE DETECTION ANALYSIS REPORT

Report ID: RPT-2025-11-08-DR1CX5

Generated: 1/11/2025 1:39:36 pm

EXECUTIVE SUMMARY

Analysis Result:	FAKE
Confidence:	97.78%

FILE INFORMATION

File Name:	video_analysis.mp4
File Type:	VIDEO
Analysis Date:	1/11/2025 1:39:36 pm
File Size:	N/A
Processing Time:	< 1 minute

DETECTION RESULTS

Detection Result:	Fake
Confidence Score:	97.78%
Model Version:	DeepGuard AI v3.1
Algorithm:	Advanced Neural Network
Status:	Completed Successfully

VIDEO FRAME ANALYSIS

DeepGuard AI - Advanced Deepfake Detection Technology

Page 1 of 1

© 2025 All rights reserved | Confidential Report

www.deepguard.ai

Fig 10.1.8 DeepGuard AI – Video Deepfake Detection Report

This image shows the DeepGuard AI Deepfake Detection Analysis Report for a video file. The executive summary identifies the video as Fake with a confidence of 97.78%. File information such as name, type, analysis date, and processing time is clearly listed. The detection results detail the model version, algorithm used, and final detection status. This report provides a structured and professional summary of the deepfake analysis outcome.

CHAPTER 11

CONCLUSION

This project focuses on addressing the growing challenge of detecting deepfake images and videos in an era where AI-generated media is increasingly used for misinformation, identity fraud, and cyber threats. With the rapid advancement of deepfake technologies, verifying the authenticity of visual content has become difficult for individuals and organizations. To overcome this issue, an AI-powered deepfake detection system integrated with liveness detection has been developed to ensure reliable and secure media verification.

The proposed system allows users to upload images or videos and analyzes them using deep learning and computer vision techniques. It identifies manipulated or synthetic content by examining facial features, texture inconsistencies, and temporal irregularities. In addition, liveness detection verifies real-time human presence by analyzing eye blinking patterns, head movements, and facial motion, thereby preventing spoofing attacks such as photo replays or recorded videos. The system outputs a clear classification of “Real” or “Fake” along with a confidence score for better user understanding.

During testing, the system demonstrated high stability, accuracy, and consistent performance across different media formats and conditions. All major modules—including media upload, preprocessing, deepfake detection, liveness verification, and result display—functioned smoothly without significant errors. The system also showed efficient response time and reliable detection even when tested with larger video files and varying input quality.

Overall, the project successfully delivers a practical and effective solution for deepfake detection and liveness verification. The system enhances digital security, reduces the risk of identity fraud, and provides a user-friendly interface accessible to both technical and non-technical users. Future enhancements such as audio deepfake detection, mobile application deployment, real-time webcam verification, and multimodal analysis can further improve the system, making it suitable for widespread use in authentication systems, social media platforms, and cybercrime prevention.

CHAPTER 12

FUTURE SCOPE

The proposed **AI-Powered Deepfake Detection with Liveness Detection system** can be further enhanced and extended in several ways to improve accuracy, usability, and real-world applicability. As deepfake generation techniques continue to evolve, future work can focus on strengthening the system to handle more complex and unseen manipulation methods.

One important future enhancement is the integration of **audio deepfake detection** to identify manipulated voice recordings and lip-sync attacks. By combining audio and visual analysis, the system can become a **multimodal deepfake detection platform**, providing higher reliability and security. Advanced transformer-based and multimodal learning models can be employed to improve robustness against sophisticated deepfake attacks.

The system can also be extended to support **real-time webcam-based verification**, enabling live authentication for applications such as online examinations, remote interviews, digital banking, and secure access control. Additionally, deploying the system as a **mobile application** would increase accessibility and allow users to verify media authenticity anytime and anywhere.

Further improvements include training the system on larger and more diverse datasets to enhance generalization across different ethnicities, lighting conditions, and video qualities. The system can also be optimized for **cloud and edge computing environments** to support large-scale deployment with faster response times. With these enhancements, the project can evolve into a comprehensive AI-powered media authentication solution suitable for government, enterprise, and social media platforms.

CHAPTER 13

REFERENCES

[1] S. Sharma, A. Verma, and R. Kumar, "Artificial Intelligence for Deepfake Detection: A Systematic Review and Impact Analysis," *ResearchGate*, 2024.
Available: https://www.researchgate.net/publication/386304850

[2] Kaggle, "Deepfake Detection Challenge Dataset," Kaggle Competition Dataset, 2020.
Available: https://www.kaggle.com/c/deepfake-detection-challenge/data

[3] R. M. S., T. S. P., K. Sree, S. A., and N. R., "AI Deepfake Detection Research Paper," *International Journal of Novel Research and Development (IJNRD)*, vol. 8, no. 10, 2023.
Available: https://www.ijnrd.org/papers/IJNRD2310407.pdf

[4] A. Trivedi, K. Jangal, and R. Gupta, "Deepfake and Biometric Spoofing: AI-Driven Identity Fraud and Countermeasures," *ResearchGate*, 2025.
Available: https://www.researchgate.net/publication/390141504

[5] P. Author et al., "Deepfake Detection Techniques and Applications," *CEUR Workshop Proceedings*, vol. 3900, Paper 9, 2024.
Available: https://ceur-ws.org/Vol-3900/Paper9.pdf

[6] Google Scholar, "AI-Powered Deepfake Detection and Liveness Detection – Research Papers," Literature Survey Resource.
Available: https://scholar.google.co.in

[7] S. Banerjee, S. K. Yadav, A. Dhara, and M. Ajij, "Deepfake Detection: A Systematic Literature Review," *ResearchGate*, 2022.
Available: https://www.researchgate.net/publication/358835043

[8] M. Verdoliva et al., "Media Forensics and Deepfake Detection," *Electronics*, vol. 13, no. 1, Article 95, MDPI, 2024.
Available: https://www.mdpi.com/2079-9292/13/1/95

CHAPTER 14

CO – PO MATRIX

Sl. No.	Description	POs
1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and computer science and business systems to the solution of complex engineering and societal **problems.**	PO1
2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering and business problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.	PO2
3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	PO3
4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	PO4
5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations	PO5
6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering and business practices.	PO6
7	Environment and sustainability: Understand the impact of the professional engineering solutions in business societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	PO7

8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering and business practices.	PO8
9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	PO9
10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	PO10
11	Project management and finance: Demonstrate knowledge and understanding of the engineering, business and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	PO11

COURSE OUTCOME:

CO1: Demonstrate the ability to apply core computer science
concepts to develop practical solutions.

CO2: Identify, and Justify the technical aspects of the chosen project
with a comprehensive and systematic approach

CO3: Design, code, and test software solutions for specific problems.

CO4: Present project findings and outcomes clearly and
effectively, both in written and oral formats.

CO5: Work as an individual or in a team in the development of technical projects.

PROGRAM OUTCOMES:

PSO1. Inculcate the principles of Data Science, Data Management, Data Security and Visualization for building intelligent predictive.

PSO2. Applying the knowledge of analytics, statistics and Machine Learning concepts to solve real world business problems.

COURSE OUTCOME ASSESSMENT MATRIX:

Cos	Program Outcome's												
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO 1	3	2	2		3							3	2
CO 2	2	3	2	2								2	3
CO 3	2		3	2	3							3	3
CO 4												3	
CO 5								3		3	2		

CHAPTER 15

PUBLICATION

The research paper titled "AI-POWERED DEEPFAKE DETECTION: LIVENESS DETECTION" has been submitted to the ETFI 2026 International Conference. As informed by the conference organizing committee, the revised vention of the paper is currently under the review process. Some papers have already received acceptance, while the remaining submissions, including this paper, are still being evaluated. The final decision regarding acceptance or rejection is expected to be shared by the first week of January 2026. Till then, the authors have been requested to wait for the official confirmation.

CHAPTER16

Author's Biography

Project Guide	
	Dr. Jagadish R M Professor Dept.CSE – Data Science Ballari Institute of Technology and Management Ballari, Karnataka rm.jagadish@gmail.com
Projects Associates:	
	Chetana HK 3BR22CD007 B.E (CSE – Data Science) Ballari Institute of Technology and Management Ballari, Karnataka chetanahk1@gmail.com
	K Shashikala 3BR22CD023 B.E (CSE – Data Science) Ballari Institute of Technology and Management Ballari, Karnataka shashikala2928@gmail.com
	Mounika M 3BR22CD039 B.E (CSE – Data Science) Ballari Institute of Technology and Management Ballari, Karnataka mounikasrinivas444@gmail.com
	Pushpitha J R 3BR22CD046 B.E (CSE – Data Science) Ballari Institute of Technology and Management Ballari, Karnataka pushpithajr260903@gmail.com