iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78728

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

AI-Based Fake Content Detection System

Sambath kumar N[1], Sivaraman M[2], Thirukumaran R[3], Asst Prof. Mrs. R. Vijaya Lakshmi[4]
[1, 2, 3]*BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING ADHIYAMAAN COLLEGE OF ENGINEERING (AN AUTONOMOUS INSTITUTION) HOSUR ANNA UNIVERSITY: CHENNAI 600025*

Abstract: ***The LASSO: AI-Powered Fake Content Detection System is an intelligent digital application designed to identify and analyze potentially fake, manipulated, or malicious digital content across multiple formats such as text, images, audio, video, and URLs. In today's rapidly evolving digital environment, misinformation, deepfakes, phishing links, and AI-generated content have become major threats that can mislead individuals, compromise privacy, and cause financial or social harm. Traditional methods of identifying fake content often rely on manual verification or limited detection tools, which are inefficient and time-consuming. This project aims to address these challenges by developing a centralized and automated platform that uses artificial intelligence to detect suspicious or fake digital content quickly and accurately. The LASSO system provides users with a simple and interactive interface where they can input or upload different types of media for analysis. Users can paste suspicious text messages, upload images, audio files, or videos, and even scan URLs to verify their authenticity. The system processes the input using AI-based detection techniques and returns a result indicating whether the content is genuine, suspicious, or potentially fake, along with a confidence score. The application also maintains a scan history and generates downloadable reports, helping users keep track of previously analyzed content. Technically, the system is developed using the Flutter framework for the frontend, enabling a modern and responsive cross-platform mobile application interface. The backend integrates AI-based detection services and APIs that analyze the uploaded content and return detection results. The system architecture is designed to ensure efficient data processing, user-friendly navigation, and secure handling of uploaded files. Additional modules such as notification alerts, result visualization, and report generation further enhance the functionality of the application. In conclusion, the LASSO Fake Content Detection System provides a powerful solution to combat digital misinformation and fraudulent media by leveraging artificial intelligence and modern mobile technologies. By enabling users to quickly verify the authenticity of digital content, the system improves digital awareness, enhances online security, and contributes to the broader effort of reducing the spread of fake information in the digital world.***

I. INTRODUCTION

In the modern digital era, the rapid growth of internet usage and social media platforms has significantly increased the spread of misinformation, fake news, phishing links, and manipulated media such as deepfake images, videos, and audio. These forms of fake digital content can mislead individuals, damage reputations, and even cause financial or social harm. As digital technologies become more advanced, identifying fake or manipulated content through manual verification becomes increasingly difficult and time-consuming. Therefore, there is a growing need for intelligent systems that can automatically detect and analyze suspicious digital content to ensure authenticity and security in the online environment.

The LASSO: AI-Powered Fake Content Detection System is developed to address these challenges by providing an intelligent and user-friendly platform capable of analyzing different types of digital content. The main objective of the system is to help users quickly verify the authenticity of text messages, images, audio files, videos, and URLs using artificial intelligence–based detection techniques. Through this platform, users can upload or input suspicious content and receive instant analysis results indicating whether the content is genuine, suspicious, or potentially fake, along with a confidence score and additional insights.

The system also includes several supporting features that enhance the user experience and functionality of the application. These features include scan history tracking, result visualization, downloadable reports, and notification alerts to inform users about completed scans. By integrating these modules, the application provides a comprehensive environment for analyzing and monitoring digital content authenticity in a structured and organized manner.

II. OBJECTIVE

ThemainObjectiveoftheprojectis

1) To develop an intelligent system for detecting fake or manipulated digital content by using artificial intelligence techniques to analyze text, images, audio, video, and URLs.

©IJRASET: All Rights are Reserved

2) To provide users with a simple and accessible platform where they can upload or input suspicious digital content and receive instant analysis results regarding its authenticity.
3) To identify potential threats such as fake news, phishing links, and deepfake media, helping users make informed decisions and avoid digital fraud or misinformation.
4) To maintain a scan history and generate reports that allow users to track previously analyzed content and review the results when needed.
5) To improve digital awareness and cybersecurity by providing reliable tools that help individuals verify online information before trusting or sharing it.
6) To design a scalable and user-friendly mobile application using modern technologies such as Flutter and AI-based detection services to ensure efficient performance and easy user interaction across different devices.

III. LITERATURE SURVEY

1) The rapid growth of digital media and online communication platforms has led to an increasing spread of misinformation, fake news, phishing links, and deepfake media. As a result, researchers and technology developers have shown significant interest in creating intelligent systems capable of detecting fake or manipulated content. Several studies highlight that traditional manual verification methods are inefficient, time-consuming, and often unreliable when dealing with large volumes of digital information. These challenges have encouraged the development of automated fake content detection systems that utilize artificial intelligence and machine learning techniques to analyze and verify digital media authenticity.
2) A study published in the IEEE Conference on Information and Communication Technologies (2020) discussed the implementation of machine learning models for detecting fake news in textual data. The research demonstrated that algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Natural Language Processing (NLP) techniques can effectively identify misleading or fabricated information in online articles and social media posts. Similarly, research on phishing detection systems has shown that analyzing URL patterns and webpage characteristics can help identify malicious links that attempt to steal user credentials or sensitive information.
3) Another significant area of research focuses on deepfake detection, where artificial intelligence is used to analyze images, audio, and videos to determine whether they have been manipulated. Studies presented in journals such as the International Journal of Advanced Computer Science and Applications (IJACSA) emphasize the use of convolutional neural networks (CNNs) and deep learning models for identifying visual inconsistencies in images and videos. These systems analyze facial features, pixel patterns, and audio characteristics to detect signs of manipulation. Additionally, modern applications increasingly integrate user-friendly interfaces developed using mobile and web technologies to allow users to upload and analyze suspicious media easily.
4) Based on the findings from these studies, it is evident that AI-based detection systems provide a practical and efficient solution for combating digital misinformation and fraudulent media. However, many existing solutions focus on a single type of content, such as text or images, rather than offering a unified platform capable of analyzing multiple media formats. The proposed LASSO: AI-Powered Fake Content Detection System addresses this limitation by providing a comprehensive application capable of detecting fake or suspicious text, images, audio, video, and URLs within a single platform. By integrating modern mobile technologies and AI-based analysis services, the system offers a scalable, user-friendly solution that helps users quickly verify the authenticity of digital content.

IV. SYSTEM ANALYSIS

A. *EXISTING SYSTEM*

In the current digital environment, identifying fake or manipulated online content is often performed through manual verification or by using separate and limited tools. Users usually rely on their personal judgment, basic internet searches, or fact-checking websites to verify whether a piece of information, image, or video is genuine. However, with the rapid growth of social media platforms and AI-generated content, this approach has become increasingly difficult and unreliable. Fake news, phishing links, and deepfake media can spread quickly across online platforms, making it challenging for individuals to determine the authenticity of digital content.

The existing methods for detecting fake content are generally fragmented and lack integration. Different tools or platforms are available for specific tasks such as fake news detection, phishing URL analysis, or image verification, but they often operate independently and require users to switch between multiple applications or websites.

This not only makes the verification process time-consuming but also reduces efficiency and convenience for users. Additionally, many available systems focus only on a single type of content, such as text or images, rather than providing a unified platform capable of analyzing multiple media formats.

Another limitation of the existing systems is the lack of accessibility and user-friendly interfaces for general users. Many advanced fake content detection tools are developed mainly for researchers or cybersecurity professionals and may require technical knowledge to operate effectively. As a result, ordinary users may find it difficult to utilize these tools for verifying suspicious content. Furthermore, many platforms do not provide features such as scan history, result reports, or integrated notifications, which limits their usability for regular monitoring and analysis of digital content.

Security and scalability also remain major concerns in existing solutions. Some verification platforms do not provide secure handling of uploaded files or user data, which can pose privacy risks. Moreover, as the volume of digital content continues to grow, manual verification methods and limited detection tools become increasingly inefficient. These challenges highlight the need for an advanced and integrated system like the LASSO: AI-Powered Fake Content Detection System, which provides a centralized, automated, and user-friendly platform capable of analyzing multiple types of digital media efficiently and securely.

B. PROPOSED SYSTEM

The proposed LASSO: AI-Powered Fake Content Detection System is a modern digital application designed to overcome the limitations of traditional manual methods used to verify the authenticity of online content. The system provides a centralized and intelligent platform that allows users to analyze different types of digital media such as text, images, audio, video, and URLs in order to detect whether the content is genuine, suspicious, or potentially fake. By integrating artificial intelligence technologies, the system helps users quickly identify misleading information, phishing links, and manipulated media, thereby improving digital safety and awareness.

The application provides a simple and interactive interface where users can upload or input suspicious content for analysis. Users can paste text messages, upload media files such as images, audio, and videos, or submit website links for verification. The system processes the provided input using AI-based detection services and returns a result with a confidence score indicating the likelihood of the content being real or fake. The platform also maintains a scan history and allows users to generate downloadable reports for previously analyzed content, enabling better tracking and documentation of detection results.

The system is developed using modern mobile application technologies to ensure efficiency and user-friendliness. The Flutter framework is used to build a responsive and interactive frontend interface, allowing the application to run smoothly across different devices. The backend integrates AI-based detection APIs and services responsible for analyzing the uploaded data and generating accurate detection results. The architecture is designed to securely handle user inputs and ensure reliable processing of multiple types of digital media.

C. PROPOSED SOLUTION

The proposed LASSO: AI-Powered Fake Content Detection System is designed as a comprehensive digital solution to address the growing problem of misinformation, fake media, and malicious online content. The system provides a centralized platform where users can easily verify the authenticity of various types of digital content such as text, images, audio, video, and URLs. By integrating artificial intelligence and modern mobile technologies, the system eliminates the need for manual verification and provides faster, more accurate analysis of suspicious digital content.

The proposed solution allows users to upload or input different types of media through a simple and intuitive mobile interface. Users can paste suspicious text messages, upload images, audio files, or videos, and even check website links to determine whether they are genuine or potentially harmful. Once the content is submitted, the system processes the input using AI-based detection services and generates an analysis result along with a confidence score. This helps users quickly understand whether the content is real, fake, or suspicious, enabling them to make informed decisions before trusting or sharing the information.

Technically, the proposed solution is developed using the Flutter framework, which enables the creation of a responsive and modern mobile application interface. The backend integrates AI-based detection APIs and services that analyze the uploaded content and return accurate detection results. The system architecture is built to ensure efficient processing, secure data handling, and scalability for future expansion. Additional features such as improved detection models, advanced analytics, and cloud-based AI integration can be incorporated in future versions of the system.

Overall, the proposed LASSO Fake Content Detection System provides a smart and efficient alternative to traditional methods of verifying online information.

By automating the content verification process and integrating advanced AI technologies, the system enhances digital awareness, improves online safety, and helps reduce the spread of fake or misleading information across digital platforms. Its scalable and modular design ensures that it can adapt to evolving technological requirements and continue contributing to a safer and more reliable digital environment.

D. IDEATION & BRAIN STORMING

The development of the LASSO: AI-Powered Fake Content Detection System began with an extensive ideation and brainstorming phase aimed at identifying the growing challenges associated with fake digital content on the internet. With the rapid expansion of social media platforms, online communication channels, and AI-generated media, the spread of misinformation, phishing links, deepfake videos, and manipulated images has become a serious concern. The project team initiated discussions to understand how these issues affect everyday internet users and explored possible technological solutions that could help people easily verify the authenticity of digital content. The brainstorming process focused on identifying practical ways to use artificial intelligence and modern mobile technologies to create a system capable of analyzing multiple types of media efficiently.

During the initial ideation stage, the team examined the real-world problems faced by users when trying to verify suspicious content online. It was observed that most individuals rely on manual verification methods such as searching the internet, checking fact-checking websites, or trusting personal judgment. These methods are often unreliable and time-consuming, especially when dealing with sophisticated AI-generated media like deepfakes. Furthermore, existing tools for fake content detection are usually fragmented, with separate platforms available for detecting fake news, phishing links, or manipulated images. This lack of integration makes the verification process inconvenient for users. Based on these observations, the team concluded that developing a single, unified application capable of analyzing multiple types of digital content would be an effective solution to simplify and improve the verification process.

The brainstorming sessions then focused on defining the core objectives and functional features of the proposed system. Several ideas were proposed, including the ability to analyze different media formats such as text, images, audio, video, and URLs. The team also discussed features like displaying confidence scores for detection results, maintaining a scan history for users, and generating downloadable reports for analyzed content. Each feature was evaluated in terms of feasibility, technical complexity, and usefulness to the end user. The goal was to ensure that the application would not only be technically functional but also simple and intuitive for users with varying levels of digital literacy.

Once the major features were finalized, the team began exploring the technological framework and development tools required to build the system. After multiple discussions, it was decided to use the Flutter framework for frontend development to create a modern and responsive mobile application interface. Flutter was chosen due to its cross-platform capabilities and ability to provide a smooth user experience. For the backend functionality, the system would integrate AI-based detection services and APIs capable of analyzing uploaded digital content and returning detection results. This combination of technologies was considered suitable for building a scalable and efficient application capable of handling different media types.

The next stage of brainstorming involved designing the system architecture and workflow. Flowcharts and interface sketches were created to visualize how users would interact with the application. The workflow was designed so that users could upload or input suspicious content through a simple interface, after which the system would process the content using AI detection services and display the results along with a confidence level. Additional features such as notification alerts, scan history tracking, and downloadable reports were also incorporated into the system design. These elements were intended to improve transparency, usability, and accessibility for users analyzing digital content.

User experience and security considerations were also key topics during the brainstorming sessions. The team emphasized the need for a clean, intuitive interface that allows users to navigate the application easily. Since users would be uploading files and sensitive content for analysis, the system needed to ensure secure handling of data and protect user privacy. As a result, the design included secure communication between the application and detection services, along with structured data management practices to maintain system reliability and trustworthiness.

Finally, the ideation process concluded with the creation of a development roadmap and task distribution plan. Responsibilities were assigned among team members for different aspects of the project, including interface design, backend integration, testing, and documentation. The team also discussed potential future enhancements, such as integrating advanced deepfake detection models, improving analysis accuracy through machine learning, and adding cloud-based analytics features. These ideas were documented to ensure that the system could evolve and adapt to future technological advancements.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

In conclusion, the ideation and brainstorming phase played a vital role in shaping the LASSO Fake Content Detection System into a structured and innovative solution for combating digital misinformation. Through collaborative discussions, problem analysis, and technology exploration, the team was able to transform an initial concept into a well-defined system design. This phase not only clarified the project objectives and technical requirements but also encouraged creative thinking and teamwork, ultimately laying a strong foundation for the successful development of the application.

E. PROBLEM SOLUTION FIT

The concept of Problem–Solution Fit focuses on ensuring that the developed system effectively addresses real-world challenges faced by users in the digital environment. For the LASSO: AI-Powered Fake Content Detection System, this involves identifying the growing problems associated with fake news, phishing links, and manipulated digital media, and designing an intelligent technological solution that helps users verify the authenticity of online content. The alignment between the identified problem and the proposed solution determines the effectiveness, usefulness, and success of the system.

1) Identified Problems

In the modern digital world, the rapid spread of misinformation and manipulated media has become a serious challenge. Social media platforms, messaging applications, and online websites allow information to be shared instantly, but they also make it easier for fake news, phishing links, and AI-generated media to circulate widely. Many users rely on manual verification methods such as searching for information online or trusting personal judgment, which are often unreliable and time-consuming. As a result, individuals may unknowingly trust or share misleading information.

2) Proposed Solution

To address these challenges, the LASSO Fake Content Detection System provides a centralized and intelligent platform that enables users to analyze and verify different types of digital media quickly and efficiently. The application allows users to upload or input suspicious content such as text messages, images, audio files, videos, and URLs. Once the content is submitted, the system processes it using AI-based detection services and provides a result indicating whether the content is genuine, suspicious, or potentially fake.

3) Fit Analysis

The proposed LASSO Fake Content Detection System effectively aligns with the identified problems by providing an automated, integrated, and user-friendly solution for detecting fake digital content. By combining multiple detection capabilities within a single platform, the system eliminates the need for users to rely on separate tools for verifying different types of media. This centralized approach improves efficiency and accessibility for users.

F. ARCHITECTURE DESIGN:

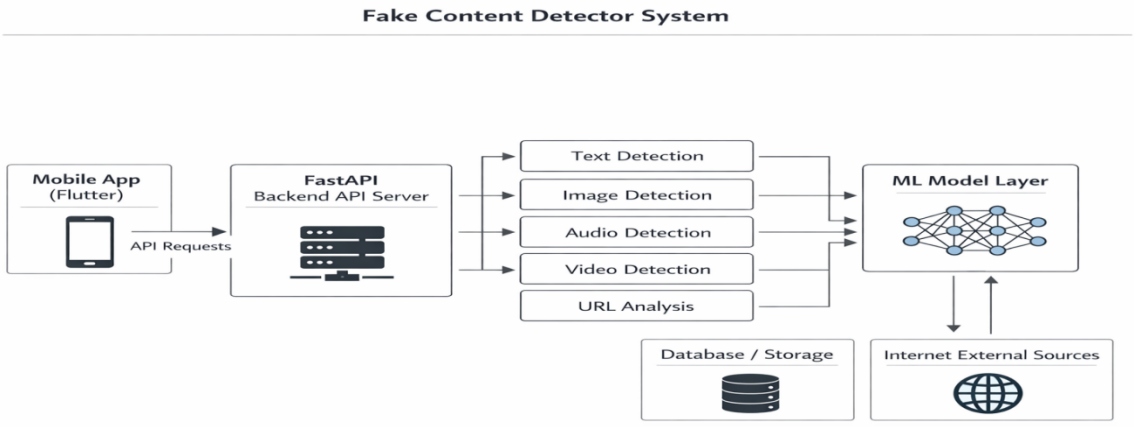


Figure1:Model Architecture

ThefigureshownaboverepresentstheSolutionArchitecturethatwemadeuseofinourProject.

The architectural diagram of the Fake Content Detector System (LASSO) illustrates how different components of the system interact to analyze and verify the authenticity of digital content. The architecture is designed to provide an efficient and structured workflow that connects the mobile application, backend server, machine learning models, and data sources. This layered architecture ensures smooth communication between all system components while maintaining scalability, security, and efficient processing of user requests.

The process begins when a user interacts with the mobile application developed using Flutter. Through the application interface, users can submit different types of digital content such as text, images, audio files, videos, or suspicious URLs for analysis. The mobile application sends these inputs as API requests to the backend server. The backend of the system is implemented using FastAPI, which acts as the central processing unit that receives, manages, and routes incoming requests from the mobile application.

Once the FastAPI backend receives the request, it directs the data to the appropriate content detection modules depending on the type of input provided by the user. These modules include Text Detection, Image Detection, Audio Detection, Video Detection, and URL Analysis. Each module is responsible for preprocessing and preparing the submitted content for further analysis. These detection modules act as intermediate processing units that communicate with the Machine Learning Model Layer.

The Machine Learning Model Layer performs the core functionality of the system by analyzing the content using trained machine learning algorithms. These models evaluate patterns, features, and characteristics within the data to determine whether the content is genuine, manipulated, or suspicious. The ML layer may also interact with external internet sources to gather additional contextual information required for more accurate detection.

Additionally, the system is supported by a database and storage component, which stores important information such as scan results, historical analysis records, and system data. This database enables efficient data management and allows users to review previous scans or generate reports based on stored results. The storage layer also helps improve system performance by maintaining structured records of processed content.

Overall, this architectural design ensures that the Fake Content Detector System operates in a modular and scalable manner. Each component performs a specific function while maintaining seamless communication with other modules in the system. By integrating the mobile application, backend server, detection modules, machine learning models, and database storage, the system provides a reliable and efficient platform for detecting fake or manipulated digital content.

G. DESCRIPTION OF MODULES

1) USER INTERACTION MODULE

The User Interaction Module is designed for individuals who want to verify the authenticity of digital content. This module provides users with a simple and intuitive interface through which they can access the system's main features. Users can open the mobile application, navigate through different detection options, and submit content such as text, images, audio, video, or URLs for analysis. The system ensures a user-friendly experience by offering a clean interface, responsive design, and easy navigation so that users with minimal technical knowledge can operate the application effectively.

Once the user submits the content, the system securely sends the data to the backend server for processing. After analysis is completed, the results are displayed on the result screen along with a confidence score indicating the reliability of the detection. Users can also view previously scanned results and access reports generated by the system. This module ensures smooth interaction between the user and the detection system while maintaining data privacy and secure communication.

2) CONTENT ANALYSIS MODULE

The Content Analysis Module is responsible for processing and analyzing the digital content submitted by the user. This module acts as the core functional component of the system, where different types of media are examined to determine whether they are authentic or manipulated. The system supports multiple types of analysis including text detection, image detection, audio detection, video detection, and URL analysis.

Each type of content follows a specific analysis pipeline where the input data is preprocessed and then forwarded to machine learning models for evaluation. The module identifies patterns, suspicious elements, and characteristics that may indicate fake or manipulated content. By handling multiple types of digital inputs, this module ensures that the system provides comprehensive detection capabilities in a single integrated platform.

3) MACHINE LEARNING PROCESSING MODULE

The Machine Learning Processing Module is responsible for performing the core intelligence of the Fake Content Detector System. This module uses trained machine learning models to analyze digital media and identify potential signs of manipulation, misinformation, or malicious content. These models examine features such as linguistic patterns in text, visual inconsistencies in images and videos, audio characteristics, and suspicious URL structures.

The machine learning models receive processed data from the analysis module and generate predictions regarding the authenticity of the content. The results are then returned to the application along with a confidence score that indicates the likelihood of the content being real or fake. This module significantly improves the accuracy and efficiency of the detection process by automating complex analysis tasks that would otherwise require manual verification.

4) API INTEGRATION MODULE

The API Integration Module plays a crucial role in connecting the mobile application with the backend detection services. This module manages the communication between the Flutter mobile application and the FastAPI backend server. When a user submits content such as text, images, audio, video, or URLs for analysis, the application sends the request through API calls to the backend system. The API module ensures that the data is properly formatted, transmitted securely, and received by the appropriate detection service.

Once the backend receives the request, the API module routes the input to the correct analysis module, such as text detection, image detection, audio detection, video detection, or URL analysis. The processed results generated by the machine learning models are then returned to the mobile application through the same API channel. This module ensures efficient and reliable data exchange between different components of the system.

Additionally, the API Integration Module handles request validation, error handling, and response management to maintain system stability. It ensures that user inputs are correctly processed and that results are delivered quickly and accurately. By managing communication between the frontend application and backend detection services, this module serves as the backbone of the system's functionality, enabling seamless interaction between all architectural components of the Fake Content Detector System.

H. DATA FLOW DIAGRAM:

The Data Flow Diagram (DFD) of the LASSO: AI-Powered Fake Content Detection System illustrates how data moves between the user, the mobile application, the backend server, the machine learning models, and the database to perform fake content detection efficiently. The system is designed to ensure a structured and secure flow of information from the moment the user submits content until the final detection result is displayed. This organized data flow allows the system to process different types of digital content such as text, images, audio, video, and URLs in a reliable and automated manner.The process begins when a user interacts with the mobile application developed using Flutter. Through the application interface, the user can input or upload suspicious content such as text messages, images, audio files, videos, or website URLs for verification. Once the content is submitted, the application sends the request to the backend server through API calls. The backend server, implemented using FastAPI, receives the request and routes the data to the appropriate detection module depending on the type of content provided by the user.

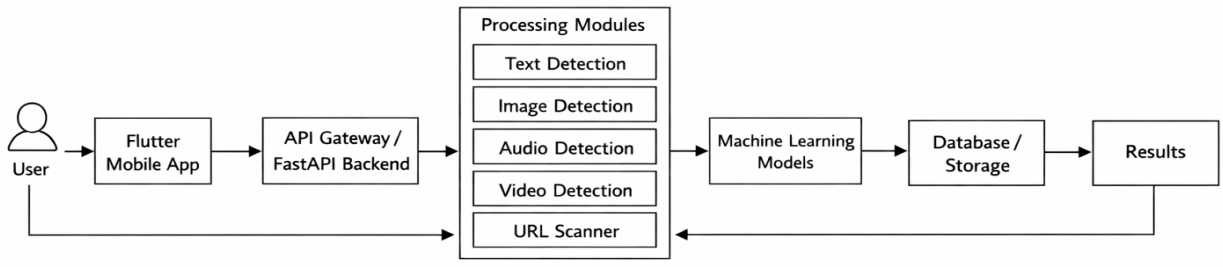


Figure2:DataFlowDiagram

V. SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENT

- Processor :A minimum of Intel Core i3 processor or AMD equivalent is required to support the development and execution of the Fake Content Detection System. For improved performance and faster processing of AI-based detection tasks, a Core i5 processor or higher is recommended.
- RAM :The system should have at least 4 GB of RAM to run the mobile development environment and backend services smoothly. However, 8 GB or more RAM is recommended for better performance, especially when running multiple tools such as the Flutter development environment, backend server, and machine learning services simultaneously.
- Hard Disk Space :A minimum of 250 GB storage capacity is required to store project source code, development tools, dependencies, datasets, and application files. Additional storage space is recommended for storing logs, backups, and system updates.
- Network Interface and Connectivity :A stable internet connection (Wi-Fi or broadband) is necessary for downloading development frameworks, accessing APIs, and communicating with backend detection services. The system must include a Network Interface Card (NIC) to enable proper network communication between the mobile application, backend server, and AI detection services.

B. SOFTWARE REQUIREMENTS

- Operating System :The application can run on Windows 10 or later, Linux (Ubuntu), or macOS. These operating systems provide a stable environment for mobile application development and backend integration, supporting necessary tools such as Flutter SDK, Python, and API services.
- Programming Languages and Frameworks : The system is developed using Dart with the Flutter framework for building the mobile application interface. The backend services are implemented using Python with FastAPI, which handles API requests, data processing, and communication with machine learning models. This combination ensures efficient application performance and scalable system architecture.
- Database Management System (DBMS) :The system uses a lightweight database system such as SQLite or local storage to maintain scan history, results, and user activity records. This database enables efficient storage and retrieval of analysis results while maintaining data integrity.
- Development Tools and IDEs :Recommended development tools include Visual Studio Code, Android Studio, or PyCharm for coding, debugging, and testing the application. These IDEs provide features such as syntax highlighting, debugging tools, integrated terminals, and plugin support that improve development efficiency.
- WebServer and Browser :Modern web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge are used for testing backend APIs and system interfaces. Additionally, Android Emulator or a physical Android device is required for testing and running the Flutter mobile application during development.
- Server and API Tools :The backend system uses FastAPI as the web framework to handle API communication between the mobile application and machine learning detection modules. API testing and development can performed using tools such as Postman or Swagger UI to ensure proper data .

VI. IMPLEMENTATION

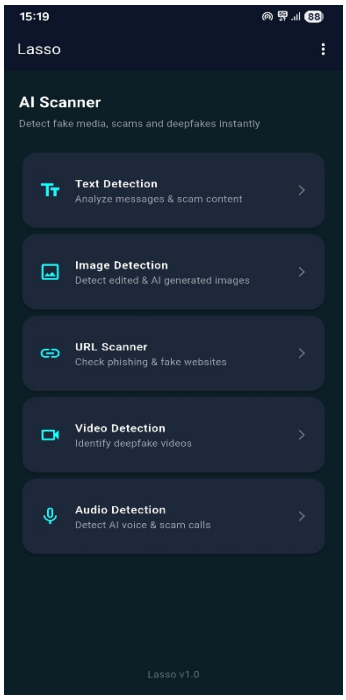
A. HOME PAGE:

The Home Page serves as the central interface of the LASSO: AI-Powered Fake Content Detection System, acting as the primary navigation hub for users after launching the application. It is designed to provide quick access to the different detection functionalities available in the system. The home page is implemented using the Flutter framework, which enables the development of a modern, responsive, and visually appealing mobile interface. The page layout includes clearly organized modules such as Text Detection, Image Detection, Audio Detection, Video Detection, and URL Scanner, allowing users to easily select the type of content they want to analyze. Each option is represented by an icon-based card or button to ensure intuitive navigation and improve the overall user experience.

The user interface of the home page is developed using Flutter's Material Design components, ensuring a clean layout and smooth interaction across various mobile devices. The design emphasizes simplicity and accessibility by arranging detection modules in a structured grid format with consistent color themes and icons.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

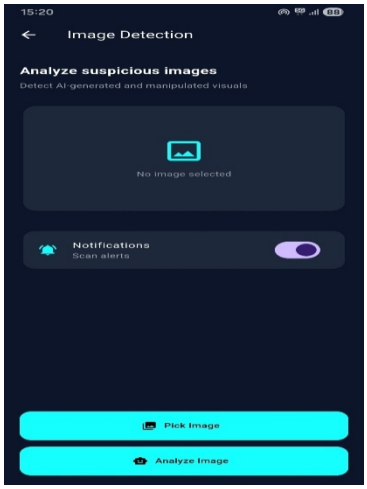
Navigation between screens is handled using Flutter's Navigator and routing system, which allows users to move from the home page to specific detection screens seamlessly. Additionally, a menu option or settings icon may be provided on the home screen to allow users to access features such as profile settings, scan history, and application configuration a user initiates a detection task.



B. IMAGE DETECTION PAGE:

The Image Detection Page is an important component of the LASSO: AI-Powered Fake Content Detection System, designed to allow users to verify whether an image is genuine or AI-generated. This page provides a simple and intuitive interface where users can upload images from their device for analysis. The frontend of the image detection page is implemented using the Flutter framework, which enables a responsive and visually organized mobile interface. The page includes features such as an image upload button, preview section, and analyze button, allowing users to easily select and review the image before submitting it for detection. Flutter's Material Design components are used to create a clean layout with icons, cards, and interactive buttons, ensuring a smooth user experience across different mobile devices.

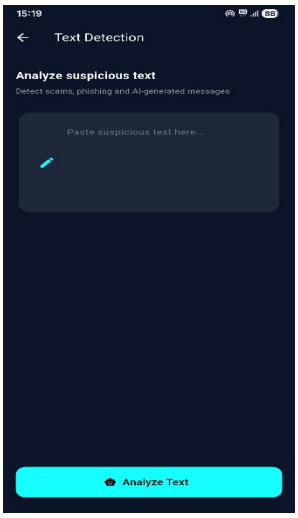
When a user selects an image using the upload option, the system utilizes Flutter plugins such as Image Picker to access the device gallery and retrieve the selected image file. Once the image is chosen, it is displayed in a preview container so that users can confirm the selected file before proceeding with analysis. After the user presses the Analyze Image button, the application sends the image data to the backend server through API requests. This request is processed by the backend service implemented using FastAPI, which acts as the communication bridge between the mobile application and the machine learning detection models.



C. TEXT DETECTION PAGE:

The Text Detection Page is a key feature of the LASSO: AI-Powered Fake Content Detection System, designed to allow users to analyze suspicious text and determine whether it is genuine, AI-generated, or potentially harmful. This page provides a simple and user-friendly interface where users can enter or paste text content for analysis. The frontend of the text detection page is implemented using the Flutter framework, which ensures a responsive and interactive mobile application interface. The page includes components such as a text input field, analyze button, and result display area, enabling users to easily submit text for verification and view the analysis results.

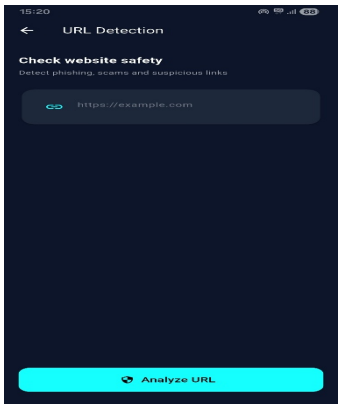
The user interface is developed using Flutter Material Design widgets, which provide a clean layout and smooth interaction. A multi-line text field allows users to paste or type suspicious messages such as emails, social media posts, or website content. Before sending the data to the backend, the application performs basic validation to ensure that the input field is not empty. Once the user presses the Analyze Text button, the application sends the entered text to the backend server through API requests for further processing.



D. URL DETECTION PAGE:

The URL Detection Page is an important component of the LASSO: AI-Powered Fake Content Detection System, designed to help users verify whether a website link is safe or potentially malicious. This page provides a simple and intuitive interface where users can paste suspicious URLs for analysis. The frontend of the URL detection page is implemented using the Flutter framework, which ensures a responsive and user-friendly mobile interface. The page includes elements such as a URL input field, analyze button, and result display section, allowing users to easily submit a website link and receive the detection results.

The user interface is built using Flutter Material Design widgets, ensuring a clean layout and smooth interaction across different mobile devices. The page contains a text input field where users can paste or type a suspicious URL. Before processing the request, the application performs basic validation to ensure that the input field is not empty and that the entered text follows a valid URL format. When the user presses the Analyze URL button, the application sends the submitted URL to the backend server through API requests.

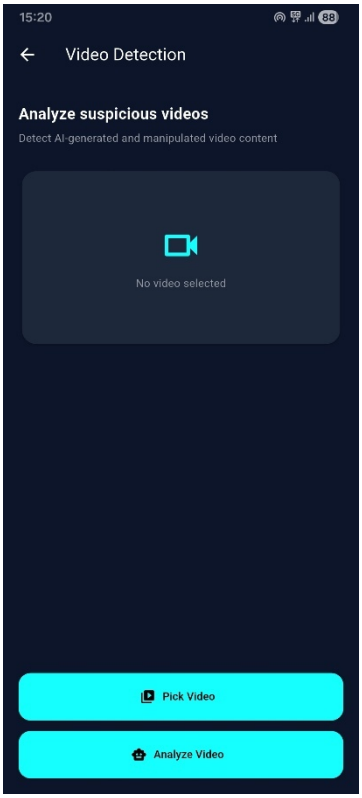


©IJRASET: All Rights are Reserved

E. VIDEO DETECTION PAGE:

The Video Detection Page is an important feature of the LASSO: AI-Powered Fake Content Detection System, designed to help users verify whether a video is authentic or potentially manipulated. This page provides a user-friendly interface where users can upload video files from their device for analysis. The frontend of the video detection page is implemented using the Flutter framework, which ensures a responsive and visually organized mobile interface. The page includes elements such as a video upload button, video preview section, and analyze button, allowing users to easily select a video file and review it before initiating the detection process.

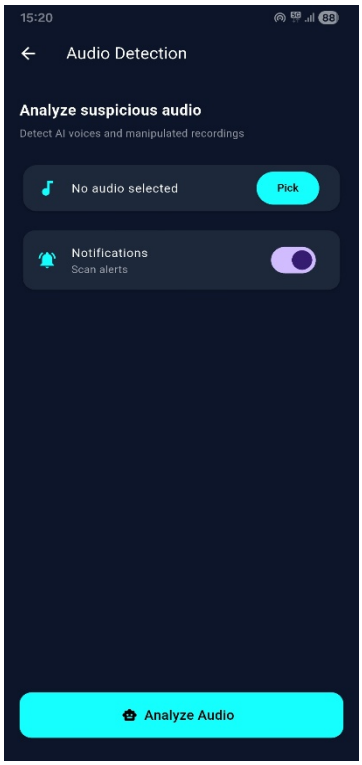
The user interface is developed using Flutter Material Design components, which provide a clean layout and smooth interaction across different mobile devices. Users can select a video file using file selection features integrated through Flutter plugins such as File Selector or Video Picker. Once a video is selected, the application displays a preview of the video using a video player component, allowing users to confirm the selected file before proceeding. After the user presses the Analyze Video button, the application sends the selected video file to the backend server through API requests.



F. AUDIO DETECTION PAGE:

The Audio Detection Page is an important feature of the LASSO: AI-Powered Fake Content Detection System, designed to allow users to analyze audio files and determine whether the audio content is genuine or artificially generated. This page provides a simple and intuitive interface where users can upload audio recordings for verification. The frontend of the audio detection page is implemented using the Flutter framework, which ensures a responsive and user-friendly mobile application interface. The page contains elements such as an audio upload button, file information display, and analyze button, enabling users to easily select and submit audio files for analysis.

The user interface is developed using Flutter Material Design components, which provide a clean layout and smooth interaction across different mobile devices. Users can select audio files from their device using file selection features integrated through Flutter plugins such as File Selector or similar file-picking libraries. Once an audio file is selected, the application displays basic file details such as the file name and size, allowing users to confirm the selected audio before proceeding with analysis. After the user presses the Analyze Audio button, the application sends the audio file to the backend server through API requests.



G. SOFTWARE DESCRIPTION:

1) VS CODE

Android Studio was chosen as the primary Integrated Development Environment (IDE) for developing the LASSO: AI-Powered Fake Content Detection System due to its powerful features, robust performance, and seamless integration with mobile development frameworks. It provides a comprehensive environment for building, testing, and debugging mobile applications efficiently. Android Studio supports the Flutter framework and Dart programming language, which are used to develop the mobile interface of this application. The IDE includes features such as syntax highlighting, intelligent code completion, debugging tools, and an integrated terminal, which help developers write and manage code more effectively.

Android Studio allows developers to organize project files in a structured manner using a clear directory system. The project structure typically includes folders for screens, services, widgets, and assets, enabling easy navigation and maintenance of the codebase. Developers can easily manage multiple components of the application such as the home page, text detection page, image detection page, audio detection page, video detection page, and URL detection page within the same environment. The built-in terminal also allows developers to execute Flutter commands such as flutter run, flutter build, and flutter pub get, enabling smooth project compilation and dependency management without leaving the IDE.

Another major advantage of Android Studio is its built-in Android Emulator, which allows developers to test the application without requiring a physical mobile device. The emulator simulates different Android devices and screen sizes, helping developers ensure that the application functions correctly across multiple devices. Android Studio also provides performance monitoring tools, layout inspection, and debugging capabilities that help identify and fix errors during development.

Android Studio also supports version control integration, allowing developers to manage project versions using systems such as Git. This enables developers to track code changes, collaborate with team members, and maintain backups of the project. By integrating version control directly into the IDE, developers can commit changes, review modifications, and synchronize project files with remote repositories efficiently.

2) FRONTEND:

The frontend of the LASSO: AI-Powered Fake Content Detection System represents the user-facing layer of the mobile application, responsible for user interaction, navigation, and visual presentation of the system's features. It is developed using the Flutter framework with the Dart programming language, which allows the creation of a responsive and interactive mobile interface.

Flutter provides a rich set of widgets that help design structured layouts such as headers, buttons, input fields, cards, and navigation components. These elements are used to build different pages of the application including the Home Page, Text Detection Page, Image Detection Page, Audio Detection Page, Video Detection Page, URL Detection Page, and Result Page.

The user interface is designed using Flutter Material Design components, ensuring a modern and visually consistent layout across all screens. The home page displays detection modules in the form of icons or cards, allowing users to quickly access different features of the application. Each detection page contains input fields, upload buttons, preview sections, and analyze buttons that enable users to submit different types of digital content for analysis. The application follows a clean and minimalist design with a consistent color theme and organized layout to improve usability and readability.

Navigation between screens is implemented using Flutter's Navigator and routing system, allowing smooth transitions from one page to another. For example, when a user selects a detection module from the home page, the application navigates to the corresponding detection screen where the user can upload or enter content. Flutter's layout widgets such as Column, Row, Container, GridView, and Card are used to structure the interface effectively and ensure the application adapts well to different mobile screen sizes.

3) BACKEND :

The backend of the LASSO: AI-Powered Fake Content Detection System is implemented using FastAPI, a modern web framework built on the Python programming language. FastAPI is selected because of its high performance, simplicity, and strong support for building RESTful APIs that can efficiently communicate with mobile applications. The backend acts as the core processing layer of the system, handling all application logic, managing API requests, and connecting the mobile frontend with the machine learning detection modules. It ensures that user requests such as text analysis, image uploads, audio or video detection, and URL verification are processed quickly and securely.

The backend architecture follows a modular design structure, which separates different system functionalities into organized components such as API services, detection modules, and data processing units. The API layer receives requests from the Flutter mobile application and routes them to the appropriate detection module depending on the type of content submitted by the user. For example, text input is forwarded to the text detection module, while uploaded media files such as images, audio, or video are sent to their respective analysis modules. This modular design improves maintainability, scalability, and system efficiency.

The backend is also responsible for integrating the machine learning models that perform the core detection tasks. These models analyze patterns and features within the submitted data to determine whether the content is genuine, manipulated, or potentially fake. After the analysis is completed, the backend generates a result along with a confidence score, which is then returned to the mobile application through structured API responses. This process ensures smooth communication between the frontend interface and the detection engine.

4) DATABASE:

The database component of the LASSO: AI-Powered Fake Content Detection System is implemented using SQLite, a lightweight and efficient relational database management system that is suitable for mobile and API-based applications. SQLite is chosen because of its simplicity, fast performance, and minimal configuration requirements. It allows the system to store and manage important data such as scan results, user activity records, detection logs, and analysis history. The database ensures that all processed information is stored securely and can be retrieved whenever required by the application.

The database design follows a structured relational model, where different types of data are organized into separate tables for efficient storage and retrieval. Key tables in the system include the Scan History Table, which stores records of analyzed content along with detection results and confidence scores; the Content Analysis Table, which stores information about uploaded files such as images, audio, video, text, or URLs; and the System Logs Table, which maintains records of processing activities and system operations. Each table is structured with appropriate attributes such as scan ID, content type, result status, confidence level, and timestamp.

H. CODE IMPLEMENTATION

Step1:Implement the Text Detection API

The text detection functionality of the LASSO Fake Content Detection System is implemented using FastAPI. This module receives text input from the mobile application and analyzes it to determine whether the content is safe or suspicious. The backend processes the request and returns the detection result along with a confidence score.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

CodeSnippet:

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class TextRequest(BaseModel):
    text: str

@app.post("/detect-text")
def detect_text(data: TextRequest):
text_input = data.text

    if "scam" in text_input.lower() or "fake" in text_input.lower():
        result = "HIGH RISK"
        confidence = "85%"
    else:
        result = "SAFE"
        confidence = "70%"

    return {
        "result": result,
        "confidence": confidence
    }
```

Step2: Implement the Image Detection API

The Image Detection Module of the LASSO Fake Content Detection System is implemented using FastAPI, which allows the backend server to receive image files from the mobile application and analyze them for authenticity. This module processes images uploaded by users and determines whether the image is genuine or AI-generated. The API endpoint receives the image file, performs basic validation, and forwards the image to the detection logic or machine learning model responsible for image analysis.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Code Snippet:

```
from fastapi import FastAPI, File, UploadFile

app = FastAPI()

@app.post("/detect-image")
async def detect_image(file: UploadFile = File(...)):

    # Example detection logic (placeholder)
    filename = file.filename

    if "ai" in filename.lower() or "fake" in filename.lower():
        result = "FAKE IMAGE"
        confidence = "88%"
    else:
        result = "REAL IMAGE"
        confidence = "72%"

    return {
        "result": result,
        "confidence": confidence
    }
```

Step3:Implement the Video Detection API

The Video Detection Module of the LASSO Fake Content Detection System is implemented using FastAPI, allowing the backend server to receive video files from the mobile application and analyze them for possible manipulation or deepfake content. This module enables users to upload suspicious videos through the Flutter application, which are then sent to the backend server through API requests. The backend processes the uploaded video file and forwards it to the video analysis logic or machine learning model responsible for detecting fake or manipulated video content.

©IJRASET: All Rights are Reserved

CodeSnippet:

```
from fastapi import FastAPI, File, UploadFile

app = FastAPI()

@app.post("/detect-video")
async def detect_video(file: UploadFile = File(...)):

    filename = file.filename

    # Example detection logic (placeholder)
    if "deepfake" in filename.lower() or "fake" in filename.lower():
        result = "FAKE VIDEO"
        confidence = "90%"
    else:
        result = "REAL VIDEO"
        confidence = "75%"

    return {
        "result": result,
        "confidence": confidence
    }
```

I. RESULT:

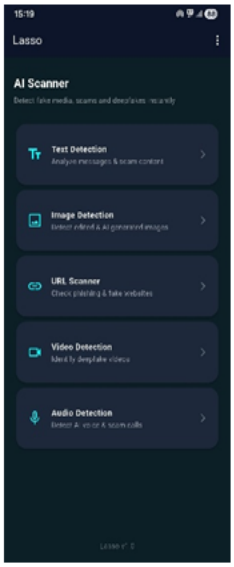
The LASSO: AI-Powered Fake Content Detection System has been successfully developed and implemented to provide a reliable and user-friendly platform for detecting fake or manipulated digital content. The application achieves its primary objective by enabling users to analyze various types of content such as text, images, audio, video, and URLs through a single mobile application. By integrating a Flutter-based frontend with a FastAPI backend and machine learning detection modules, the system efficiently processes user inputs and provides detection results along with confidence scores.

The application interface allows users to easily navigate through different detection modules from the home screen and submit suspicious content for analysis. Once the content is uploaded or entered, the system communicates with the backend server through API requests, where the detection models analyze the data and generate accurate results. The results are then displayed on the result screen, providing users with clear information about whether the content is real, suspicious, or potentially fake. Throughout testing, the application demonstrated smooth performance, quick response times, and reliable data processing.

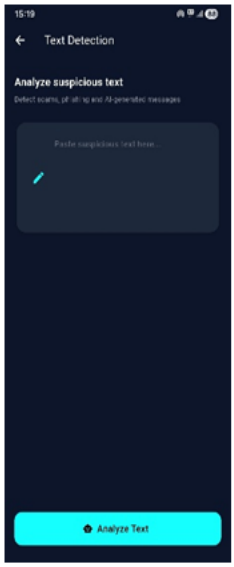
©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The integration of Flutter, FastAPI, and SQLite ensures seamless communication between the frontend interface, backend services, and data storage components. The system has been tested across multiple scenarios including text input validation, file uploads, API communication, and result generation. The application performed consistently without major errors, demonstrating stability and efficiency in handling different types of digital content analysis.

Overall, the results of this project confirm that the LASSO Fake Content Detection System successfully fulfills its intended purpose of helping users identify misleading or manipulated content in the digital environment. The system simplifies the process of verifying digital media by providing an easy-to-use mobile interface combined with intelligent detection capabilities. With its modular design and scalable architecture, the application can be further enhanced in the future by integrating advanced machine learning models, cloud storage, and real-time threat intelligence systems. Thus, the project represents an important step toward improving digital media verification and promoting safer online information consumption.



HOME PAGE



TEXT DETECTION PAGE

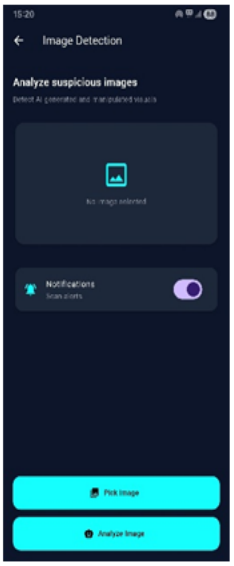
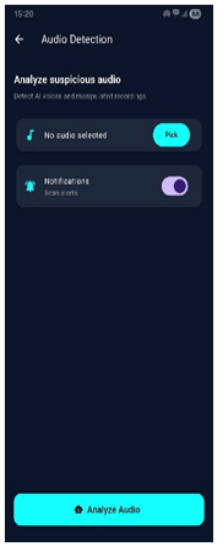


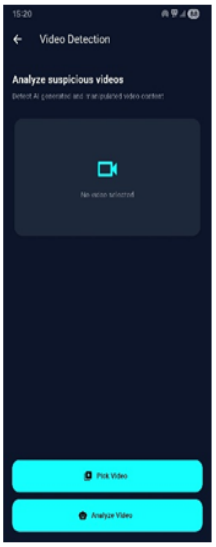
IMAGE DETECTION PAGE



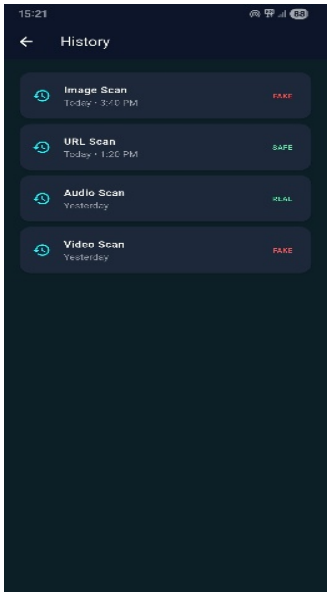
URL DETECTION PAGE



AUDIO DETECTION PAGE



VIDEO DETECTION PAGE



HISTORY PAGE

VII. CONCLUSION AND FUTURE ENHANCEMENT

A. CONCLUSION:

The LASSO: AI-Powered Fake Content Detection System successfully demonstrates the development and implementation of a modern mobile application designed to detect and analyze fake or manipulated digital content. The system provides a centralized platform where users can verify the authenticity of different types of media including text, images, audio, video, and URLs. By integrating advanced technologies such as Flutter for the mobile interface, FastAPI for backend services, and SQLite for data storage, the application delivers a reliable, efficient, and user-friendly solution for identifying suspicious digital content.

The implementation of this system highlights the growing importance of technological solutions in combating digital misinformation and fraudulent online activities. Through a simple and intuitive interface, users can easily submit suspicious content and receive analysis results along with confidence scores. Features such as multiple detection modules, result visualization, and scan history management enhance the usability of the application and allow users to verify information quickly and effectively. The integration of machine learning-based detection methods further strengthens the system's capability to identify manipulated or misleading digital content.

From a technical perspective, the project demonstrates the effective integration of mobile development frameworks, backend APIs, and machine learning processing modules within a unified architecture. The modular design of the system ensures scalability, maintainability, and efficient communication between the frontend and backend components. The use of Flutter ensures a responsive and interactive mobile interface, while FastAPI enables high-performance backend processing and seamless API communication with detection modules.

In conclusion, the LASSO Fake Content Detection System successfully achieves its primary objective of providing a smart and efficient platform for detecting fake digital content. The system improves digital awareness, enhances online safety, and helps users make informed decisions when interacting with online information. With its scalable architecture and modular design, the application can be further expanded in the future by integrating more advanced machine learning models, cloud-based processing, and real-time threat detection features. This project represents a significant step toward improving digital content verification and promoting a safer online environment.

B. FUTURE SCOPE

The LASSO: AI-Powered Fake Content Detection System has been developed with scalability and flexibility in mind, allowing it to support future enhancements as digital technologies continue to evolve. In the future, the system can be expanded by integrating advanced artificial intelligence and deep learning models to improve the accuracy of fake content detection across different media types such as text, images, audio, and videos. More sophisticated algorithms could be implemented to detect complex deepfake media and advanced phishing techniques, ensuring that the system remains effective against emerging digital threats.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Additionally, real-time detection capabilities could be introduced, enabling users to instantly verify suspicious content while browsing the internet or using messaging platforms.

Another potential improvement for the system is the integration of cloud-based services and large-scale data processing frameworks. By using cloud infrastructure, the application could process larger volumes of data and provide faster analysis results. Cloud integration would also enable centralized storage of detection results, allowing users to access their scan history across multiple devices. Furthermore, the system could incorporate real-time threat intelligence databases, which would continuously update the detection engine with information about newly discovered malicious websites, fake media patterns, and cyber threats.

Future versions of the application can also focus on expanding platform accessibility. While the current system is implemented as a mobile application, future development could include web-based and cross-platform versions, allowing users to access the system through desktop browsers as well as mobile devices. Browser extensions could also be developed to enable users to analyze suspicious URLs, images, or text directly while browsing the internet. These enhancements would significantly increase the usability and reach of the system.

From a security perspective, additional measures such as multi-factor authentication, encrypted data transmission, and advanced user verification systems can be implemented to enhance user privacy and system security. Integration with popular social media platforms and messaging applications could also allow users to quickly analyze suspicious links or media shared through these platforms. Moreover, the system could incorporate data analytics and reporting dashboards that provide insights into detection trends, types of threats identified, and user activity statistics.

In conclusion, the LASSO Fake Content Detection System has strong potential for future expansion and technological advancement. By integrating advanced machine learning models, cloud computing, improved security mechanisms, and cross-platform accessibility, the system can evolve into a comprehensive digital security solution. These future improvements will enhance the system's ability to combat misinformation, detect manipulated media, and protect users from online threats, contributing to a safer and more trustworthy digital environment.

VIII. ACKNOWLEDGEMENT

Itisoneofthemostefficienttasksinlifetochoosetheappropriate wordstoexpress one'sgratitude tothebeneficiaries. Weareverymuchgratefulto Godwhohelpedusallthewaythroughtheprojectandhow molded us into what we are today.

WearegratefultoourbelovedPrincipalDr.R.RADHAKRISHNAN,M.E.,Ph.D.,AdhiyamaanCollege of Engineering (An Autonomous Institution), Hosur for providing the opportunity to dothis work in premises.

We acknowledge our heartful gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

We are highly indebted to Mrs. R. VIJAYALAKSHMI, M.E., Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering(An Autonomous Institution), Hosur, whose immense support encouragement and valuable guidance were responsible to complete the project successfully.

WealsoextentourthankstoProjectCoordinatorandallStaffMembersfortheirsupportincomplete this project successfully.

Finally,wewouldliketothanktoourparents,withouttheirmotivationalandsupportwouldnot have been possible for us to complete this project successfully.

IX. APPENDICES

SOURCECODE

Authentication:

```
from fastapi import FastAPI, UploadFile, File
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from PIL import Image
import io
import numpy as np
import librosa
import os
import shutil
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
import cv2
import json
from dotenv import load_dotenv
from openai import OpenAI
import base64
import re
from fastapi import Body

# ==========================
# LOAD ENV + GROQ
# ==========================
load_dotenv()
NVIDIA_API_KEY = os.getenv("NVIDIA_API_KEY")
nim_client = OpenAI(
base_url="https://integrate.api.nvidia.com/v1",
api_key=NVIDIA_API_KEY
)

# ==========================
# CREATE APP
# ==========================
app = FastAPI(title="Fake Content Detection API (Groq Powered)")

app.add_middleware(
CORSMiddleware,
allow_origins=["*"],
allow_credentials=True,
allow_methods=["*"],
allow_headers=["*"],
)

# ==========================
# REQUEST MODEL
# ==========================
class TextRequest(BaseModel):
    text: str

# ==========================
# HOME + STATUS
# ==========================
@app.get("/")
def home():
    return {"message": "Groq Fake Detection API Running"}

@app.get("/status")
def status_check():
    return {"status": "ok"}

# -------------------------
# TEXT DETECTION (GROQ)
```

©IJRASET: All Rights are Reserved

```
# -------------------------
@app.post("/detect-text")
def detect_text(request: TextRequest):
    try:
        prompt = f"""
Detect if this text is REAL or FAKE or SCAM.
Return ONLY JSON:
{{"result":"FAKE","confidence":"90%"}}

Text:
{request.text}
"""

        completion = nim_client.chat.completions.create(
            model="meta/llama3-70b-instruct",
            messages=[{"role": "user", "content": prompt}],
            temperature=0
        )

        raw = completion.choices[0].message.content

        import re, json
        match = re.search(r"\{.*\}", raw, re.DOTALL)
        if match:
            return json.loads(match.group())

        return {"result": "ERROR", "confidence": "0%"}

    except Exception as e:
print("NVIDIA TEXT ERROR:", e)
        return {"result": "ERROR", "confidence": "0%"}


# -------------------------
# URL DETECTION (NVIDIA)
# -------------------------
@app.post("/detect-url")
def detect_url(data: dict = Body(...)):
    try:
url = data.get("url")

        prompt = f"""
You are a cybersecurity system.

Analyze this URL and classify it as:
SAFE / PHISHING / SCAM / SUSPICIOUS

Check for:
- fake login pages
- bank phishing
```

```
- lottery or prize scams
- shortened or masked links
- suspicious domains
- misleading URLs

Respond ONLY in JSON:
{{"result":"SAFE","confidence":"90%"}}

URL:
{url}
"""

        completion = nim_client.chat.completions.create(
            model="meta/llama-3.1-8b-instruct",
            messages=[{"role": "user", "content": prompt}],
            temperature=0,
max_tokens=200
        )

        raw = completion.choices[0].message.content
print("NVIDIA URL RAW:", raw)

        import re, json

        raw = raw.replace("json", "").replace("", "").strip()

        match = re.search(r"\{.*\}", raw, re.DOTALL)

        if match:
            try:
                parsed = json.loads(match.group())

                #  IMPORTANT: send only these 2 fields to Flutter
                return {
                    "result": parsed.get("result", "SAFE"),
                    "confidence": parsed.get("confidence", "75%")
                }

            except Exception as e:
print("URL JSON PARSE ERROR:", e)

        # fallback
        lower = raw.lower()

        if any(w in lower for w in ["phishing", "scam", "malicious", "fake"]):
            return {"result": "PHISHING", "confidence": "85%"}

        return {"result": "SAFE", "confidence": "75%"}
```

```
    except Exception as e:
print("NVIDIA URL ERROR:", e)
    return {"result": "ERROR", "confidence": "0%"}

# -------------------------
# IMAGE DETECTION (NVIDIA)
# -------------------------
@app.post("/detect-image")
async def detect_image(file: UploadFile = File(...)):
    try:
        content = await file.read()
        image_base64 = base64.b64encode(content).decode("utf-8")

        completion = nim_client.chat.completions.create(
            model="meta/llama-3.2-11b-vision-instruct",
            messages=[
                {
                    "role": "user",
                    "content": [
                        {
"type": "text",
"text": """
You are an AI image forensic detector.

Your job is to detect AI-generated images, deepfakes, and manipulated visuals.

Mark image as FAKE if ANY of these are present:
- AI generated artwork or portraits
- Unreal skin texture or symmetry
- Distorted fingers, eyes, teeth
- Over-smooth lighting
- Text artifacts
- Inconsistent shadows
- Background blending errors
- Synthetic or diffusion-style visuals

Mark REAL only if it is a natural camera photograph.

Respond ONLY in JSON:
{"result":"FAKE","confidence":"0-100%"}
"""
},
                        {
                            "type": "image_url",
                            "image_url": {
                                "url": f"{image_base64}"
                            }
                        }
                    ]
                }
```

```
            ],
max_tokens=200,
            temperature=0
        )

        raw = completion.choices[0].message.content
print("NVIDIA IMAGE RAW:", raw)

        # 🔧 Fix invalid % formatting automatically
        raw = re.sub(r'"confidence":\s*(\d+)%', r'"confidence":"\1%"', raw)

        # Extract JSON safely
        match = re.search(r"\{.*\}", raw, re.DOTALL)

        if match:
            try:
                return json.loads(match.group())
            except Exception as e:
print("JSON PARSE ERROR:", e)

                # ADD THIS HERE (AI-image detection fallback)
lower_raw = raw.lower()

                if any(word in lower_raw for word in [
                    "ai generated",
                    "synthetic",
                    "diffusion",
                    "unreal",
                    "render",
                    "not a real photo",
                    "computer generated"
                ]):
                    return {"result": "FAKE", "confidence": "90%"}

                # Final fallback
            return {"result": "REAL", "confidence": "75%"}
        #  Fallback logic if JSON still invalid
lower_raw = raw.lower()

        if any(word in lower_raw for word in ["scam", "fake", "manipulated", "deepfake", "ai generated"]):
            return {"result": "FAKE", "confidence": "85%"}

        return {"result": "REAL", "confidence": "75%"}

    except Exception as e:
print("NVIDIA IMAGE ERROR:", e)
        return {"result": "ERROR", "confidence": "0%"}
```

```
# -------------------------
# AUDIO DETECTION (NVIDIA)
# -------------------------
@app.post("/detect-audio")
async def detect_audio(file: UploadFile = File(...)):
    try:
audio_bytes = await file.read()
        audio_base64 = base64.b64encode(audio_bytes).decode("utf-8")

        completion = nim_client.chat.completions.create(
            model="meta/llama-3.1-8b-instruct",
            messages=[
                {
                    "role": "user",
                    "content": f"""
You are an audio forensic AI.

If audio is:
- AI generated
- voice cloned
- synthetic
- scam call
- impersonation

Result must be FAKE.
If natural human speech → REAL.

Reply ONLY JSON:
{{"result":"FAKE or REAL","confidence":"0-100%"}}
Do not explain.
"""

                },
            ],
            temperature=0,
max_tokens=200
        )

        raw = completion.choices[0].message.content
print("AUDIO RAW:", raw)

        import re, json
        match = re.search(r"\{.*\}", raw, re.DOTALL)

        if match:
            return json.loads(match.group())

lower_raw = raw.lower()
```

```
ai_audio_keywords = [
            "ai generated",
            "synthetic",
            "text to speech",
            "tts",
            "computer generated",
            "voice cloning",
            "deepfake voice",
            "digitally altered",
            "unnatural pitch",
            "robotic",
            "uniform tone",
            "lack of breathing",
            "processed audio"
        ]

        if any(word in lower_raw for word in ai_audio_keywords):
            return {"result": "FAKE", "confidence": "90%"}

scam_keywords = [
            "otp",
            "bank account",
            "urgent payment",
            "verify account",
            "credit card",
            "lottery",
            "prize",
            "click link",
            "send money"
        ]

        # AI voice signals
        if any(word in lower_raw for word in ai_audio_keywords):
            return {"result": "FAKE", "confidence": "90%"}

        # scam content signals
        if any(word in lower_raw for word in scam_keywords):
            return {"result": "FAKE", "confidence": "88%"}

        # if model unsure
        return {"result": "UNCERTAIN", "confidence": "60%"}


    except Exception as e:
print("AUDIO ERROR:", e)
        return {"result": "ERROR", "confidence": "0%"}

# -------------------------
# VIDEO DETECTION (NVIDIA)
```

```
# -------------------------
@app.post("/detect-video")
async def detect_video(file: UploadFile = File(...)):
    try:
        # save temp video
video_bytes = await file.read()
        with open("temp_video.mp4", "wb") as f:
f.write(video_bytes)

        cap = cv2.VideoCapture("temp_video.mp4")

frames_checked = 0
fake_hits = 0

        while cap.isOpened() and frames_checked< 8:
            ret, frame = cap.read()
            if not ret:
                break

            _, buffer = cv2.imencode(".jpg", frame)
            image_base64 = base64.b64encode(buffer).decode("utf-8")

            completion = nim_client.chat.completions.create(
                model="meta/llama-3.2-11b-vision-instruct",
                messages=[
                    {
                        "role": "user",
                        "content": [
                            {
                                "type": "text",
                                "text": """
You are a digital forensic AI.

If the video frame is:
- AI generated
- deepfake
- face swapped
- synthetic
- edited
- manipulated

Then result MUST be FAKE.

If it is natural camera footage, result must be REAL.

Reply ONLY JSON:
{"result":"FAKE or REAL","confidence":"0-100%"}
Do not explain.
"""
                            },
```

```
                {
                  "type": "image_url",
                  "image_url": {
                    "url": f"{image_base64}"
                  }
                }
              ]
            }
          ],
          temperature=0,
max_tokens=150
        )

        raw = completion.choices[0].message.content.lower()

        # AI / synthetic keywords
ai_keywords = [
            "ai generated",
            "synthetic",
            "deepfake",
            "computer generated",
            "rendered",
            "diffusion",
            "not a real video",
            "unnatural",
            "inconsistent lighting",
            "face manipulation",
            "fake"
        ]

cap.release()

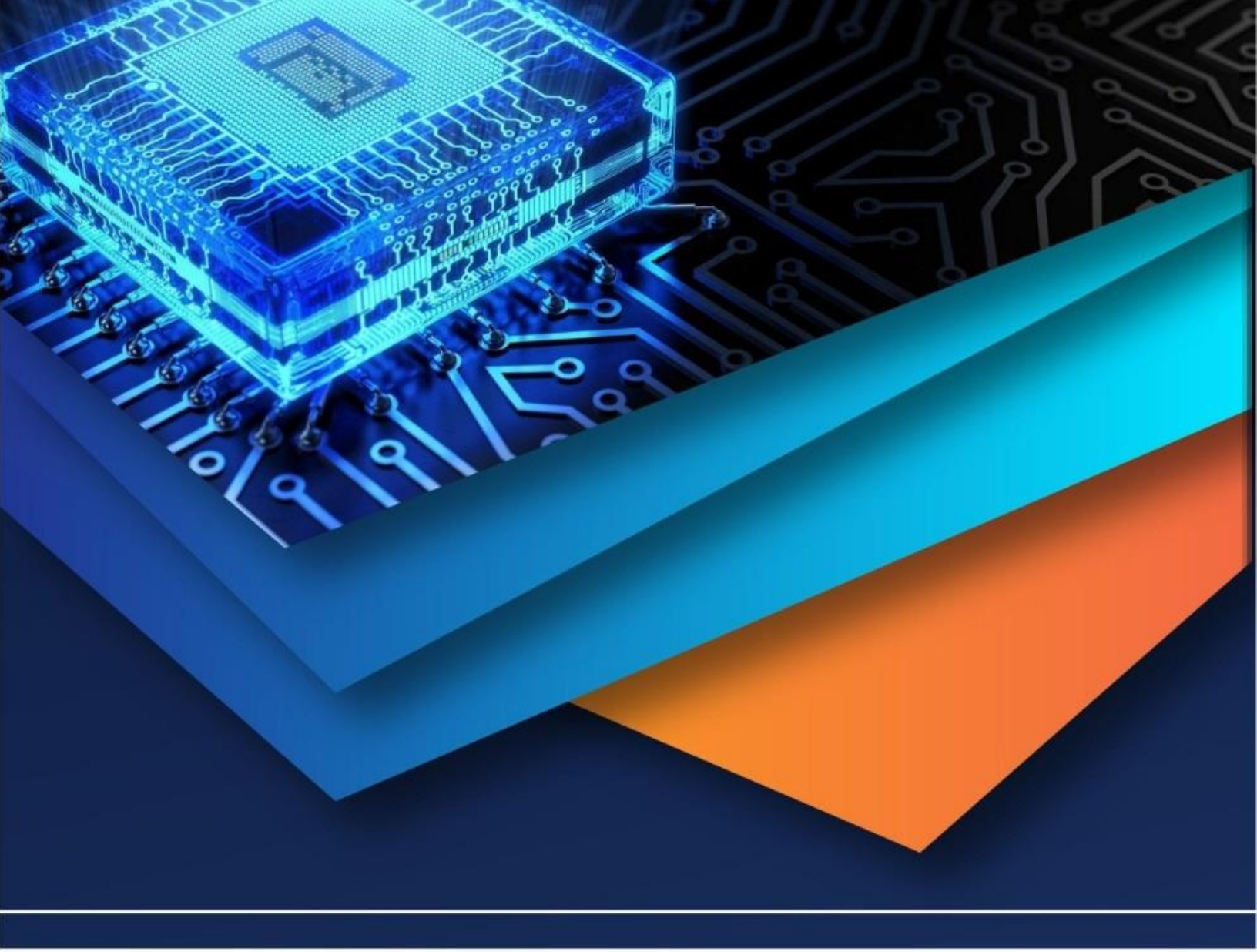
      if fake_hits>= 2:
        return {"result": "FAKE", "confidence": "88%"}
      else:
        return {"result": "REAL", "confidence": "78%"}

   except Exception as e:
print("VIDEO ERROR:", e)
      return {"result": "ERROR", "confidence": "0%"}
```

REFERENCES

[1] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media, 2019.
[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.
[3] Mark Lutz, Learning Python, 5th Edition, O'Reilly Media, 2013.
[4] Eric Matthes, Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2nd Edition, No Starch Press, 2019.
[5] Flutter Documentation, Flutter Development Framework, Available at: https://flutter.dev/docs(Accessed: October 2025).
[6] FastAPI Official Documentation, FastAPI Framework for Building APIs, Available at: https://fastapi.tiangolo.com/(Accessed: October 2025).
[7] SQLite Documentation, SQLite Database Engine, Available at: https://www.sqlite.org/docs.html(Accessed: October 2025).
[8] Django Official Documentation, Django Web Framework, Available at: https://docs.djangoproject.com/(Accessed: October 2025).

[9] W3Schools, Web Development Tutorials (HTML, CSS, JavaScript), Available at: https://www.w3schools.com/(Accessed: October 2025).
[10] GeeksforGeeks, Machine Learning and Web Development Tutorials, Available at: https://www.geeksforgeeks.org/(Accessed: October 2025).
[11] TutorialsPoint, Django Framework Tutorial, Available at: https://www.tutorialspoint.com/django/(Accessed: October 2025).
[12] Stack Overflow, Programming Solutions and Developer Community, Available at: https://stackoverflow.com/(Accessed: October 2025).
[13] Research Paper: S. Agarwal and T. Varshney, "Detection of Fake News Using Machine Learning Techniques," International Journal of Computer Applications, Vol. 183, No. 15, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)