Faculty of Computers & Artificial Intelligence

Benha University

Catch The AI

A senior project submitted in partial fulfilment of the requirements for the degree of Bachelor of Computers and Artificial Intelligence.

Artificial Intelligence Department,

Project Team

1- Romani Nasrat Shawqi
2- Ahmed Mohamed Ali
3- Zeyad Elsayed Abdel-Azim
4- Sara Reda Moatamed
5- Reham Moustafa Ali
6- Rawan Abdel-Aziz Ahmed
7- Abdallah Mohammed Abdel-monnem
8- Mohannad Ayman salah
9- Mohammed Abdallah Abdel-salam

Under Supervision of

Dr. Eman Abdel-Latef

Eng. Sahar Mohamed

Benha, JUNE 2024

DEDICATION

This work is dedicated to our families, whose unwavering support and encouragement have been our guiding light throughout this journey. To our parents, who instilled in us the values of perseverance and hard work, and to our siblings and friends, whose constant belief in us has been a source of inspiration. Your love and sacrifices have made this accomplishment possible.

ACKNOWLEDGMENT

We, the team of nine AI engineers, would like to extend our heartfelt gratitude to everyone who has contributed to the completion of this work.

Firstly, we are deeply grateful to our supervisor, Prof. Eman Abdel-Latef, for her invaluable guidance, insightful feedback, and continuous support. Your expertise and encouragement have been instrumental in shaping this work.

We also wish to thank our industry partners and sponsors, whose financial support and practical insights have greatly enhanced the quality of this research. Special thanks to the faculty members of the Department of Computers and Artificial Intelligence, whose knowledge and dedication have profoundly impacted our academic journey.

To our friends and colleagues, thank you for your camaraderie, motivation, and the countless discussions that enriched our understanding. Lastly, our deepest appreciation goes to our families for their unwavering support and love throughout this process. This achievement would not have been possible without you.

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in Artificial Intelligence Engineering, is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not, to the best of our knowledge, breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Date: Monday, 24 June 2024

ABSTRACT

The Growing Challenge of AI-Generated Media: A Multimodal Detection System for Enhanced Trust

The rapid evolution of deep learning, particularly Generative Adversarial Networks (GANs) and Variational Auto-encoders (VAEs), has yielded remarkably realistic AI-generated media content. While advantageous for creative fields like film and advertising, this technology presents a growing challenge for Multimedia Information Process and Retrieval (MIPR) systems, facial recognition, and speech recognition. The potential for manipulating these systems with AI-generated content raises concerns about the spread of misinformation and the erosion of trust in online interactions.

This project tackles this challenge by proposing a comprehensive model for detecting the origin of multimedia content, encompassing text, images, and audio. Utilizing advanced algorithms, the system identifies subtle manipulations indicative of AI generation. Three primary detection models form the core of this system:

- **AI Image Generation Detection:** Accurately identifies images created by artificial intelligence.
- **AI Text Generation Detection:** Recognizes text authored by AI algorithms and distinguishes it from human-generated content.
- **AI-Generated Audio Detection:** Identifies and differentiates between audio content generated by artificial intelligence and authentic human-recorded audio.

By prioritizing content authenticity across all these modalities, this project aims to safeguard against the potential pitfalls of AI-generated media. It promotes a digital environment characterized by transparency, trust, and ethical AI use, ultimately fostering a more reliable and trustworthy online ecosystem.

TABLE OF CONTENTS

Table of Contents

Table of Contents

2 LIST OF TABLES

3 LIST OF FIGURES

3 List of Figures

4 LIST OF ACRONYMS/ABBREVIATIONS

ACRONYMS	Definition
AI	**Artificial Intelligence**: A field of computer science focused on creating systems capable of performing tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. (Chapter 1: Introduction)
LLMs	**Large Language Models**: Deep learning models that are trained on vast amounts of text data to understand and generate human-like language, often used for tasks such as translation, summarization, and question answering. (Chapter 1: Introduction)
GANs	**Generative Adversarial Networks**: A class of neural networks used in unsupervised learning, where two networks (generator and discriminator) compete against each other to create data that is indistinguishable from real data. (Chapter 1: Introduction)
UML	**Unified Modelling Language:** A standardized modelling language in software engineering used to specify, visualize, construct, and document the artifacts of a system. (Chapter 2: System Design and Analysis)
GPT	**Generative Pre-trained Transformer:** A type of large language model developed by OpenAI that uses deep learning to produce human-like text. (Chapter 1: Overview)
RoBERTa	**Robustly optimized BERT approach:** A derivative of BERT (Bidirectional Encoder Representations from Transformers) with improved training strategies, achieving better performance on NLP tasks. (Chapter 2: Literature Review & Chapter 4: Methodology)

BERT	**Bidirectional Encoder Representations from Transformers**: A transformer-based model designed to understand the context of a word in search queries. (Chapter 2: Literature Review & Chapter 4: Methodology)
DeBERTa	**Decoding-enhanced BERT with disentangled attention:** An improvement over BERT that introduces disentangled attention and enhanced decoding for better language understanding. (Chapter 2: Literature Review & Chapter 4: Methodology)
MFCCs	**Mel-Frequency Cepstral Coefficients:** Features used in automatic speech and audio processing that represent the short-term power spectrum of a sound. (Chapter 2: Literature Review & Chapter 4: Methodology)
API	**Application Programming Interface:** A set of protocols and tools for building and interacting with software applications, allowing different systems to communicate. (Chapter 5: Implementation)
RESTful	**Representational State Transfer (REST):** An architectural style for designing networked applications, relying on stateless communication and standard HTTP methods. (Chapter 3: System Design & Chapter 4: Methodology)
MVC	**Model-View-Controller:** A design pattern for developing user interfaces that separates an application into three interconnected components: Model, View, and Controller. (Chapter 3: Design and Implementation)

UI	**User Interface:** The space where interactions between humans and machines occur, including design elements such as buttons, menus, and forms. (Chapter 2: System Design)
ASVspoof	**Automatic Speaker Verification Spoofing:** A challenge or technology focused on detecting spoofing attacks in speaker verification systems to ensure the authenticity of the speaker's identity. (Chapter 4: Data Acquisition)
SFHQ	**Single-Frequency Harmonic Quantization:** A technique in signal processing for analysing and representing periodic signals. (Chapter 4: Signal Processing)
CNN	**Convolutional Neural Network:** A class of deep neural networks primarily used for analysing visual data, known for its ability to automatically and adaptively learn spatial hierarchies of features. (Chapter 2: Existing Techniques & Chapter 4: Methodology)
ROC	**Receiver Operating Characteristic:** A graphical plot used to illustrate the diagnostic ability of a binary classifier system, showing the trade-off between sensitivity and specificity. (Chapter 4: Evaluation Metrics)

AUC	**Area Under the Curve:** A performance measurement for classification models, particularly in ROC analysis, representing the degree of separability achieved by the model. (Chapter 4: Evaluation Metrics)
JSX	**JavaScript XML:** It's a syntax extension for JavaScript that allows HTML to be written inside JavaScript code.
DOM	**Document Object Model:** Represents the structure of HTML documents and provides a way to interact with and manipulate them in a platform-independent manner.
CSS	**Cascading Style Sheets:** Used to style HTML elements in React applications.
SPA	**Single Page Application:** A type of web application that loads a single HTML page and dynamically updates the content as the user interacts with the application.
HTTP	**Hypertext Transfer Protocol:** Managed by libraries (axios, use-http) to handle network requests and data fetching within the React application.
NPM	**Node Package Manager:** Used for managing project dependencies (NPM) and executing scripts (NPM start, NPM build, NPM test) defined in package. Json.

CLI	**Command-Line Interface:** Tools like react-scripts provide a CLI for running development servers (NPM start), building production bundles (NPM build), and other tasks (NPM eject).
CI/CD	**Continuous Integration/Continuous Deployment:** Practices facilitated by tools like GitHub Actions, Travis CI, or GitLab CI to automate testing (NPM test) and deployment (NPM build) processes.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

The realm of artificial intelligence (AI) is experiencing rapid growth [1], [2] with tech giants vying for dominance in groundbreaking technologies like transformers, Generative Adversarial Networks (GANs), and Large Language Models (LLMs). However, alongside these advancements arises a critical challenge: the increasing difficulty in differentiating AI-generated content from authentic human-created material. This challenge extends to images, text, and even audio, raising concerns about the spread of misinformation and the erosion of trust in online interactions. Our project proposes a comprehensive online platform equipped with specialized detection models to address this issue. By leveraging machine learning techniques, we aim to empower users to discern between AI-generated and human-created content, fostering transparency and trust in the digital landscape.

1.2 PROBLEM STATEMENT

The rapid evolution of AI, particularly with advanced techniques like GANs [3] and transformers, has enabled the creation of remarkably lifelike and convincing AI-generated images, text, and audio. This progress presents a significant challenge: the increasing difficulty in distinguishing AI-generated content from real content. This situation opens doors for malicious actors to create and disseminate deceptive information, like the challenges posed by "fake news." Large Language Models (LLMs) exacerbate this problem by generating human-quality text [4], posing unique challenges in educational settings. As open-source AI technologies advance, the line between machine-generated and human-created media blurs, paving the way for potential misuse [5]. Robust systems to identify content origin become paramount to protect trust and transparency in online interactions.

The potential consequences of undetected AI-generated media are vast, including the spread of misinformation, erosion of trust, academic dishonesty, and cyberbullying.

1.3 PROJECT OBJECTIVE

The objective of this project is to develop a comprehensive detection system that identifies and differentiates between human-generated and AI-generated multimedia content. By leveraging advanced algorithms and deep learning techniques, this system aims to safeguard against the proliferation of deceptive and manipulated media. [6]The project focuses on three primary detection models:

- **AI Image Generation Detection**: Accurately identify images generated by artificial intelligence and distinguish them from human-created images.

- **AI Text Generation Detection**: Recognize text generated by AI and determine its authorship [7].
- **AI-Generated Audio Detection**: Identify and differentiate between audio content generated by artificial intelligence and authentic human-recorded audio [8].

By addressing these challenges, the project seeks to enhance the reliability of multimedia information, protect facial and speech recognition systems, and promote a trustworthy digital ecosystem through transparency and ethical AI use.

1.4 SCOPE AND REQUIREMENTS

1.4.1 key Features:

- AI-Generated Image Detection: Clearly identify and differentiate between AI-created and human-created images, safeguarding against the misuse of visuals.
- AI-Generated Text Detection: Recognize text generated by AI and distinguish it from human-written content, ensuring the authenticity of information.
- AI-Generated Audio Detection: Identify and differentiate between audio content generated by AI and authentic human-recorded audio [9].

1.4.2 Core Requirements

- Machine Learning Model Development: Building robust machine learning models for accurate detection across all modalities [10].
- Data Collection and Preprocessing: A substantial and well-categorized dataset for each media type (images, text, and audio) is crucial for training effective models [11].

1.4.3 Scope Exclusions

- **Privacy and Security Measures:** While important, these are not the focus of this project but should be considered during development.
- **Technical Challenges:** Acknowledging the exclusion of addressing all technical challenges, especially advanced, hidden AI techniques.

1.5 TARGET AUDIENCE

The target audience for this project includes:

- **Academics and Researchers:** Individuals conducting research in AI media detection.
- **Software Developers:** Professionals interested in implementing AI-driven solutions.

- **Students:** Those studying computer science, AI, and related fields.
- **Media Analysts:** Experts needing tools to identify AI-generated content.
- **Law Enforcement Agencies:** Utilizing AI-generated audio detection for investigating deepfakes.[12]

1.6 METHODOLOGY

The project follows a structured methodology:

1. **Requirement Analysis:** Identify and document requirements, engage with stakeholders.
2. **System Design:** Design architecture, create UML diagrams.
3. **Data Collection and Preprocessing:** Gather and preprocess datasets.
4. **Model Development:** Utilize pretrained models for training and fine-tuning.
5. **Implementation:** Develop frontend with React, backend with Django.
6. **Integration and Testing:** Perform thorough testing.
7. **Deployment:** Deploy using Docker, host on cloud platforms [13].
8. **Documentation:** Document development process, prepare user manuals.

1.7 DOCUMENT STRUCTURE

The document is structured as follows: [14]

1. **Title Page**: Project title, author, institution, date.
2. **Abstract**: Summary of objectives, methods, outcomes.
3. **Table of Contents**: Sections and subsections with page numbers.
4. **Introduction**: Background, motivation, problem statement, objectives, scope.
5. **Literature Review**: Existing research and techniques.
6. **Project Methodology**: Project plan, technologies, system design, data handling, model development.
7. **Implementation**: Frontend and backend development, integration, deployment strategies.
8. **Testing and Results**: Testing methodologies, performance analysis, results.
9. **User Manual**: Installation instructions, system requirements, usage guide.
10. **Conclusion and Future Work**: Findings, limitations, future enhancements.
11. **References**: Academic papers, books, other sources.
12. **Appendices**: Additional materials such as code snippets, data samples, detailed diagrams.

2. LITERATURE REVIEW

2.1 INTRODUCTION

The proliferation of AI-generated media, encompassing audio, text, and images, presents both remarkable opportunities and significant challenges. AI technologies, particularly those driven by advanced machine learning models like Generative Adversarial Networks (GANs) and deep neural networks, have reached a level of sophistication where synthetic content is often indistinguishable from real. This rapid advancement necessitates robust detection mechanisms to maintain the integrity and authenticity of digital content. This chapter explores the current techniques employed to detect AI-generated media, detailing their methodologies, applications, tools, strengths, and limitations. Through this exploration, we aim to provide a comprehensive understanding of how these techniques function individually and how they contribute collectively to the field of media forensics.

2.2 EXISTING TECHNIQUES

In recent years, the proliferation of AI-generated media has necessitated the development of robust detection techniques. Various methods have been proposed and implemented across different modalities—text [15], audio, and images. Each modality presents unique challenges and has prompted the development of specialized techniques.

2.2.1 Text Detection Techniques

AI-generated text [16] detection has evolved significantly with the advent of sophisticated language models like GPT-3 and beyond. Early approaches relied on statistical methods and linguistic features such as word frequency and sentence structure. However, these methods were often inadequate against advanced AI-generated text. Current state-of-the-art techniques leverage deep learning models, particularly those based on transformers.

2.2.1.1 Stylometry

- Method: Analyses text for stylistic features such as syntax and grammar.
- Application: Detects AI-generated text based on stylistic deviations.
- Tools: Linguistic models, frequency analysis.

- **Strengths**:
  - Captures subtle stylistic differences.
  - Proven technique for authorship attribution.
- **Limitations**:
  - Struggles against AI mimicking human styles.
  - May misclassify atypical legitimate text styles.

2.2.1.2 Contextual Analysis

• Method: Examines text coherence, context, and relevance.

• Application: Identifies unnatural topic shifts and logical inconsistencies.

• Tools: NLP algorithms, context-aware models.

- **Strengths**:
  - Effective in identifying logical inconsistencies.
  - Broadly applicable to different text types.
- **Limitations**:
  - Subjective nature can lead to false positives.
  - Computationally intensive for complex contexts.

2.2.1.3 Entropy and Perplexity Measure

• Method: Measures text randomness or predictability.

• Application: Detects higher entropy and perplexity in AI-generated text.

• Tools: Statistical models, entropy calculators.

- **Strengths**:
  - Quantitative metrics for predictability.
  - Simple to implement.
- **Limitations**:
  - AI can be trained to minimize these metrics.
  - Requires a reference baseline for accuracy.

2.2.1.4 Machine Learning Classifiers

• Method: Uses models trained to distinguish human and AI-generated text.

• Application: Automated detection of synthetic text.

• Tools: SVMs, Transformer models.

- **Strengths**:
  - Capable of real-time analysis.
  - Learns and adapts with more data.

- **Limitations**:
    - Can produce false positives/negatives.
    - Resource-intensive training and operation.

2.2.2 Audio

Detection of AI-generated audio, such as synthesized speech or deepfake voices, has also seen substantial advancements. Traditional methods focused on spectral analysis and acoustic features. With the rise of powerful speech synthesis models like Wav2Vec2[17], more sophisticated techniques are required. Modern approaches include:

2.2.2.1 Spectral Analysis

- **Method**: Spectral analysis decomposes audio signals into their constituent frequencies to reveal anomalies.
- **Application**: Identifies irregularities in frequency patterns that are indicative of synthetic audio.
- **Tools**: Spectrograms, Fourier transforms.
    - **Strengths**:
        - Effective at spotting subtle spectral artifacts.
        - High sensitivity to minor deviations from natural patterns.
    - **Limitations**:
        - Vulnerable to advanced AI techniques mimicking natural spectra.
        - Less effective with low-quality recordings.

2.2.2.2 Voice Characteristics Analysis

- Method: Analyses voice features such as pitch, tone, and rhythm.
- Application: Detects unnatural variations in speech characteristics.
- Tools: Voiceprint analysis, prosody analysis.
    - **Strengths:**
        - Captures human-like voice features.
        - Sensitive to contextual speech anomalies.
    - **Limitations:**
        - Natural voice variability can cause false positives.
        - Limited effectiveness against highly advanced synthetic voices.

2.2.2.3 Acoustic Environment Detection

- **Method**: Evaluates environmental sounds like background noise and reverberation.
- **Application**: Identifies inconsistencies in the recording environment.
- **Tools**: Acoustic scene analysis, noise profiling.
    - Strengths:
        - Detects environmental inconsistencies that are hard to fake.
        - Useful for situational context analysis.

- Limitations:
  - Less effective in controlled environments.
  - Genuine poor-quality recordings may be misclassified.

2.2.2.4 Machine Learning Models

- **Method**: Trains models to differentiate between natural and synthetic audio.
- **Application**: Automates the detection of AI-generated audio.
- **Tools**: CNNs, RNNs.

  - Strengths:
    - Scalable and capable of handling large datasets.
    - Adaptive learning improves detection over time.
  - Limitations:
    - Relies on extensive and varied training data.
    - Needs regular updates to handle new AI techniques.

2.2.3 Images

The detection of AI-generated images, such as those created by GANs (Generative Adversarial Networks), involves analysing various visual features. Traditional image analysis techniques are often insufficient against high-quality GAN-generated images.

2.2.3.1 Forensic Analysis

- **Method**: Looks for inconsistencies in lighting, shadows, and reflections.
- **Application**: Detects image manipulation and AI-generated content.
- **Tools**: Error Level Analysis (ELA), chromatic aberration analysis.

  - **Strengths**:
    - Effective at detecting visual inconsistencies.
    - Widely applicable to various manipulations.
  - **Limitations**:
    - Advanced techniques may bypass detection.
    - Requires expert interpretation for subtle anomalies.

2.2.3.2 Pixel-Level Anomalies

- **Method**: Examines pixel irregularities.
- **Application**: Identifies artifacts in AI-generated images.
- **Tools**: Noise pattern analysis, JPEG compression artifacts.
  - **Strengths**:
    - Detailed pixel-level analysis.
    - Automatable for large datasets.
  - **Limitations**:
    - Struggles with high-fidelity AI-generated images.
    - Computationally expensive.

2.2.3.3 GAN Fingerprints

- **Method**: Analyses patterns left by GANs.
- **Application**: Detects AI-generated images with GAN-specific artifacts.
- **Tools**: Feature extraction algorithms, GAN-specific detection models.
    - **Strengths**:
        - Recognizes unique GAN patterns.
        - Effective for detecting GAN-generated images.
    - **Limitations**:
        - Newer GANs may hide these patterns.
        - Not applicable to non-GAN images.

2.2.3.4 Deep Learning Models

- **Method**: Trains models to distinguish real from AI-generated images.
- **Application**: Automates image detection.
- **Tools**: CNNs, ResNet, EfficientNet.
    - **Strengths**:
        - Scalable for large-scale analysis.
        - Adaptive and improves with data.
    - **Limitations**:
        - Requires diverse training data.
        - Vulnerable to adversarial attacks.

2.3 CROSS-MEDIA TECHNIQUES

2.3.1 Metadata Analysis

- **Method**: Inspects metadata for inconsistencies.
- **Application**: Identifies anomalies in file creation and modification details.
- **Tools**: Metadata readers, forensic tools.
    - **Strengths**:
        - Quick way to identify potential anomalies.
        - Complementary to other techniques.
    - **Limitations**:
        - Metadata can be easily manipulated.
        - Provides limited authenticity insight.

4.2 Consistency Checks

- **Method**: Evaluates alignment across different media types.
- **Application**: Detects discrepancies between text, images, and audio.
- **Tools**: Multimodal analysis models.
    - **Strengths**:
        - Holistic approach to media consistency.
        - Effective for verifying cross-media alignment.
    - **Limitations**:

- Requires sophisticated models for effective analysis.
- Natural inconsistencies may lead to false positives.

2.4 STRENGTHS AND LIMITATIONS [18]

2.4.1 Strengths

- **Accuracy**: Modern deep learning models, particularly those based on transformers, have significantly improved the accuracy of AI-generated media detection.
- **Scalability**: These models can be scaled to handle large datasets, making them suitable for real-world applications.
- **Versatility**: Techniques developed for one modality (e.g., text) can often be adapted for another (e.g., audio), leveraging common underlying principles.

2.4.2 Limitations

- **Evolving Nature of AI:** As AI models improve, detection techniques must continually adapt. This ongoing "arms race" presents a significant challenge [19].
- **Resource Intensive:** Training and deploying sophisticated models require substantial computational resources.
- **False Positives/Negatives:** No method is perfect; there are always trade-offs between sensitivity and specificity, leading to potential misclassification.

2.5 RESEARCH GAP

While substantial progress has been made, several research gaps remain:

- **Multimodal Detection**: Most existing techniques focus on a single modality. There is a need for integrated approaches that can simultaneously handle text, audio, and images [20].
- **Real-time Detection:** Developing models that can operate in real-time, especially for audio and video streams, remains a significant challenge [21].
- **Generalizability:** Ensuring that models generalize well across different datasets and contexts is critical. Current models often perform well on specific datasets but struggle with out-of-domain samples.
- **Explainability:** Providing clear, understandable reasons for why a piece of media is classified as AI-generated or human-created is essential for trust and adoption.

By addressing these gaps, this project aims to advance the field of AI-generated media detection, providing more robust, scalable, and explainable solutions across multiple modalities.

2.6 CONCLUSION

The detection of AI-generated media is a critical and evolving field that addresses the challenges posed by advanced AI technologies capable of producing highly realistic synthetic content. Each technique discussed in this chapter—ranging from spectral analysis for audio to deep learning models for images—offers unique strengths and faces specific limitations. The integration of these techniques provides a comprehensive toolkit for identifying synthetic media, enhancing our ability to verify content authenticity in an increasingly digital world. Future developments, including the adoption of blockchain for media provenance and synthetic media watermarking, promise to further advance the effectiveness of these detection methods. Continued research and technological innovation are essential to staying ahead of the evolving capabilities of AI in media generation.

3. SYSTEM DESIGN AND ANALYSIS

3.1 SYSTEM ARCHITECTURE

The system architecture of our AI-generated content detection platform consists of several interconnected components designed to provide a seamless and efficient user experience. The architecture is divided into three main layers: the presentation layer, the application layer, and the data layer.[22]

3.1.1 High-Level System Diagram:

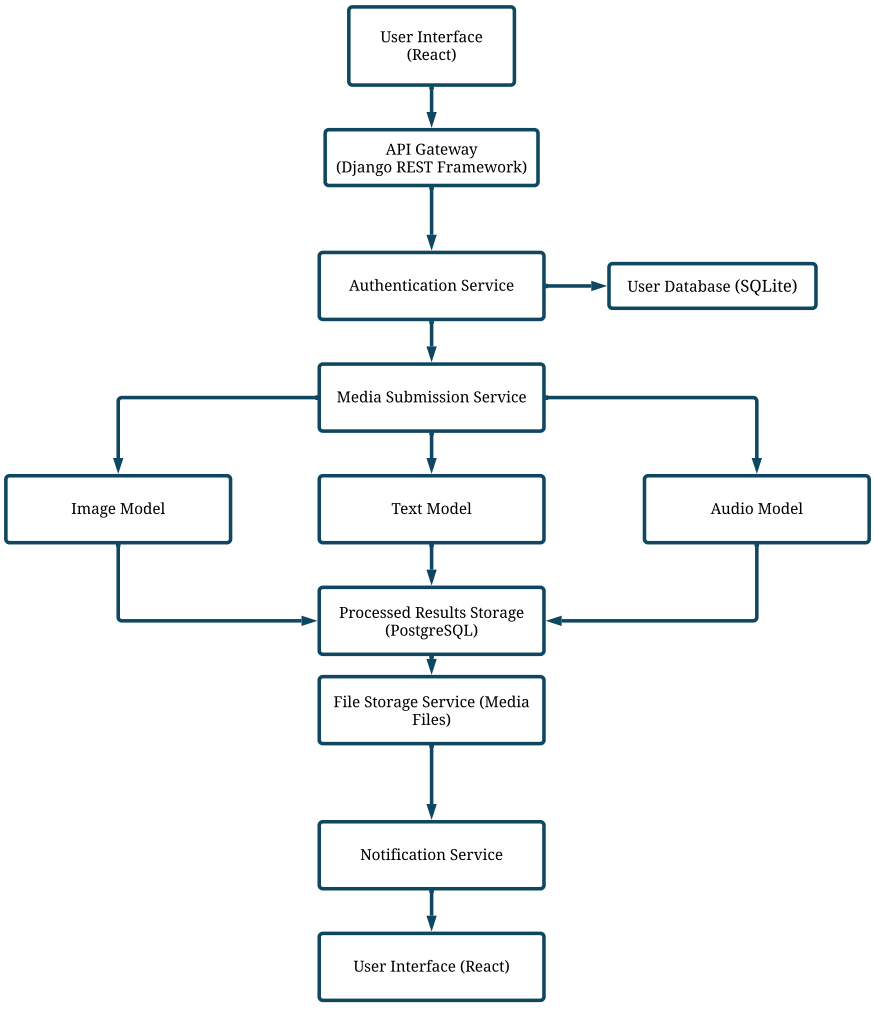


Figure 3.1 System Architecture

- **Presentation Layer**: This layer is responsible for user interactions and consists of the frontend application built using React. It includes user interfaces for authentication, media submission, and result visualization [23].
- **Application Layer:** The core functionality of the system resides in this layer, implemented using Django. It handles business logic, user authentication, media processing, and interaction with the machine learning models [24].
- **Data Layer:** This layer manages data storage and retrieval, using SQLite for structured data (user information, media metadata) and a file storage service for media files. It also includes connections to pre-trained machine learning models for AI content detection.

Interaction Between Components:

1. **User Interaction:** Users interact with the system through the frontend, submitting media for analysis and viewing results.
2. **API Requests:** The frontend communicates with the backend via RESTful APIs to perform actions like user authentication, media submission, and fetching detection results.
3. **Media Processing:** Submitted media is processed by the backend, which invokes the appropriate machine learning models to analyse the content.
4. **Results Storage:** Analysis results are stored in the database and made available to users through the frontend interface.

3.2 SYSTEM REQUIREMENTS

3.2.1 Functional Requirements

The system offers several key functionalities from a user's perspective: [25]

1. **User Authentication:** Secure login and registration functionalities to manage user access.
2. **Media Submission:** Users can submit images, text, and audio files for AI-generated content detection.
3. **Detection Results:** The system provides detailed reports on the authenticity of the submitted media, indicating whether it is AI-generated or human-created.
4. **Subscription Plans:** Users can subscribe to different plans offering various levels of access and functionality (e.g., number of submissions, advanced analysis features).
5. **Admin Management:** Administrators can manage users, review system usage, and update detection models.

3.2.1.1Use Case Diagram [26]:

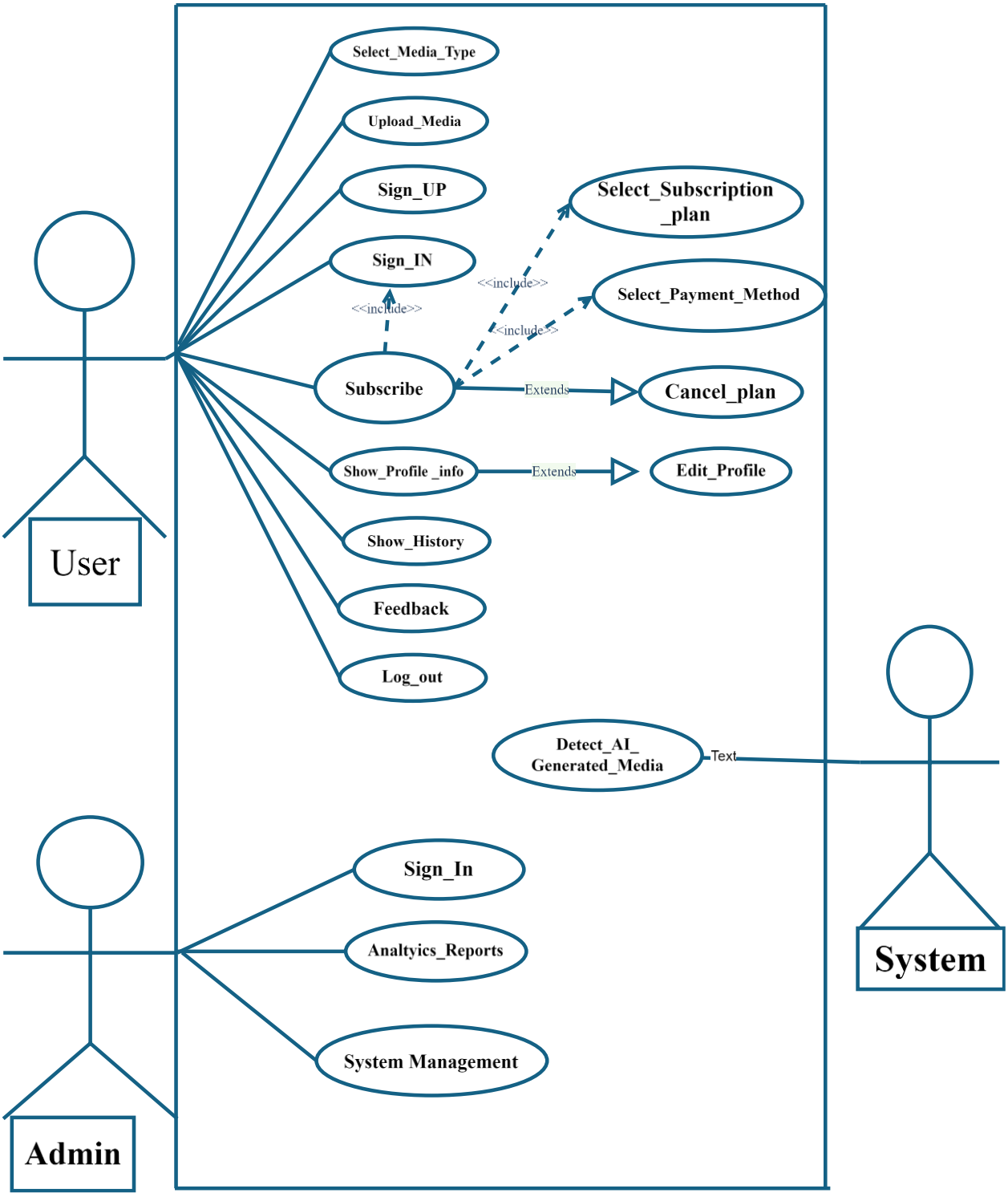


Figure 3.2 Use Case Diagram

Use Case Definitions:

- **User Authentication:** Users can create accounts, log in, and manage their profiles.
- **Media Submission:** Users can upload images, text, and audio for analysis.

- **View Detection Results:** Users can view detailed reports on the submitted media.
- **Subscription Management:** Users can subscribe to different service plans.
- **Admin Management:** Administrators can manage users and system settings.

3.2.1.2 Activity Diagram: [27]

1. Sign up activity:

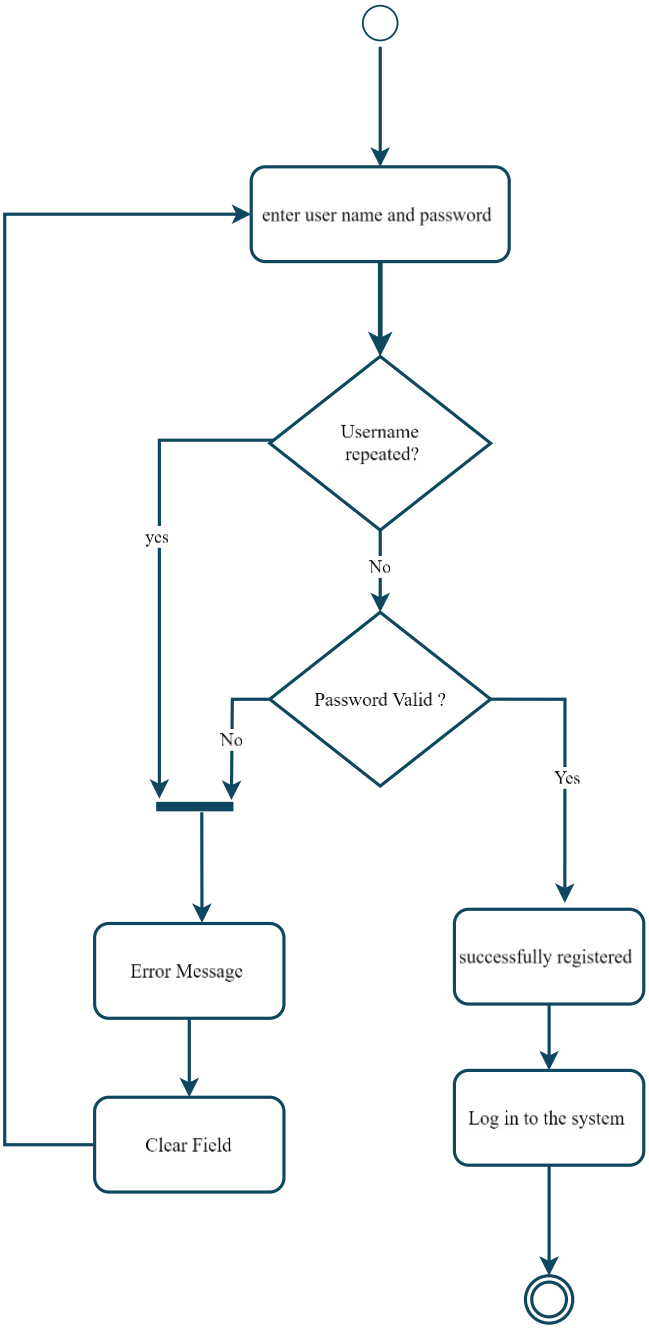


Figure 3.3 Sign up activity

2. Sign in activity:

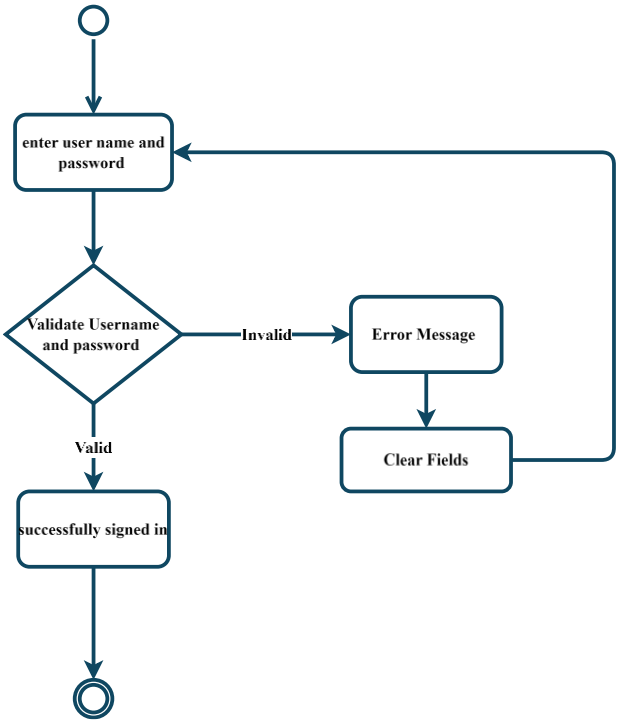


Figure 3.4 Sign in activity

3. Select media type activity:

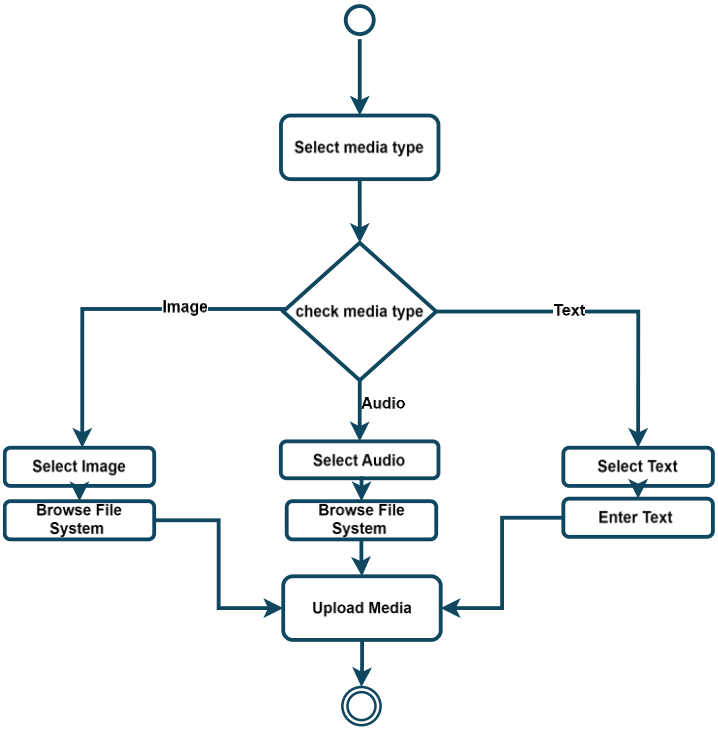


Figure 3.5 Select media type activity

4. Upload media activity

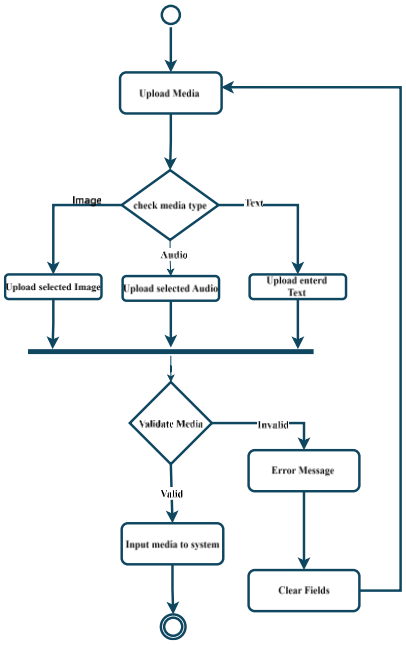


Figure 3.6 Upload media activity

5. Detect AI-generated Media activity:

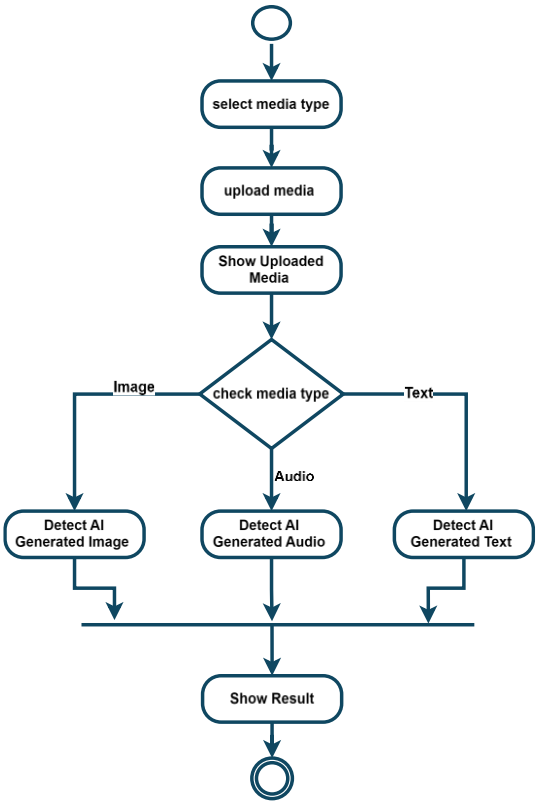


Figure 3.7 Detect AI-generated Media activity

6. Feedback activity:

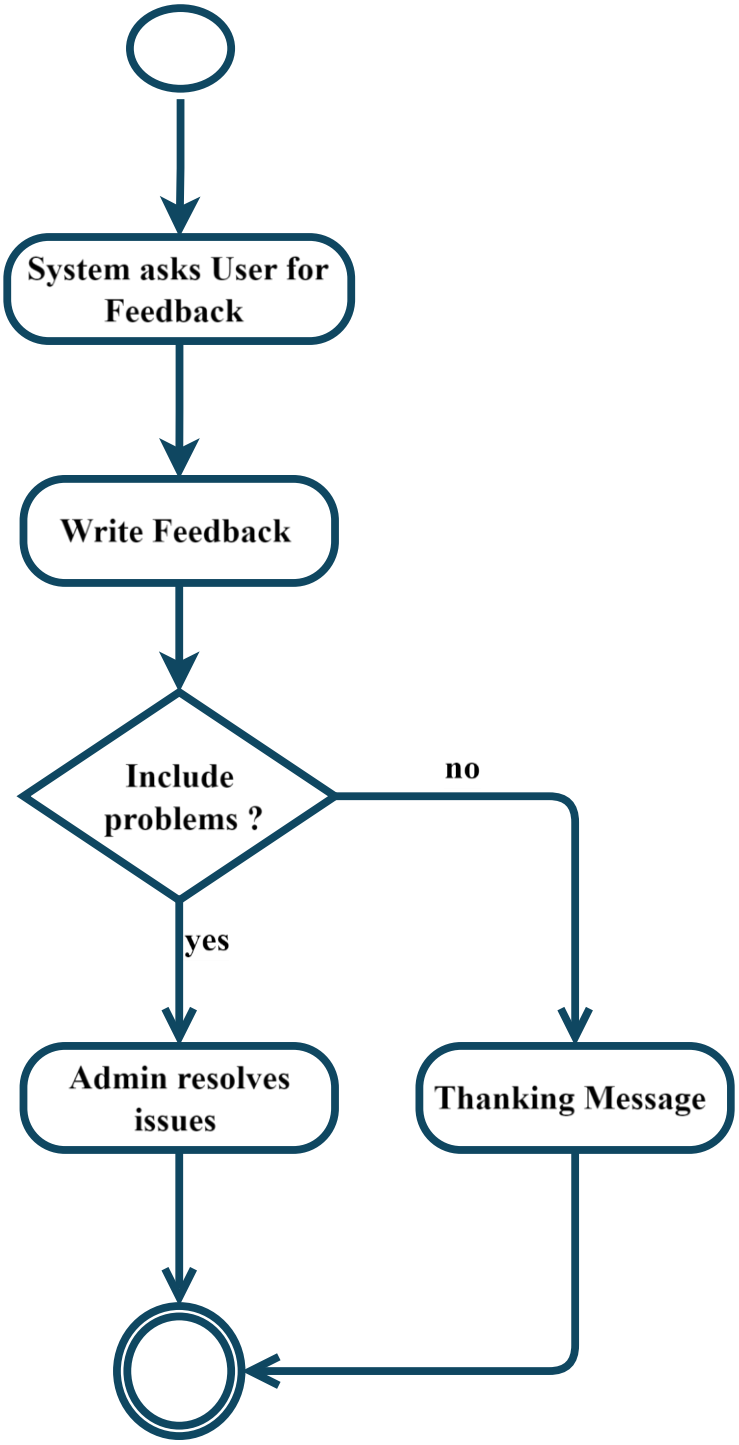


Figure 3.8 Feedback activity

7. Subscription activity:

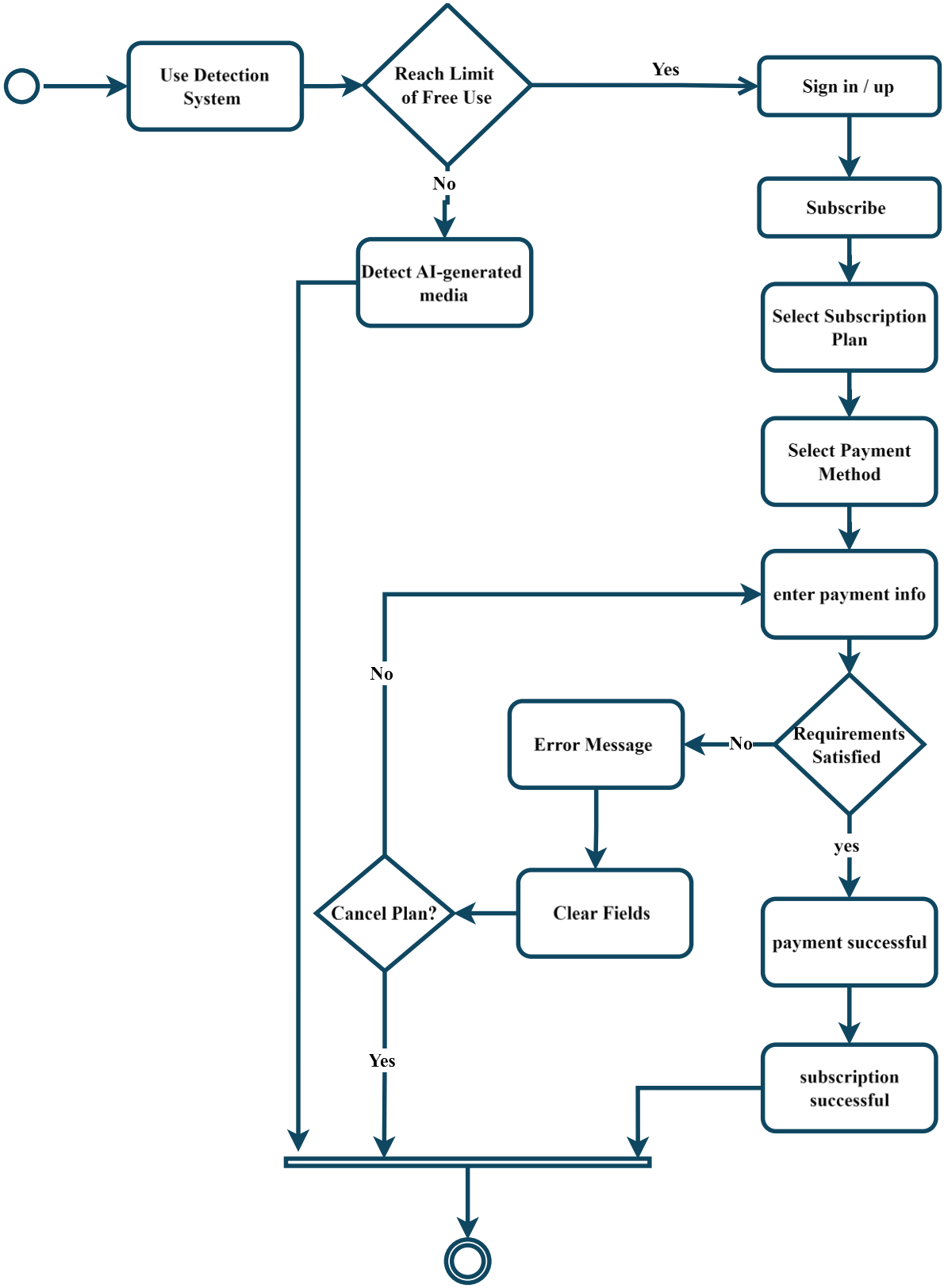


Figure 3.9 Subscription activity

8. Admin Management activity:

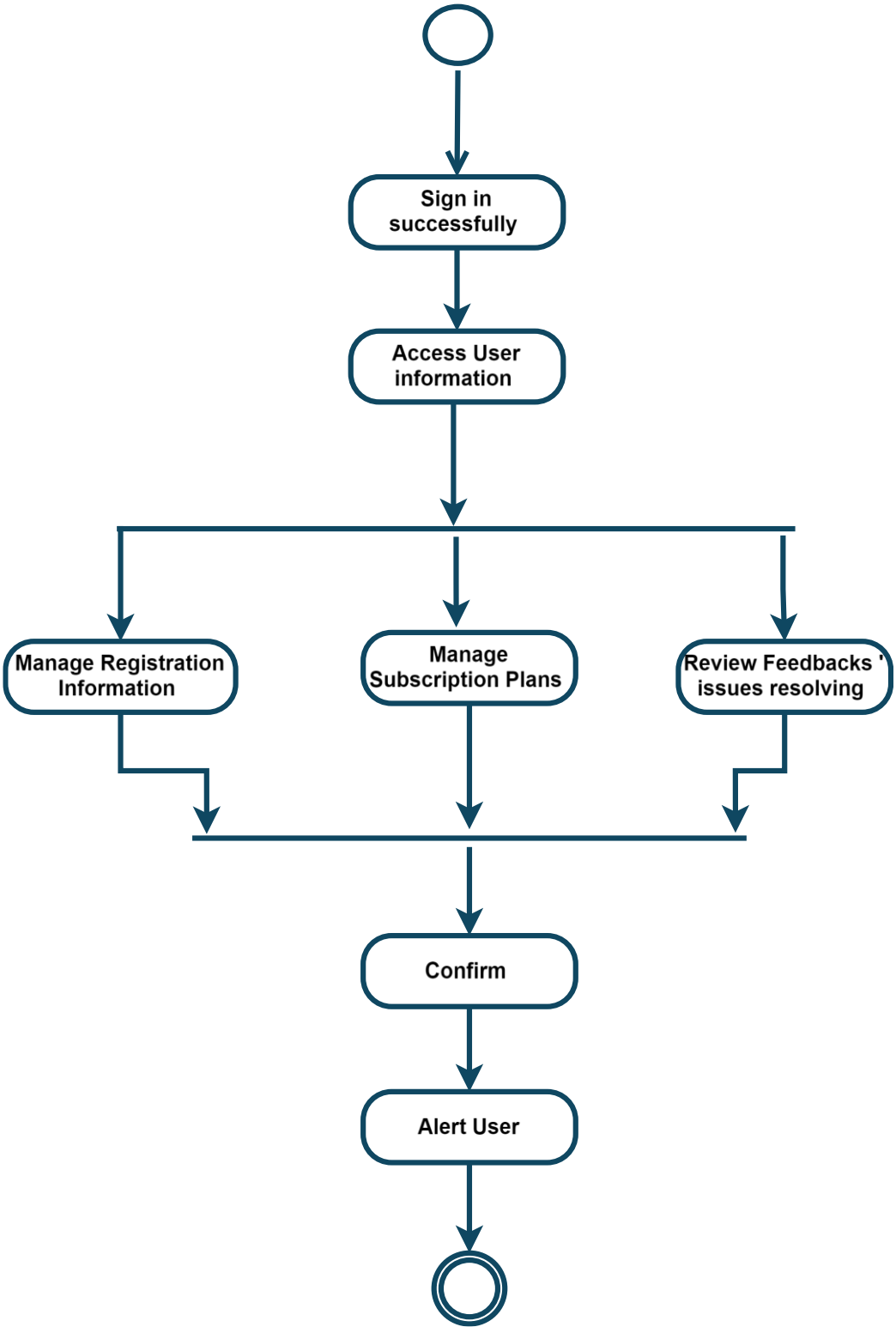


Figure 3.9 Admin Management activity

9. Show History activity:

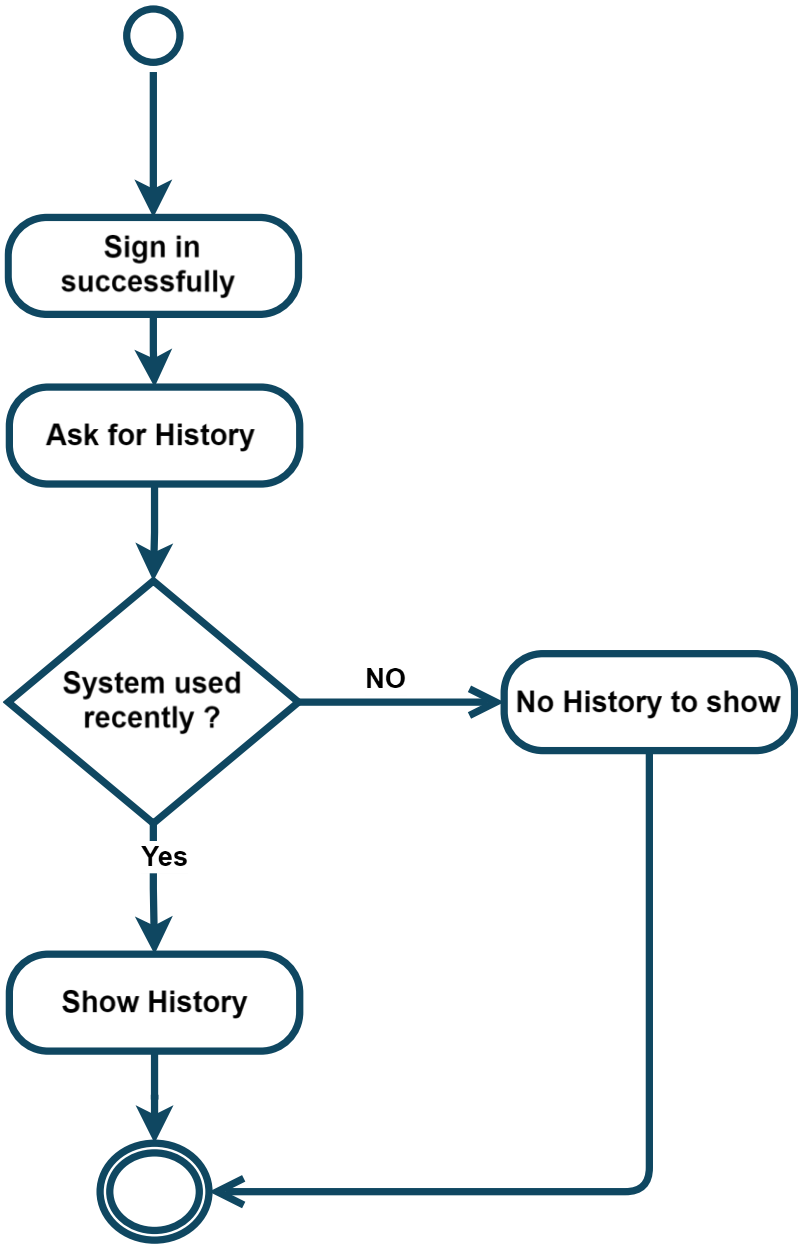


Figure 3.10 Show History activity

10. Edit Profile activity:

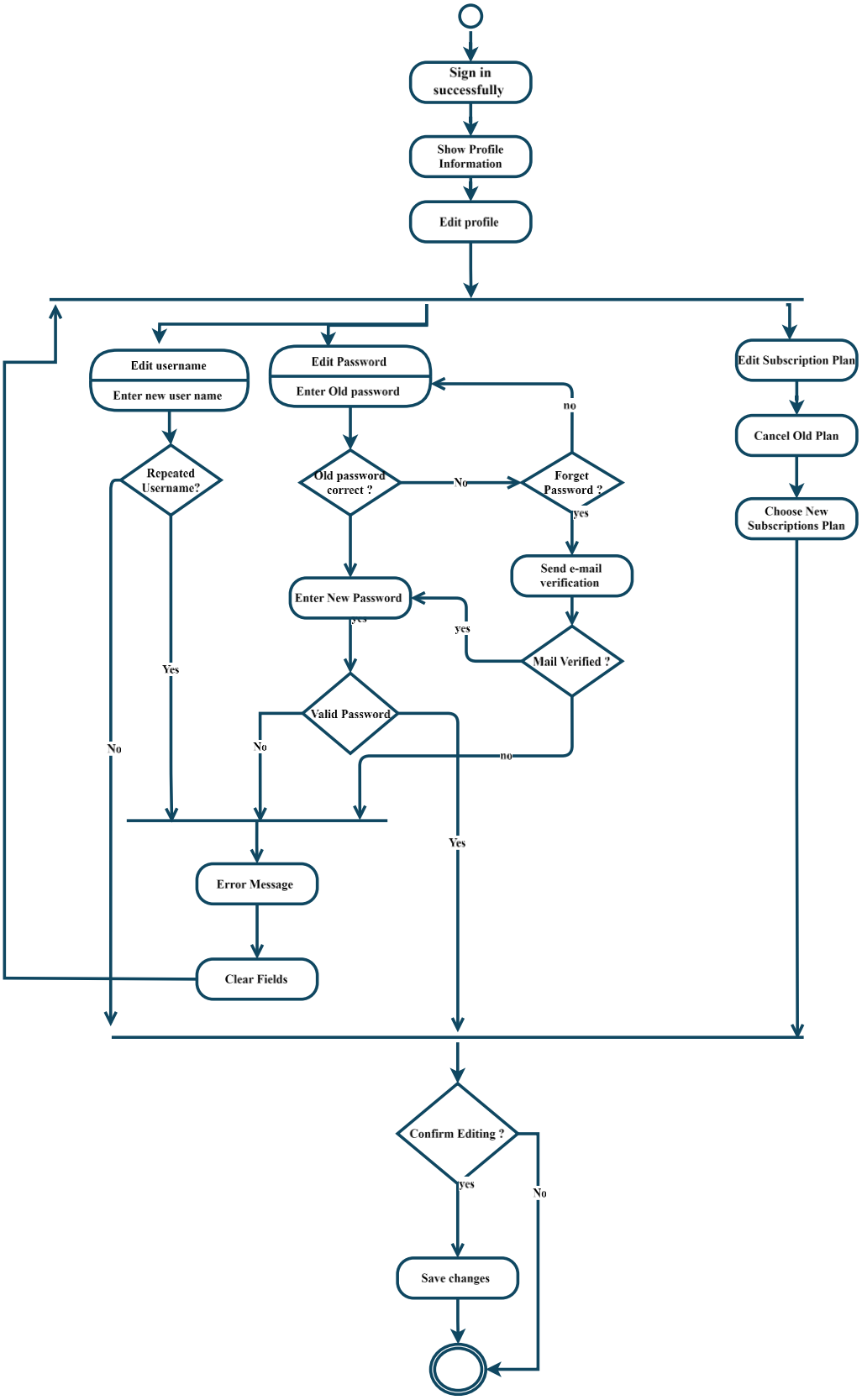


Figure 3.11 Edit Profile activity

3.2.2 NON-FUNCTIONAL REQUIREMENTS

During the development of the system, several non-functional aspects were considered to ensure a robust and user-friendly application:

1. **Security**: Ensuring secure user authentication, protecting user data, and preventing unauthorized access to the system [28].
2. **Usability**: Designing an intuitive and user-friendly interface to facilitate easy interaction with the system.
3. **Performance**: Optimizing the system to handle multiple concurrent users and process large media files efficiently.
4. **Scalability**: Designing the system to scale horizontally, allowing for the addition of more servers to handle increased load.
5. **Reliability**: Ensuring the system is reliable and available with minimal downtime, using strategies like load balancing and regular backups.

Context Diagram:

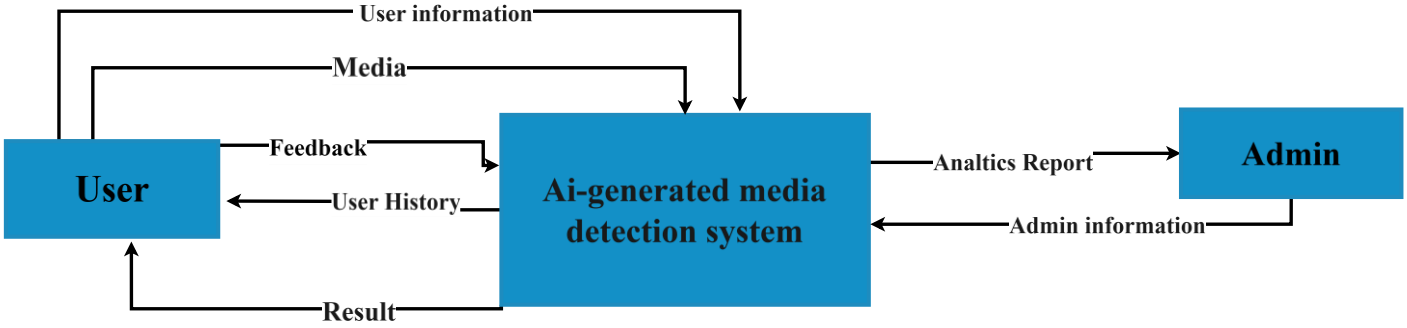


Figure 3.012 Context Diagram

Data Flow Diagram Level 0:

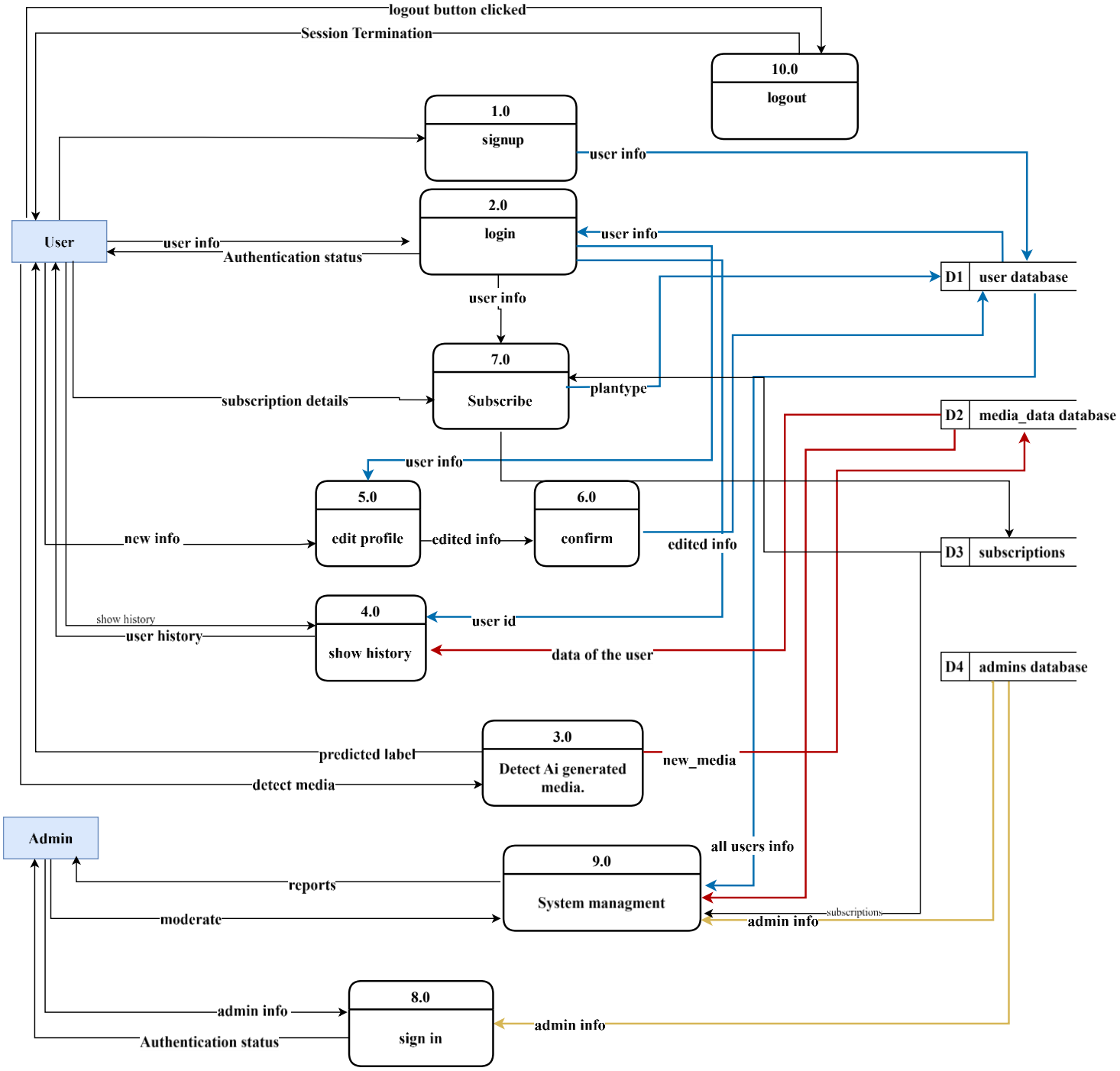


Figure 3.13 Data Flow Diagram Level 0

Data Flow Diagram Level 1:

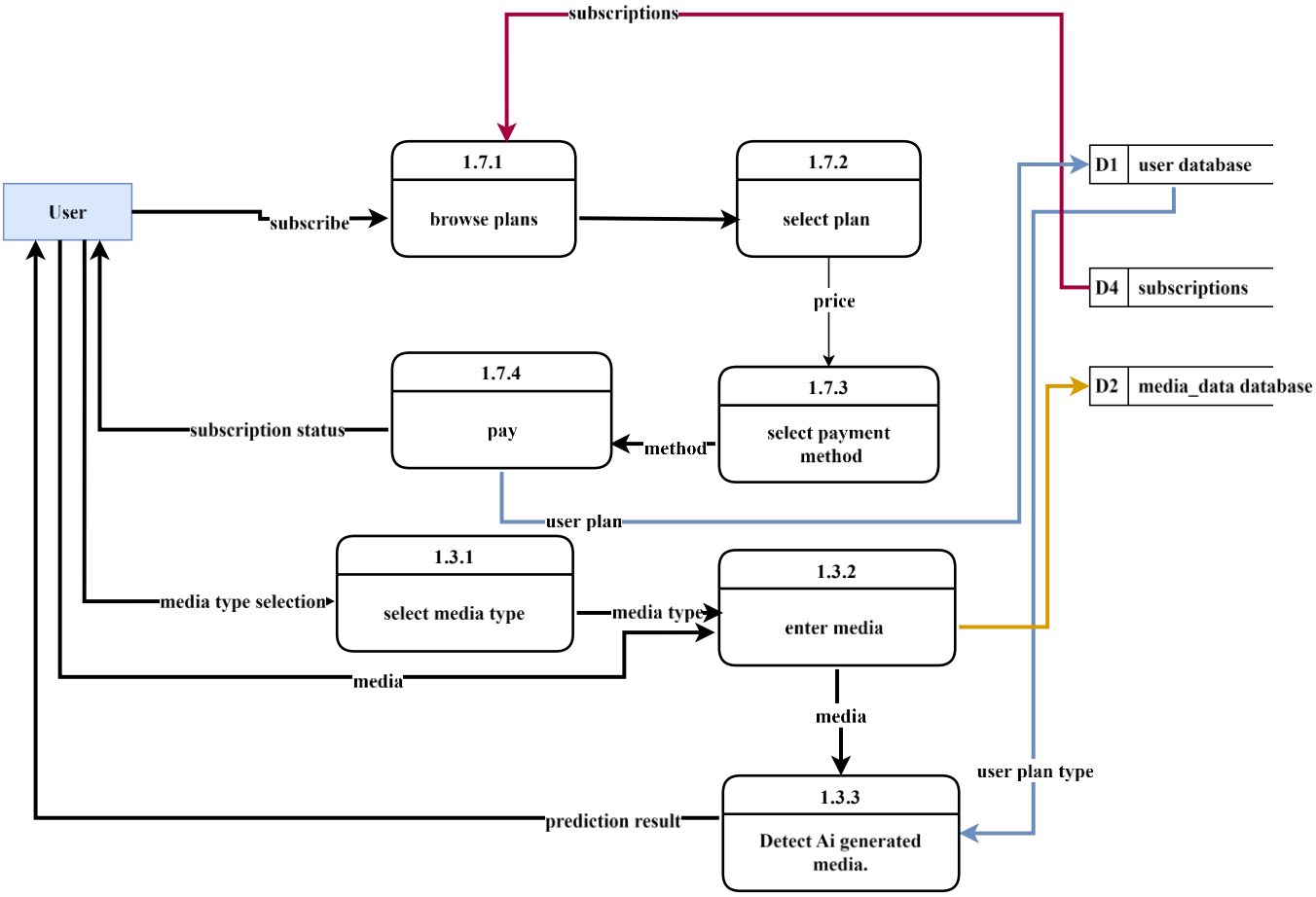


Figure 3.14 Data Flow Diagram Level 1

3.2.3 Design Patterns

To enhance the system's design and maintainability, several design patterns were implemented:

1. **Model-View-Controller (MVC):** The MVC pattern was used for the overall structure of the system, separating concerns into three main components: Models (data), Views (UI), and Controllers (business logic). This separation improves maintainability and allows for independent development and testing of each component.
    - Benefit: Facilitates a clear separation of concerns, making the system easier to manage and extend.
2. **Multimodal Architecture:** This pattern was used to integrate different modalities (image, text, audio) into the system, allowing it to process and analyse various types of media through a unified interface.
    - **Benefit:** Provides flexibility to handle different types of input data using a cohesive architecture, improving extensibility.
3. **Decorator Pattern:** Implemented for flexible processing pipelines, allowing dynamic addition of processing steps for media analysis without modifying the core logic.
    - **Benefit:** Enhances the flexibility and reusability of the media processing pipeline, enabling easy modification and extension of processing steps.

4. **Observer Pattern:** Used for event handling, particularly for real-time notifications and updates, such as informing users about the status of their media analysis [29].
    - **Benefit:** Decouples event handling from core logic, allowing for scalable and maintainable event-driven architecture.

3.2.3.1Sequence Diagram [30]:

1. User

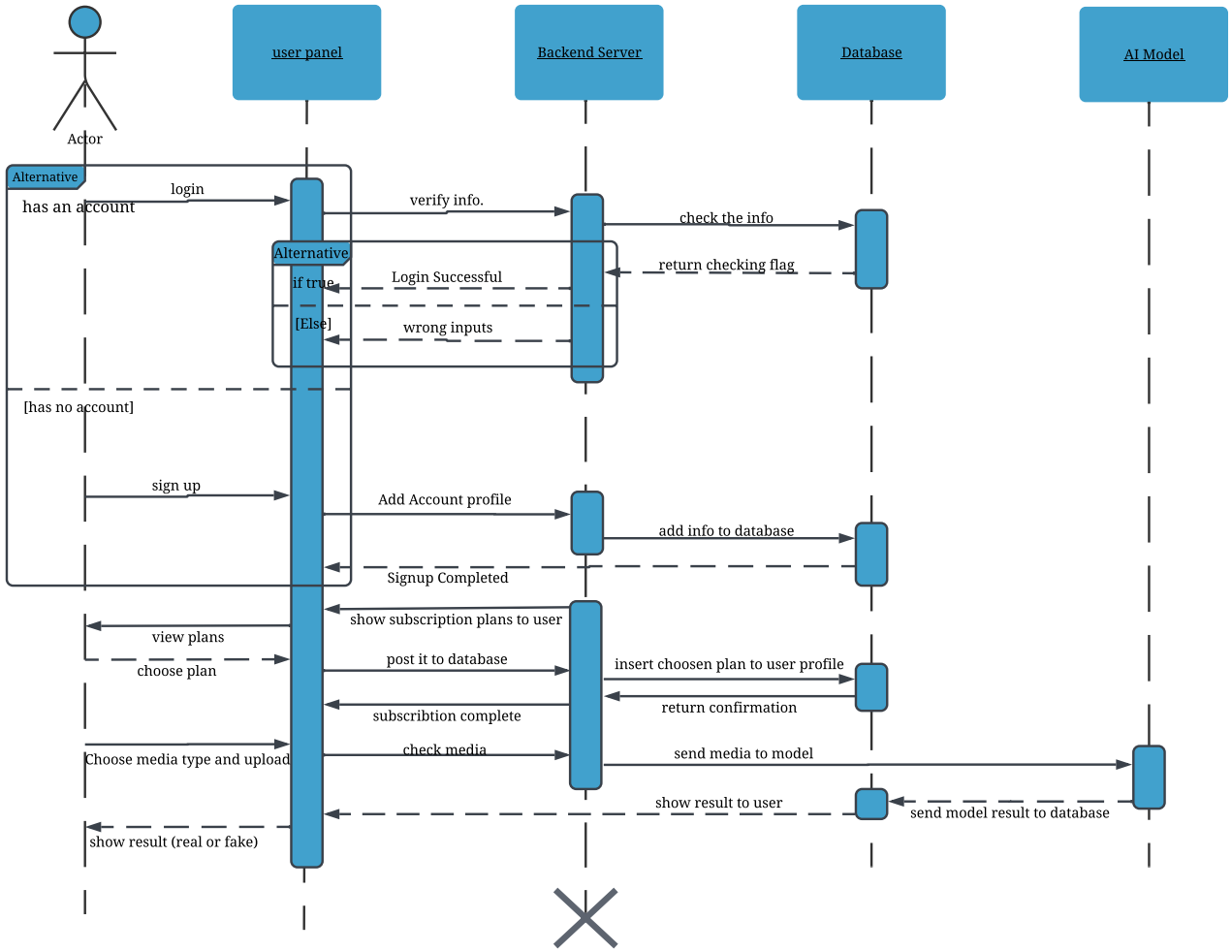


Figure 3.15 User Sequence Diagram

2. Administrator

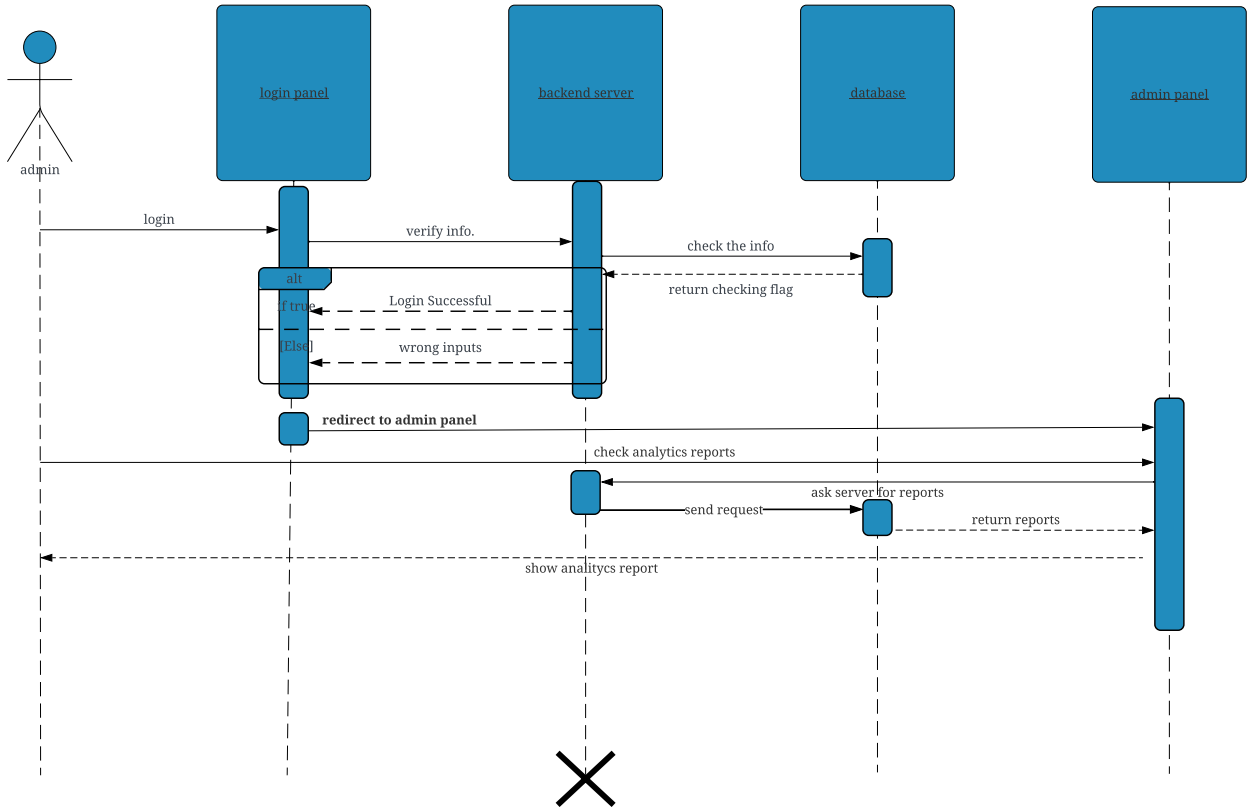


Figure 3.16 Administrator Sequence Diagram

Class Diagram:

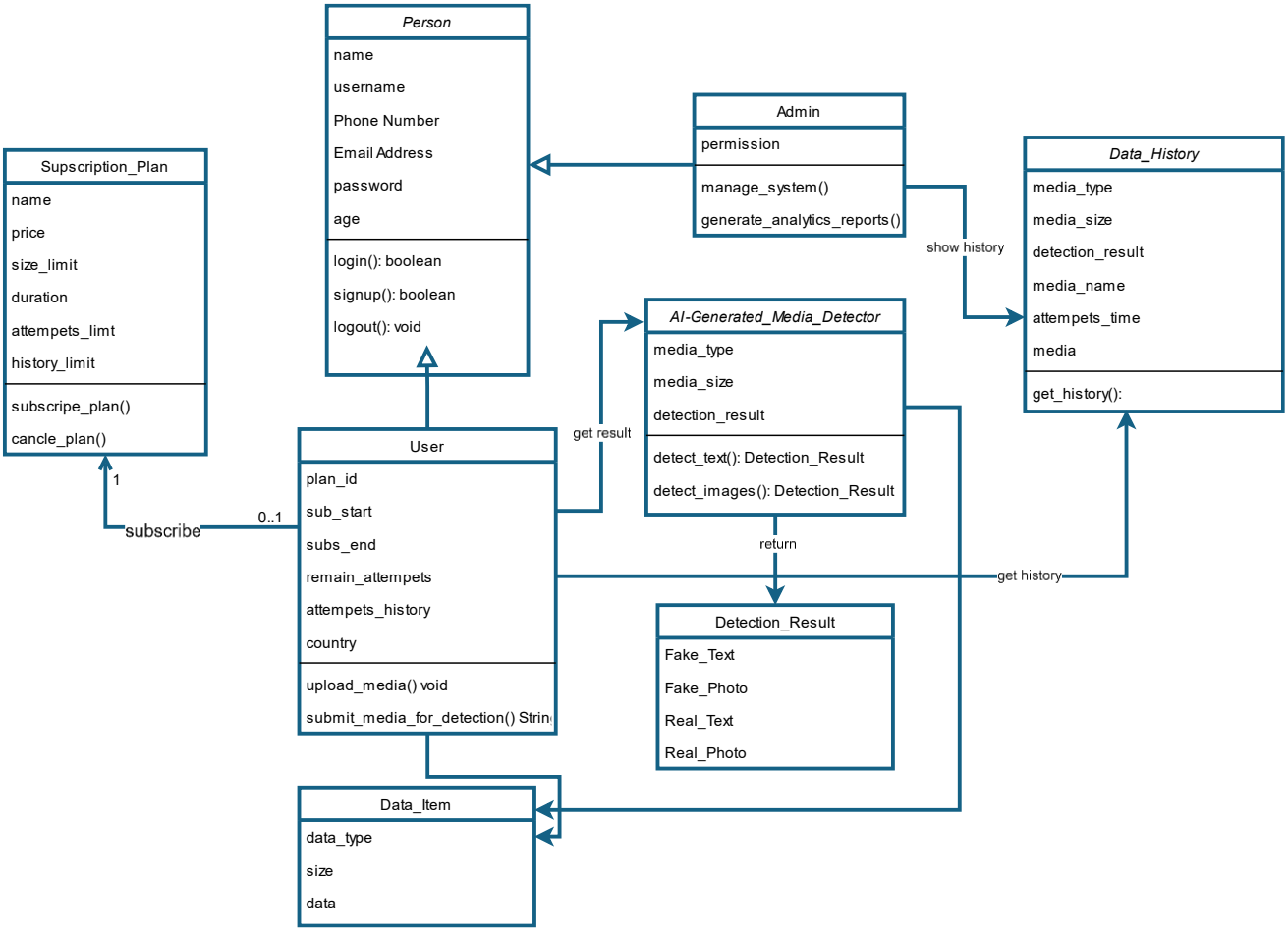


Figure 3.17 Class Diagram

3.2.4 DATABASE DESIGN

Database Schema [31]:

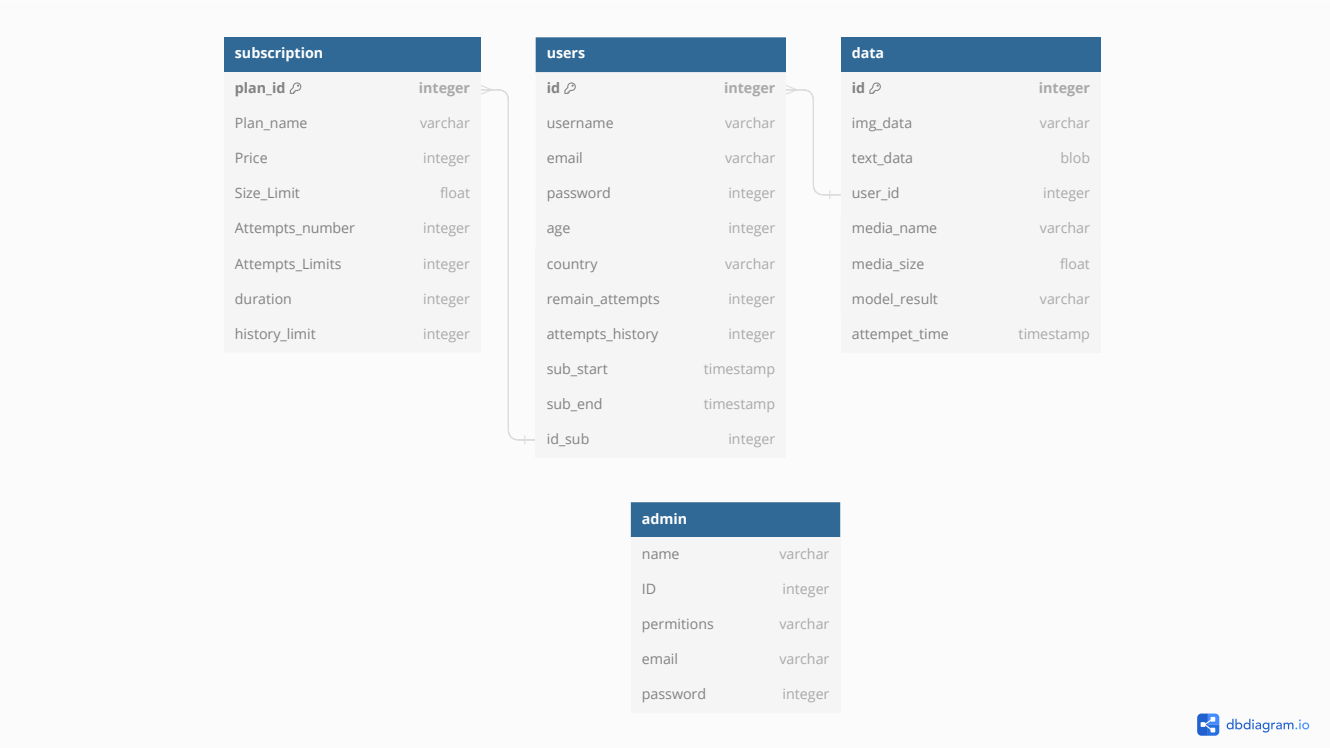


Figure 3.18 Database Schema

- Users Table: Stores user information, including authentication details.
- Media Table: Stores metadata and file paths for submitted media.
- Subscription Plans Table: Stores subscriptions plan specifications.
- Admins Table: Stores admins credentials and permissions.

ERD Diagram:

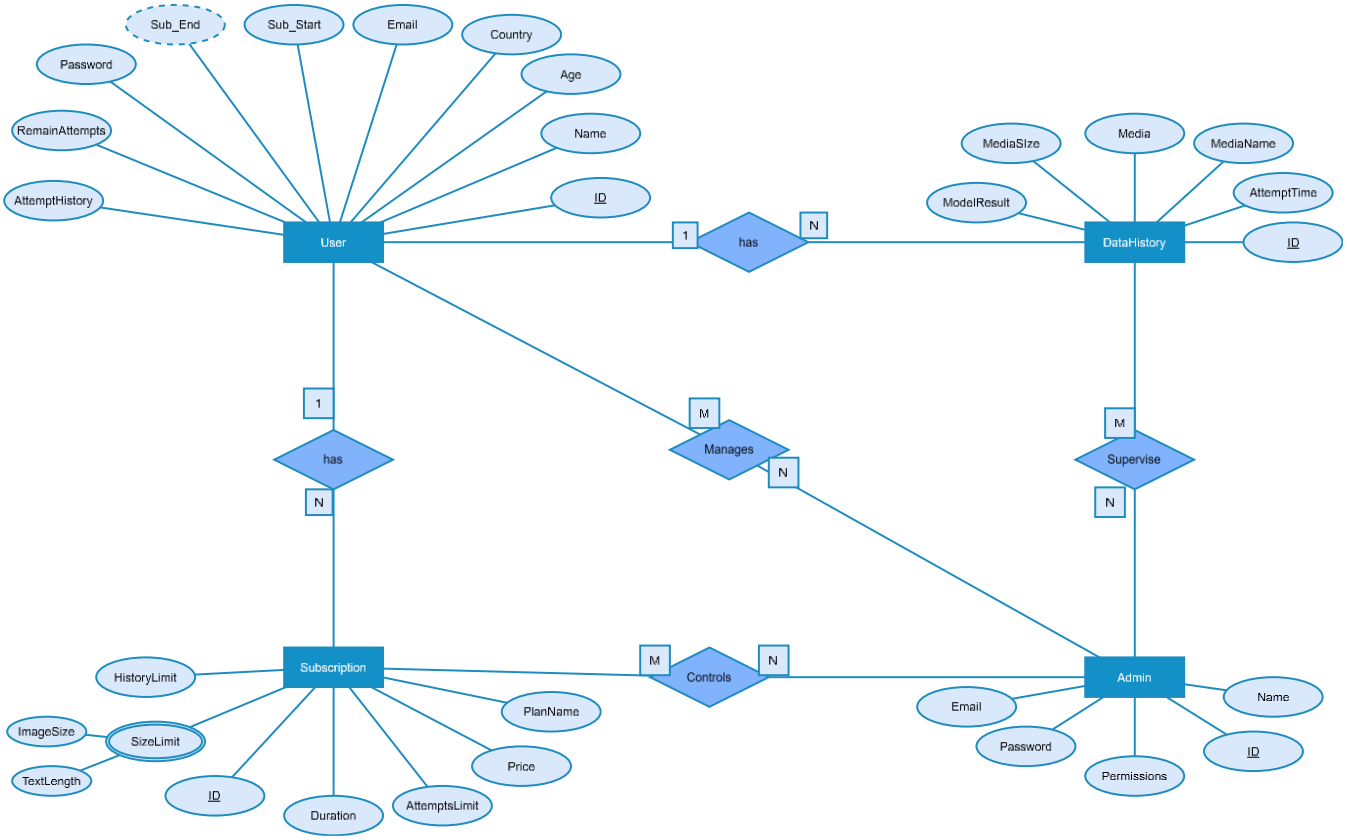


Figure 3.19 ERD Diagram

Used Technologies and Tools

- **Frontend**: React, Bootstrap
- **Backend**: Django, Django REST Framework
- **Database**: SQLite
- **Machine** Learning: Hugging Face (RoBERTa, DeBERTa, Wav2Vec2, EfficientNet)

3.3 SUMMARY

This chapter outlined the comprehensive system design and analysis for the AI-generated content detection platform. We discussed the overall system architecture, functional and non-functional requirements, and detailed the design patterns used to ensure a robust and maintainable system. Additionally, we included various diagrams to illustrate the system's structure, interactions, and data flow, providing a clear overview of the project's technical foundation

4. PROJECT METHODOLOGY

4.1 DATA ACQUISITION

4.1.1Data Sources and Datasets:

Audio:

- **Fake-or-Real Dataset (FoR):** A collection of audio files comprising both real and fake audio samples, used to establish baseline detection capabilities [32].
- **Scenefake Dataset:** Includes a variety of deepfake audio clips generated using different techniques, enhancing the dataset's diversity [33].
- **In the Wild Dataset:** Contains real and fake audio samples collected from various internet sources, reflecting more natural and diverse scenarios [34].
- **ASVspoof 2019 Dataset:** Used in Automatic Speaker Verification and Spoofing Countermeasures challenges, featuring a mix of genuine and spoofed audio [35].
- **ASVspoof 2021 Dataset:** An updated dataset with more recent examples of spoofed audio, capturing advancements in deepfake audio generation [36].

•	ASVspoof	In-the-Wild	SceneFake	FoR
Year	2021	2022	2022	2019
Language	English	English	English	English
Goal	Detection	Detection	Detection	Detection
Fake Types	VC, TTS	TTS	TTS	TTS
Condition	Noisy	Noisy	Clean	Clean
Format	FLAC	WAV	WAV	WAV
SR (Hz)	16k	16k	16k	16k
SL (s)	0.5~12	2~8	1~13	0.5~20
#Hours	325.8	38.0	64	150.3
#Real Utt	22, 617	19, 963	11,407	108, 256
#Fake Utt	589, 212	117, 985	47,367	87, 285
#Real Spk	48	58	48	33
#Fake Spk	48	58	48	140
Accessibility	Public	Public	Public	Public

Table 4-1

Image:

- **140k Real and Fake Faces:** Combines 70,000 real faces from the Flickr dataset with 70,000 fake faces generated by StyleGAN, resized to 256px [37].
- **CelebA-HQ resized (256x256):** Contains 30,000 high-quality celebrity faces, suitable for training and evaluating generative models [38].
- **Synthetic Faces High Quality (SFHQ) Part 2:** Includes 91,361 high-quality 1024x1024 curated face images, enhanced using StyleGAN2 techniques [39].
- **Face Dataset Using Stable Diffusion v1.4:** Comprises real faces from the Flickr dataset and fake faces generated by Stable Diffusion models, resized to 256px [40].
- **Stable Diffusion Face Dataset:** AI-generated human faces using Stable Diffusion 1.5, 2.1, and SDXL 1.0 checkpoints, covering resolutions of 512x512, 768x768, and 1024x1024 [41].
- **Synthetic Faces High Quality (SFHQ) Part 3:** Contains 118,358 high-quality 1024x1024 face images generated by StyleGAN2, utilizing advanced truncation tricks [42].
- **Synthetic Human Faces for 3D Reconstruction:** Generated by drawing samples from the EG3D model, resulting in high-quality 512x512 synthetic face images [43].” Refer to Appendix A 2”

<table>
<tr><th colspan="6">200K images</th></tr>
<tr><td colspan="6">100 k Real
100 k Fake</td></tr>
<tr><td colspan="2">Train</td><td colspan="2">Validate</td><td colspan="2">Test</td></tr>
<tr><td>Real: 70 k</td><td>Fake: 70 k</td><td>Real: 15 k</td><td>Fake: 15 k</td><td>Real: 15 k</td><td>Fake: 15 k</td></tr>
<tr><td rowspan="6">70k Flickr</td><td>3k SD V2.1</td><td rowspan="6">15 k CelebA HQ</td><td>3k SDXL 1.0</td><td rowspan="6">15 k CelebA HQ</td><td>2536 SD 1.4</td></tr>
<tr><td>3k SD V1.5</td><td>3k SG1</td><td>3116 SG1</td></tr>
<tr><td>16k SG1</td><td>3k EG3D</td><td>3116 EG3D</td></tr>
<tr><td>16k EG3d</td><td>16k S SG2</td><td>3116 S SG2</td></tr>
<tr><td>16k S SG2</td><td>16k S SD1.4</td><td>3116 S SD1.4</td></tr>
<tr><td>16k S SD1.4</td></tr>
<tr><td colspan="2">70% Data</td><td colspan="2">15% Data</td><td colspan="2">15% Data</td></tr>
</table>

Table 4-2

Text:

- **LLM Generated Essays for the Detect AI Comp:** Contains 700 essays, with 500 generated using GPT-3.5-turbo and 200 with GPT-4.

- **DAIGT Data - Llama 70b and Falcon 180b [44]:**
  - Llama Falcon v3: 7,000 LLM-generated essays.
  - Llama 70b v2: 1,172 LLM-generated essays.
  - Llama 70b v1: 1,172 LLM-generated essays.
  - Falcon 180b v1: 1,055 LLM-generated essays.

- **Persuade Corpus 2.0:**
  Comprises over 25,000 argumentative essays by 6th-12th grade students in the U.S.

- **DAIGT External Dataset:**
  Includes 2,421 student-generated texts and 2,421 AI-generated texts, used to balance the training data with real and AI-generated samples.

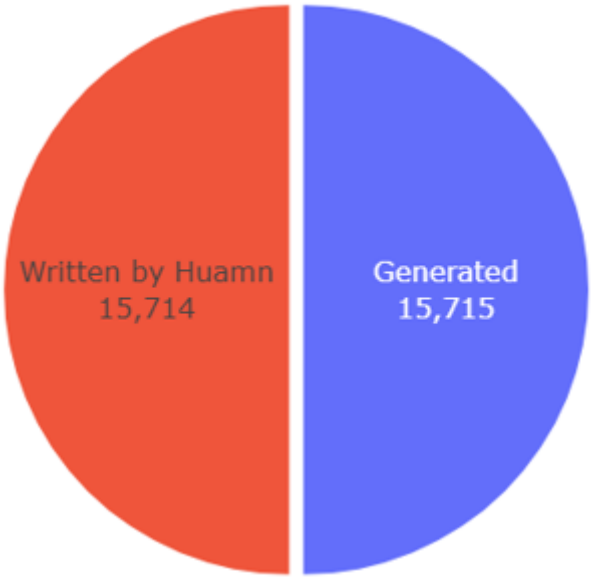


Figure 4.1

Refer to [Appendix A, C] for detailed graphs of the data.

4.1.2Pre-processing Steps:

Text:

- **No Preprocessing:** To preserve the context and integrity of the data, no cleaning steps were applied to the text datasets. This approach ensures that all nuances and characteristics of the text are retained for model training [45].

Audio:

  - **Noise Reduction:** Background noise was minimized using spectral subtraction and other noise reduction techniques to ensure that the features extracted from the audio were not influenced by unwanted noise.
  - **Normalization:** The audio files were normalized to maintain consistent volume levels across all samples, facilitating accurate feature extraction.
  - **Sampling Rate Standardization:** All audio files were resampled to a common sampling rate of 16 kHz, ensuring uniform input data for the model [46].
  - **Truncation:** Each audio file was truncated to the first 15 seconds to reduce computational load and ensure consistent input length for the model [47].

Image:

- **Resizing:** Images were resized to standard dimensions (260x260) to ensure compatibility with the model input requirements [48].
- **Normalization:** Pixel values were normalized to a common scale to improve model training stability.

4.2 MODEL SELECTION

4.2.1 Overview of Initial Model Candidates

We initially evaluated several models for each modality—text, audio, and image detection—based on their reported performance in literature and suitability for our dataset characteristics.

Text:

- **BERT**: Chosen for its foundational architecture and general text classification capabilities.
- **Roberta**: Selected for its robustness in handling nuanced language patterns [49].
- **Deberta**: Considered for its advanced attention mechanisms and improved language understanding [50].

Audio:

- **Wav2Vec2:** Selected for its state-of-the-art performance in speech recognition and anomaly detection.
- **Mel-spectrogram + CNN:** Evaluated for its traditional approach in audio feature extraction [51].
- **ResNet-based Model:** Considered for its ability to capture detailed spectral features.

Image:

- **EfficientNet:** Chosen for its balance between accuracy and computational efficiency [52].
- **ResNet:** Evaluated for its strong performance in image classification [53].
- **Xception:** Considered for its capability to detect fine-grained image features [54].

4.2.2 Evaluation Criteria

Models were evaluated based on the following criteria [55]:

- **Accuracy**: Overall classification performance.
- **Precision and Recall:** Balance between false positives and false negatives.
- **F1 Score**: Combined measure of precision and recall.
- **ROC-AUC**: Trade-off between true positive and false positive rates.

4.2.3 Experimental Setup

Each model was trained using a standardized dataset split of 70% training, 20% validation, and 10% testing. Hyperparameters were tuned using grid search and random search methods, and models were trained on high-performance GPUs.

4.2.4 Results of Initial Models

Text Models:

- **BERT**: Achieved an accuracy of 90% but exhibited signs of overfitting on smaller datasets.
- **RoBERTa**: Outperformed BERT with an accuracy of 99% and better generalization.
- **DeBERTa**: Achieved the highest accuracy of 99%, showing superior handling of complex text patterns.

Audio Models:

- **Wav2Vec2**: Demonstrated excellent performance with a word error rate of 7% and robust anomaly detection.
- **Mel-spectrogram + CNN**: Achieved reasonable accuracy but was outperformed by Wav2Vec2 in detecting subtle anomalies.
- **ResNet-based Model**: Provided good results but was computationally intensive.

Image Models:

- **EfficientNet**: Balanced accuracy and computational efficiency, with an accuracy of 99%.
- **ResNet**: Achieved an accuracy of 99% but required more computational resources.
- **Xception**: Offered detailed feature extraction but was less efficient compared to EfficientNet.

4.2.5 Refinement and Final Model Selection

Based on the initial evaluations, the following refinements were made:

Text:

- **RoBERTa** and **DeBERTa** showed complementary strengths in language understanding and text classification.
- To leverage their combined advantages, we developed an ensemble model where a final linear layer takes the concatenated outputs of both RoBERTa and DeBERTa models.

Audio:

- **Wav2Vec2** was optimized for its learning rate and batch size to reduce training time while maintaining accuracy.

Image:

- **EfficientNet** was selected for its optimal balance between accuracy and computational efficiency, with additional data augmentation applied to improve generalization.

4.2.6 Final Model

The final models chosen for each modality were:

Text:

Ensemble of RoBERTa and DeBERTa - The outputs of RoBERTa and DeBERTa are concatenated and fed into a final linear layer. This approach was chosen for its superior combined performance, leveraging the strengths of both models.

- **Architecture**:
  - **RoBERTa Output:** Captures robust language patterns.
  - **DeBERTa Output:** Provides nuanced language understanding.
  - **Final Linear Layer:** Integrates the concatenated outputs to enhance overall classification performance.

Audio:

- **Wav2Vec2** - Chosen for its state-of-the-art performance in audio anomaly detection.

Image:

- **EfficientNet** - Preferred for its efficiency and high accuracy in distinguishing real from AI-generated images.

The ensemble model for text was validated on additional test datasets to confirm its robustness and ability to generalize across various scenarios. This ensemble approach demonstrated significant improvements over individual models, providing a more comprehensive understanding and classification of text inputs.

5. IMPLEMENTATION

5.1 INTRODUCTION

This chapter dives into the implementation phase of the AI-Generated Media Detection project, where theoretical concepts are translated into a robust system. The primary objective is to develop a functional platform capable of distinguishing between media content created by humans and artificial intelligence. This phase is pivotal as it transforms ideas into a tangible product ready for practical use.

Throughout this chapter, we detail the technical strategies, methodologies, and tools employed to achieve our objectives. Key components include algorithm development, integration of machine learning models, user interface design considerations, and deployment strategies. Each step is meticulously crafted to ensure the reliability, accuracy, and user-friendliness of the AI-Generated Media Detection system [56].

By the conclusion of this chapter, readers will gain a comprehensive understanding of how the project evolved from concept to implementation, setting the stage for testing, refinement, and potential future enhancements.

5.2 SYSTEM ARCHITECTURE

5.2.1 Overview of System Architecture:

The AI-Generated Media Detection project adopts a modular architecture, separating frontend and backend components to ensure scalability, maintainability, and flexibility. This architecture facilitates independent development and deployment of each layer while promoting efficient communication through RESTful APIs [57].

5.2.2 Model or Pattern: Client-Server Architecture with RESTful APIs:

The AI-Generated Media Detection project adopts a Client-Server Architecture with a clear separation of concerns between the frontend (client) and backend (server) components. This architectural pattern is structured around the following principles.

5.2.3 Client (Frontend):

- **Responsibilities:** Handles presentation logic and user interaction, providing a responsive and intuitive interface for users to interact with the application.
- **Communication:** Communicates with the backend server via HTTP requests using RESTful APIs to fetch data, submit user actions (e.g., authentication, media detection requests), and display results.

5.2.4 Server (Backend):

- **Responsibilities:** Implements business logic, manages data persistence, and integrates with external services (such as deep learning models for media detection).
- **APIs:** Exposes RESTful APIs that the frontend consumes, handling requests for user authentication, media detection, user profile management, and other application functionalities.
- **Security:** Implements JWT tokens for secure authentication and authorization, ensuring data integrity and privacy.

5.2.5 Deep Learning Models:

- **Responsibilities:** Hosted separately from the web application in a scalable environment (cloud infrastructure).
- **Integration:** Accessed via APIs exposed by the backend, these models perform real-time inference to detect AI-generated media based on user requests initiated through the frontend.

5.2.6 Advantages of Client-Server Architecture with RESTful APIs:

- **Scalability:** Allows each component (frontend, backend, and deep learning models) to scale independently based on demand.
- **Modularity:** Separation of concerns facilitates easier maintenance, updates, and enhancements to specific components without impacting others.
- **Flexibility:** Supports diverse client applications (web, mobile) that can interact with the same backend services through standardized APIs.
- **Security:** Enables secure communication between client and server using token-based authentication and HTTPS protocols.

5.3 TECHNOLOGIES USED:

5.3.1Languages

In our web project, we use several languages to create a dynamic and well-designed application:

Table 5-3

1. JavaScript: JavaScript is our main programming language. It handles the functionality of the application, including user interactions and data processing.
2. JSX (JavaScript XML): JSX is a special syntax used with React. It looks like HTML but is written inside JavaScript. It helps us build the user interface easily.
3. HTML (Hypertext Markup Language): HTML is used to structure the content on our web pages. It defines the layout and elements on the page.
4. CSS (Cascading Style Sheets): CSS is used to style the web application. It controls the look and feel, such as colours, fonts, and layout. We use CSS directly and through libraries like Bootstrap and styled components.

By using these languages, we create a functional and attractive web application.

5.3.2 Frameworks

In our web project, we use several frameworks to build a robust and efficient application:

1. React [58]**:** React is a JavaScript library for building user interfaces. It helps us create reusable components, making our code more modular and easier to manage.
2. Bootstrap [59**]:** Bootstrap is a front-end framework for designing responsive and mobile-first websites. It provides pre-designed components and a grid system, which helps in building a consistent and attractive UI.
3. React Bootstrap: React Bootstrap is a library that integrates Bootstrap components with React, allowing us to use Bootstrap's design elements directly within our React components.
4. Styled Components: Styled Components is a library for styling React components using CSS-in-JS. It allows us to write CSS directly within our JavaScript code, providing a more dynamic and scoped styling solution.
5. React Router DOM: React Router DOM is a routing library for React applications. It helps us manage navigation and rendering of different components based on the URL.
6. Reactstrap: Reactstrap is another library that provides Bootstrap components specifically for React. It offers a set of React components that follow Bootstrap's design guidelines.
7. MDB React UI Kit: MDB React UI Kit is a collection of React components based on Material Design Bootstrap. It provides additional UI elements and styles to further enhance the visual appeal and user experience of our application.

Table 5-4

5.3.3 Tools & Libraries

In addition to the frameworks, we utilize various tools and libraries to streamline our development process and enhance the functionality of our website:

1. Font Awesome o Packages: @fortawesome/fontawesome-free, @fortawesome/free-solid-svg-icons, @fortawesome/react-fontawesome • Purpose: Provides a wide range of icons to enhance the visual appeal and usability of our application
2. Testing Libraries • Packages: `@testing-library/jest-dom`, `@testing-library/react`, `@testing-library/user-event` • Purpose: Used for testing React components to ensure reliability and performance.
3. AOS (Animate on Scroll) • Package: `aos` • Purpose: Adds scroll animations to our application, improving the user experience.

4. Axios • Package: `axios` • Purpose: A promise-based HTTP client used for making API requests and handling responses efficiently.
5. Formik • Package: `formik` • Purpose: Simplifies form management, validation, and handling in React applications.
6. Lodash • Package: `lodash` • Purpose: A utility library that provides helpful functions for common programming tasks, making our code cleaner and more efficient.
7. Use HTTP • Package: `use-http` • Purpose: A custom hook for making HTTP requests in React applications, simplifying data fetching and handling.
8. Web Vitals • Package: `web-vitals` • Purpose: Measures and reports essential web performance metrics to ensure a smooth user experience.
9. Yup • Package: `yup` • Purpose: A schema validation library used for validating form inputs and ensuring data integrity.
10. React Icons • Package: `react-icons` • Purpose: A library of popular icons for React projects, providing a convenient way to include various icons in our application.

Table 5-5

5.3.4 IDEs & Tools

- VS Code: Popular IDE known for its extensibility and robust feature set.
- Postman [60]: API testing and development tool simplifying API workflow.
- Kaggle: Platform for data science and machine learning competitions and datasets.
- Hugging Face: Repository for NLP models and datasets, facilitating model sharing and deployment.

5.3.5 Dependencies Management

- NPM [61](Node Package Manager): Used for managing dependencies in frontend development with React.js. It allows developers to install, update, and manage packages necessary for building modern web applications
- PIP: Used for managing dependencies in backend development with Python. PIP (Python Package Index) is the standard package manager for Python and facilitates the installation and management of Python packages needed for backend functionalities.

5.4 SYSTEM COMPONENTS

5.4.1 Backend Side

We used Django framework [62], it was selected for its robustness, scalability, and security features, crucial for handling sensitive user data and ensuring efficient communication between frontend and deep learning model APIs.

JWT Authentication:

JSON Web Tokens (JWTs) are employed for secure user authentication and authorization:

- **Token Generation:** Upon successful authentication (login), the backend issues a JWT containing encoded user information and expiration details.
- **Token Verification:** For subsequent requests, the frontend includes the JWT in the Authorization header. The backend verifies the token's authenticity and validity before processing the request.
- **Session Management:** JWTs facilitate stateless session management, reducing server-side storage and enhancing scalability.

APIs:

The backend exposes RESTful APIs using Django REST Framework (DRF), facilitating seamless communication between the frontend and backend components of the application. Key API endpoints include:

- **Authentication APIs**: Handle user registration, login token issuance using JWT, and logout.
- **Media Detection APIs**: Facilitate requests for detecting AI-generated media content, integrating with deep learning model inference endpoints.
- **User Profile Management APIs**: Enable viewing, updating profiles, resetting passwords, activating email, viewing user history, and retrieving detected media.

Third-party Services:

- **Hugging Face**: Integration for model inference.

- **Database**: We used SQLite for storing user data and Django built-in models DBMS to manage user access and modification of the database.
- **Mailing System (SMTP)**: Used Google SMTP to send emails for account verification and password reset.

5.4.2 FRONTEND

The frontend of the AI-Generated Media Detection project is structured as a Single Page Application (SPA) leveraging React.jsx. This architectural choice enables the application to deliver a highly responsive and interactive user experience by updating the user interface dynamically without the need for full page reloads. React.jsx, renowned for its component-based development approach, facilitates modular and reusable code, enhancing maintainability and scalability across the front-end components. This architecture also supports seamless integration with backend services through RESTful APIs, ensuring efficient data exchange and synchronization between the user interface and server-side functionalities. Overall, the SPA design with React.jsx empowers the AI-Generated Media Detection project to deliver a smooth, engaging, and efficient user experience.

Key Pages and Features:

1. Sign Up Page:

1–Sign Up Page	
Purpose	User Registration: Designed to facilitate the creation of new user accounts within the application.
Key Features	Form Input: Provides fields for users to input essential details like name, email, username, country, age, gender, password, and password confirmation. • Validation: Implements validation checks to ensure data integrity and accuracy (e.g., email format, password strength). • Backend Integration: Utilizes `use-http` library to securely transmit user data to the backend (`BASE_DOMAIN_URL/users/register/`) for account creation. • Error Handling: Displays informative error messages to guide users through correcting any input errors or omissions. • User Interaction: Includes options for users to specify gender and age, along with a checkbox to remember preferences. Navigation: Provides a direct link to the Sign In page (sign-in) for existing users, promoting seamless transition between registration and login processes.

Table 5-6

2. User Profile Page:

2-UserProfile Page:	
Purpose	Primary Function: Allows users to view and manage their personal information and subscription details within the application.
Key Features	User Information Display: Shows user-specific data such as name, username, email, age, address, and profile image. Subscription Details: Displays current subscription plan, remaining attempts, start date, and end date. Activation Status: Alerts users if their account needs activation and provides a link to resend the activation email. Profile Management: Enables users to edit their profile information and update subscription plans as necessary.

Table 5-7

3. Edit Profile Page

3-Edit Profile Page:	
Purpose	Primary Function: Allows users to update their personal details, including name, email, password, and profile picture.
Key Features	User Information Update: Form inputs for modifying user details such as first name, last name, username, and email address. Password Update: Fields for entering and confirming a new password, along with verification against the current password. Profile Picture Management: Options to upload a new profile picture, delete the current picture, and preview the selected image.

Table 5-8

4. FAQs Page

4-FAQs Page:	
Purpose	Provides answers to common questions about the application.
Key Features	List of frequently asked questions and detailed answers to help users understand the system better.

Table 5-9

5. Terms of Use Page

5-Terms of Use Page:	
Purpose	Outlines the legal terms and conditions for using the application.
Key Features	Detailed documentation of user rights, responsibilities, and the legal framework governing the use of the application.

Table 5-10

6. Navbar

6-Navbar	
Purpose	Provides a dynamic navigation interface for seamless user interaction within the web application.
Key Features	Conditional Rendering: Adjusts content based on user authentication status. Navigation Links: Includes links to Home, Start, Pricing, About Us, and Contact Us sections. Authentication Management: Handles user login/logout functionality using local storage. Responsive Design: Utilizes Bootstrap for responsive and adaptive layout. Dark Mode Toggle: Allows users to switch between light and dark mode for personalized visual preferences. User Profile Integration: Displays user-specific information and profile image with direct links to profile-related pages

Table 5-11

7. Media Detector Page

7-MediaDetector page	
Purpose	Facilitates detection of media type (image, text, or audio) using AI prediction.
Key Features	Media Type Selection: Allows users to choose between Image, Text, or Audio for analysis. Dynamic UI Rendering: Renders different UI components based on the selected media type. Text Input: Provides a text area for users to input text for analysis. Image Upload: Supports drag-and-drop or file upload for image analysis (maximum size 10 Mb). Audio Recording: Integrates Audio Recorder Uploader component for audio analysis. Prediction Functionality: Initiates AI prediction based on the selected media type. Result Display: Shows the predicted result (AI or Human) based on the analysis. Backend Integration: Fetches data from the server using `fetch` API for media type prediction.

Table 5-12

8. Pricing Page

9-Pricing page	
Purpose	Displays different pricing plans (Basic, Professional, Enterprise) and allows users to toggle between yearly and monthly billing cycles.
Key Features	Toggle Buttons: Enables users to switch between Yearly and Monthly billing cycles. Plan Details: Shows detailed information for each plan including pricing and features. Selection Handling: Implements functionality to handle selection of a pricing plan. Responsive Design: Utilizes Bootstrap grid system for a responsive layout across different screen sizes. Animation: Integrates animations using AOS (Animate on Scroll) for a smoother user experience.

Table 5-13

5.4.2.2 User Interface Design

Design Approach:

- Tool: Figma was utilized for designing the UI/UX, ensuring precise visualization and collaboration among team members.
- Principles: The design focused on creating an intuitive, user-friendly interface with an emphasis on:
    - Ease of Navigation: Ensured users can seamlessly move between sections such as Home, Profile, Sign In, Sign Up, FAQs, and Terms of Use through a cohesive navigation bar.
    - Responsiveness: Designed to adapt smoothly across various devices and screen sizes, maintaining usability.
    - Aesthetic Appeal: Incorporated modern design elements to enhance visual appeal and user engagement.

UI Components:

- Navigation Bar:
    - Provides clear and accessible links to main sections, facilitating easy navigation throughout the application.
    - Designed using React Router to ensure smooth client-side routing without page reloads.
- Forms:
    - Implemented for critical functionalities like user authentication, profile management, and media detection requests.
    - Designed with user-centric principles, ensuring clarity, ease of input, and validation mechanisms to prevent erroneous submissions.
- Tables and Lists:
- Utilized to present structured data such as user detection history and relevant information, enhancing readability and organization.
- Implemented sorting and filtering options where applicable to improve data interaction.
- Modals and Alerts:
- Serve crucial roles in providing feedback to users:
    - Modals: Used for confirmation dialogs, ensuring clarity on critical actions before execution.
    - Alerts: Provide real-time notifications for form submissions, errors, and success messages, enhancing user feedback and interaction.

5.5 DEPLOYMENT

In this section, we document the deployment strategy used for the various components of our AI-generated media detection system.

5.5.1 Backend Deployment

We deployed the backend of our application on PythonAnywhere, a cloud-based hosting platform optimized for Python applications. PythonAnywhere allowed us to easily deploy our Flask/Django application with minimal configuration, providing a reliable and scalable environment for our backend services.

5.5.2 Frontend Deployment

We deployed the backend of our application on PythonAnywhere, a cloud-based hosting platform optimized for Python applications. PythonAnywhere allowed us to easily deploy our Flask/Django application with minimal configuration, providing a reliable and scalable environment for our backend services.

5.5.3 Model Deployment

The frontend of our system was deployed using GitHub Pages, a static site hosting service that integrates seamlessly with GitHub repositories. GitHub Pages enabled us to host our React application with continuous deployment, ensuring that any changes pushed to the master branch are automatically reflected in the live application.

5.6 CONCLUSION

In this chapter, we have detailed the implementation phase of the AI-Generated Media Detection project, covering the architecture, technologies, system components, and deployment strategies. This comprehensive overview highlights the transformation of theoretical concepts into a functional and user-friendly application. The next steps involve rigorous testing and validation to ensure the system's accuracy and reliability in real-world scenarios.

6. EVALUATION

6.1 EXPERIMENTAL SETUP

6.1.1 Hardware:

- **Cloud Environment:**
    - Platform: Kaggle Notebooks
    - GPUs: Tesla P100 and Tesla T4 x2

6.1.2 Software:

Machine Learning Models:

- **Text Model:**
    - Base Models: DeBERTa, RoBERTa
    - Method: Ensemble of fine-tuned DeBERTa and RoBERTa models
    - Final Layer: Concatenated outputs fed into a feed-forward layer for final prediction
- **Audio Model:**
    - Base Model: Wav2Vec 2.0
- **Image Model:**
    - Base Model: Vision Transformer (ViT)

6.1.3 Dataset:

Source:

AI-generated and human-created media from public repositories and social media platforms.

Preprocessing:

- **Text**: No preprocessing made on text.
- **Audio**: Feature extraction using Wav2Vec 2.0's built-in processing
- **Images**: Resized to 260x260 pixels and normalized.

6.1.4 Model Training:

Data Split:

- Training set: 70%
- Validation set: 20%

- Test set: 10%

Hyperparameters:

- Text Model:
  - Batch size: 16
  - Learning rate: 3e-5
  - Epochs: 2

- Audio Model:
  - Batch size: 32
  - Learning rate: 2e-5
  - Epochs: 10

- Image Model:
  - Batch size: 16
  - Learning rate: 5e-5
  - Epochs: 2.5

6.2 RESULTS

6.2.1 Performance Metrics:

- **Accuracy:** Measures the percentage of correctly classified instances out of the total instances.
- **Precision:** The ratio of true positive instances to the sum of true positive and false positive instances.
- **Recall:** The ratio of true positive instances to the sum of true positive and false negative instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a single metric that balances both concerns.
- **ROC-AUC:** Area Under the Receiver Operating Characteristic Curve, which measures the model's ability to distinguish between classes.

6.2.2 Results Summary:

Model	Accuacy	Precision	Recall	F1 Score	ROC-AUC
Text	98.30%	99.85%	96.78%	98.29%	98.32%
Audio	93.58%	93.60%	93.58%	93.35%	
Image	98.88%	98.88%	98.88%	98.88%	

Table 6-1

6.2.3 Visualization

Confusion Matrices:

Text Model:

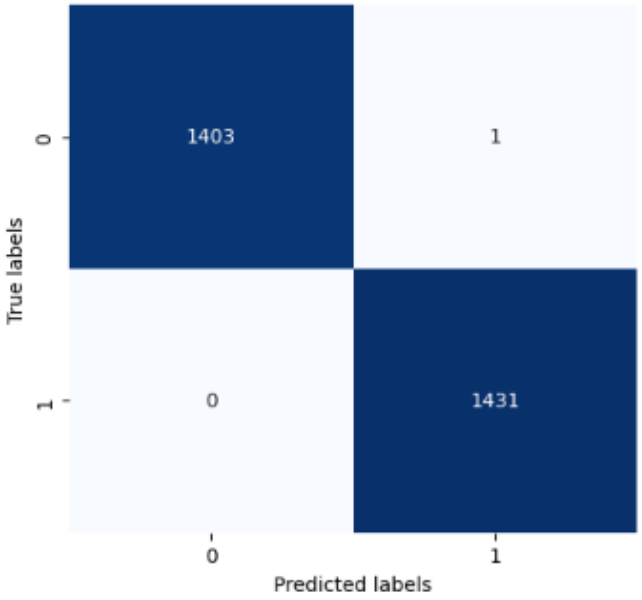


Figure 6.1

Audio Model:

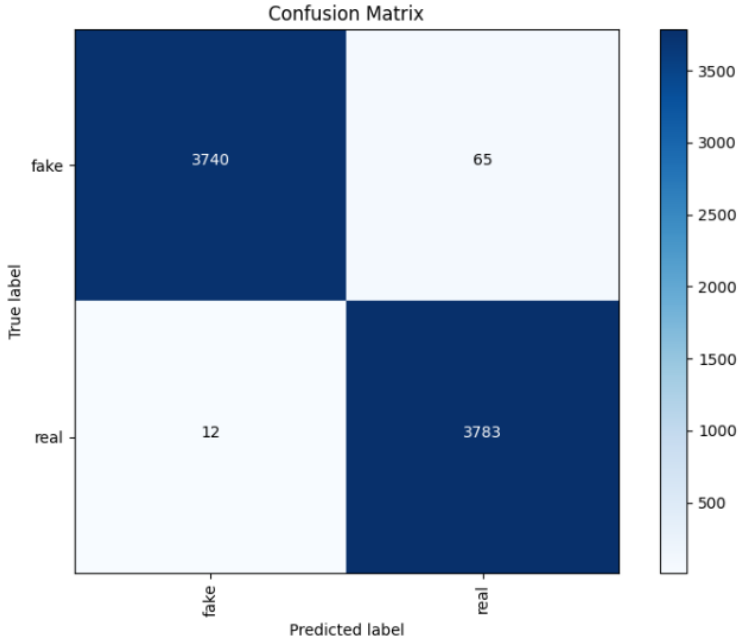


Figure 6.2

Image Model:

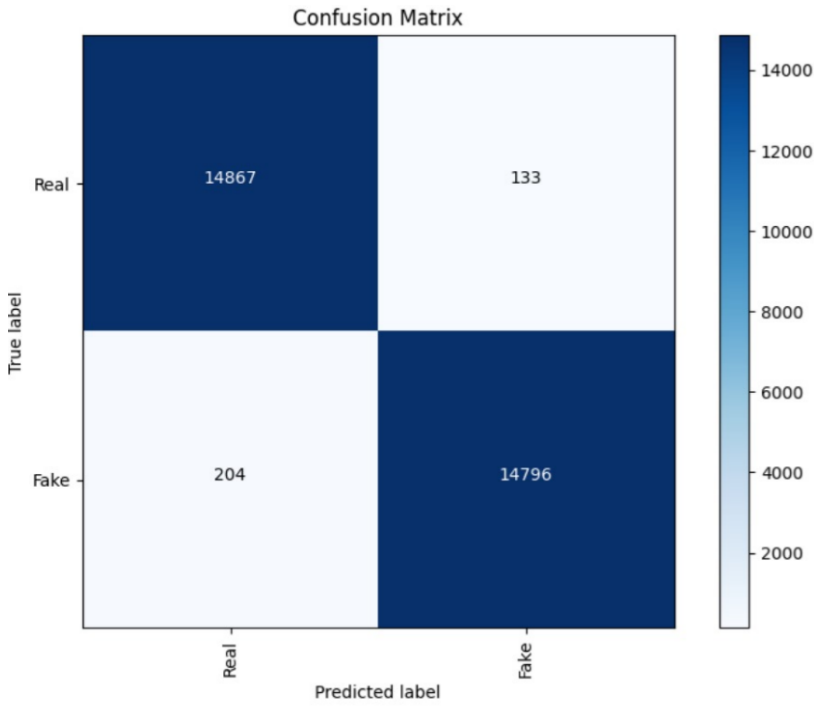


Figure 6.3

Curves Across the Models:

AUC-ROC curve for Audio Model

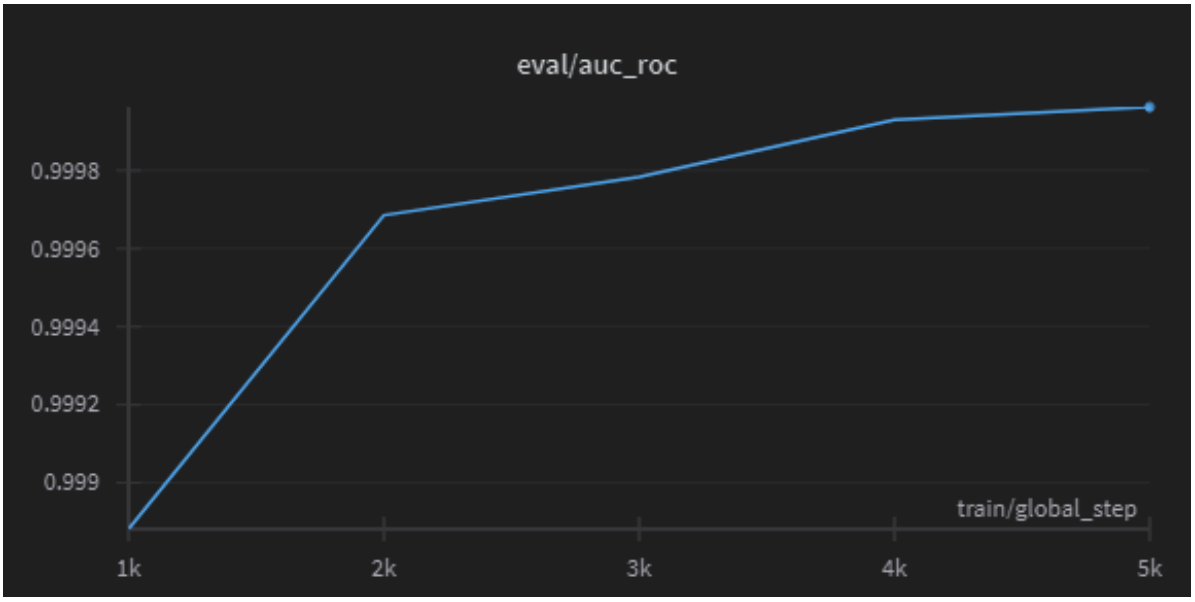


Figure 6.4

Training Loss for Text Model

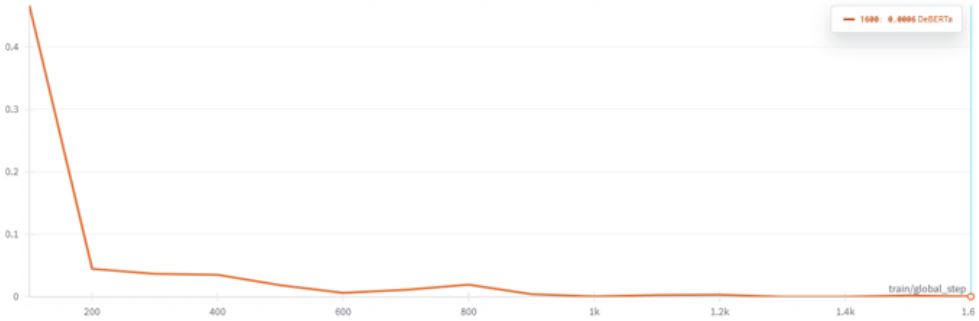


Figure 6.5

Training Loss for Audio Model

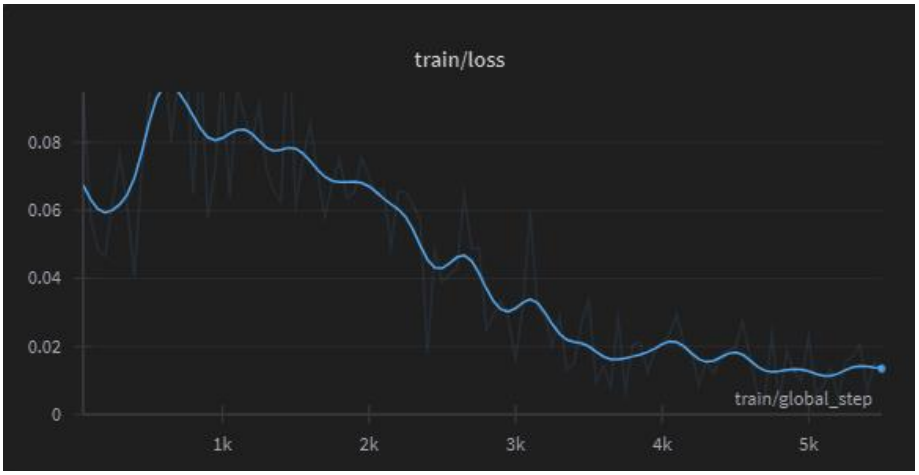


Figure 6.6

Learning rate on training Audio Model

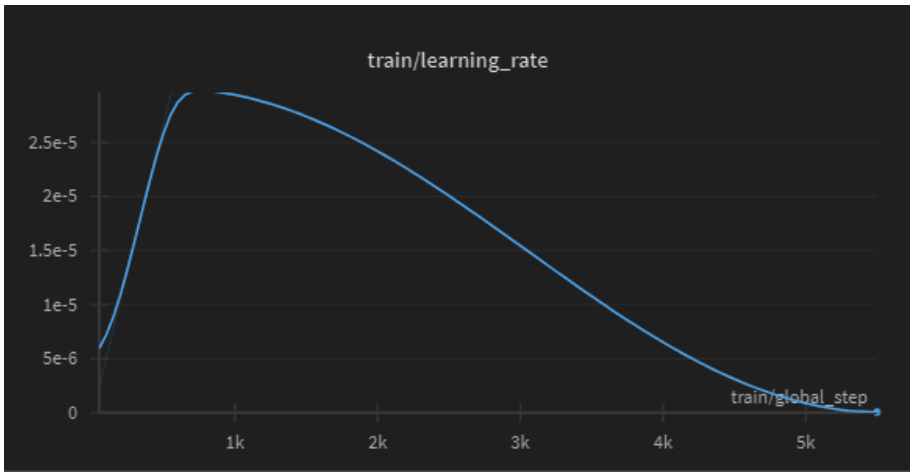


Figure 6.7

6.3 DICUSION

6.3.1 Successes:

- **High Accuracy Across Models:** Each model achieved over 90% accuracy, indicating strong performance in their respective domains.
- **Balanced Performance Metrics:** High precision, recall, and F1 scores across all models demonstrate their reliability in identifying true positives and minimizing false positives and negatives.
- **Generalization Capability:** High ROC-AUC scores indicate that the models generalize well to unseen data, maintaining robust performance across diverse datasets.

6.3.2 Limitations:

- **Data Imbalance:** Despite efforts to balance the dataset, certain categories, particularly within deepfake types, were underrepresented, which may affect the model's performance on these classes.
- **Computational Requirements:** Training and fine-tuning these models were resource-intensive, necessitating powerful GPUs and substantial memory, potentially limiting accessibility for deployment.
- **Complex Media Handling:** The models sometimes struggled with media that combined both AI-generated and human-created elements, leading to occasional misclassifications.

6.3.3 Potential Reasons for Performance:

- **Diverse and High-Quality Data:** The use of diverse and high-quality datasets contributed significantly to the models' ability to learn and distinguish complex patterns.
- **Advanced Model Architectures:** Leveraging state-of-the-art models like DeBERTa, RoBERTa, Wav2Vec 2.0, and ViT provided a strong foundation for handling the intricacies of text, audio, and image data.
- **Effective Hyperparameter Tuning:** Systematic hyperparameter tuning ensured that each model was optimized for its specific task, balancing various performance metrics.

6.3.4 Future Work:

- **Augmented Data Collection:** Enhancing the dataset with more examples, especially for underrepresented categories, to improve model robustness and accuracy.
- **Model Optimization:** Investigating model compression and optimization techniques to reduce computational overhead and facilitate deployment in resource-constrained environments.
- **Real-time Processing:** Developing and testing real-time detection capabilities to improve the system's practicality and usability in dynamic scenarios.

APPENDICES

APPENDIX A: DATA GRAPHS

A.1 Audio Data Distribution

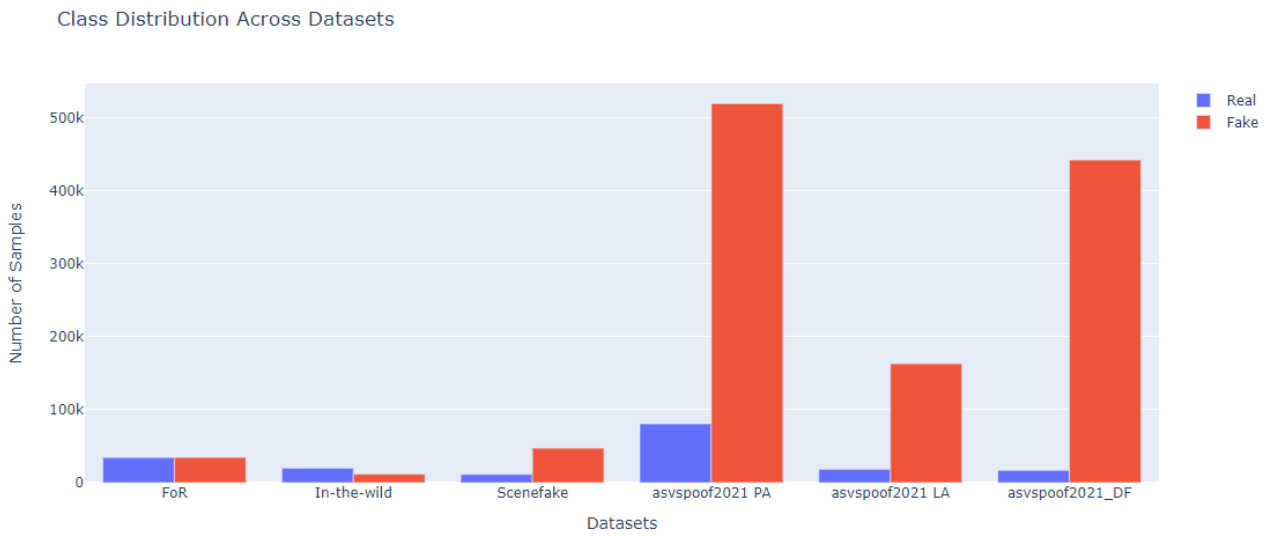


Figure A.1 Audio Data Distribution

A.2 Audio Duration Box Plot

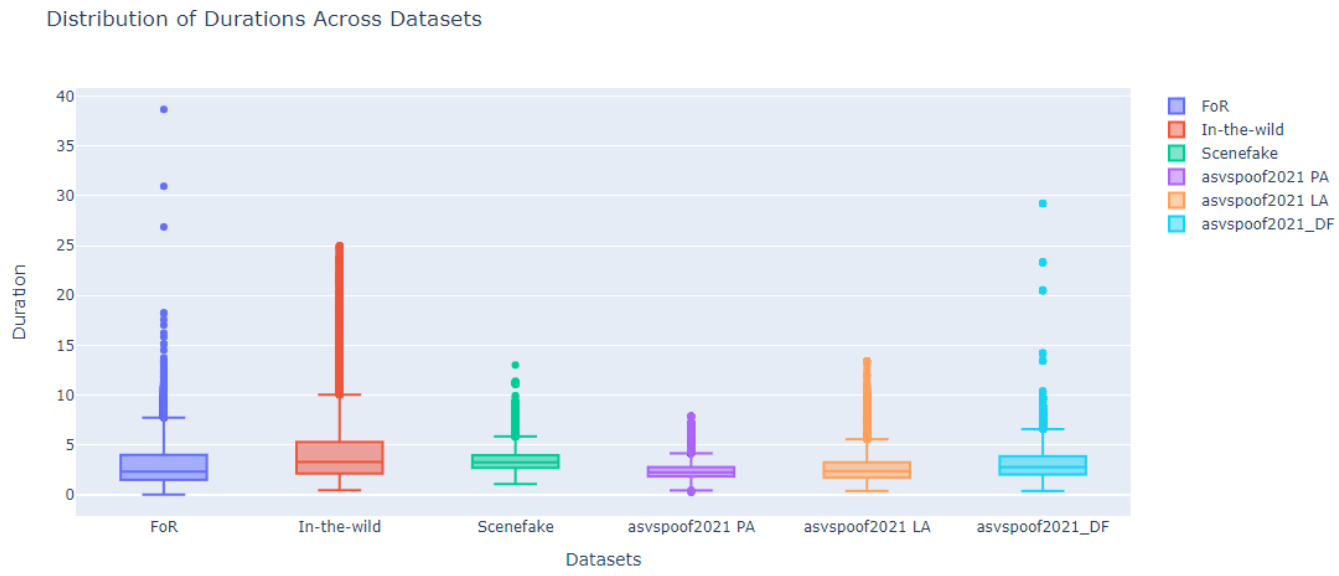


Figure A.2 Audio Duration Box plot

A.3 Audio File Format Distribution

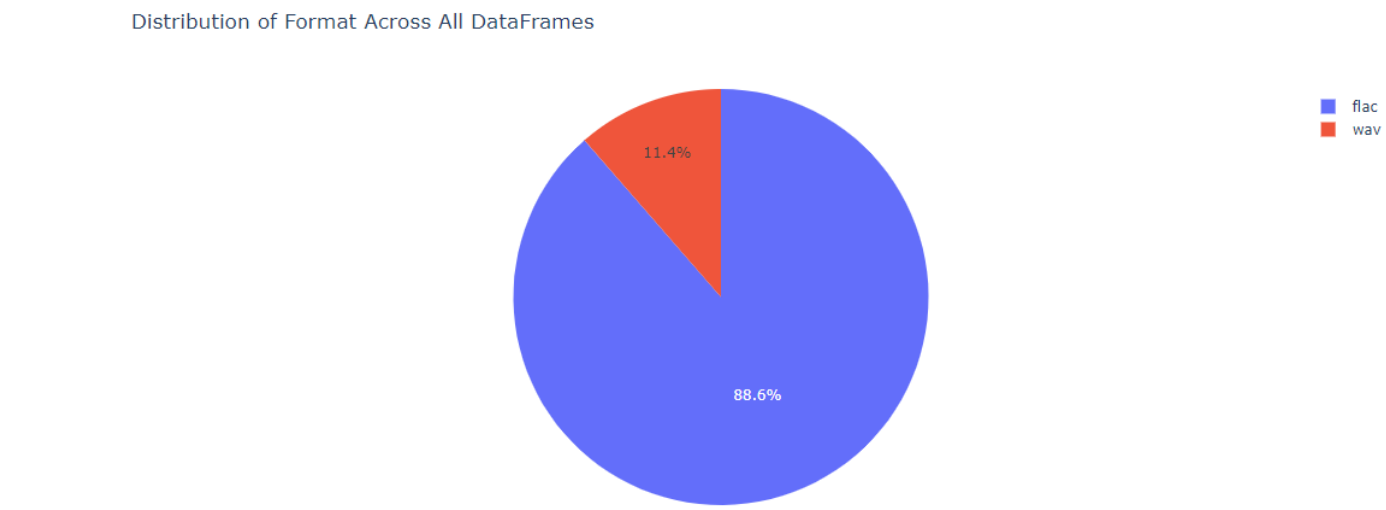
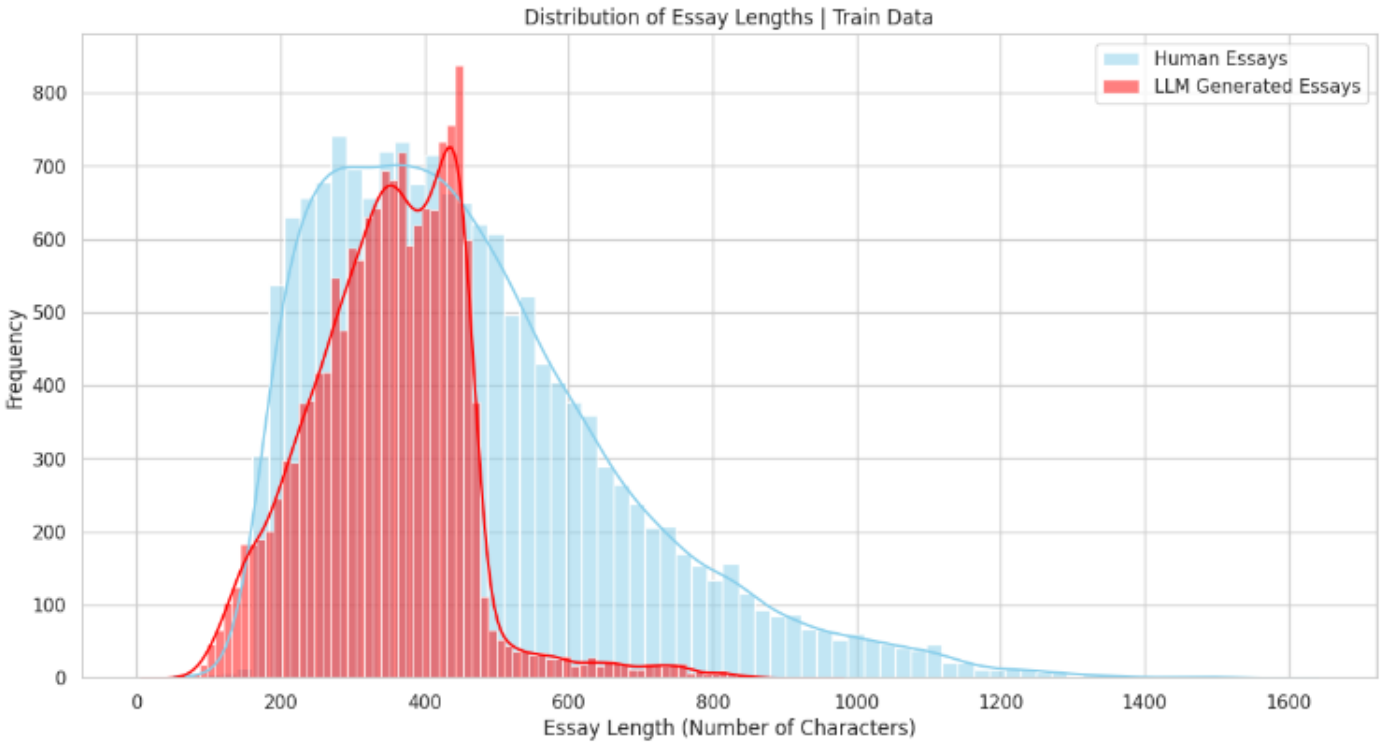


Figure A.3 Audio File Format Distribution

A.4 Histogram of Text Data length



FigureA.4 Histogram of Text Data length

A.4 Generated Text Data Distribution Across Models

Figure A.4 Generated Text Data Distribution

A.5 Human Made Essays Distribution

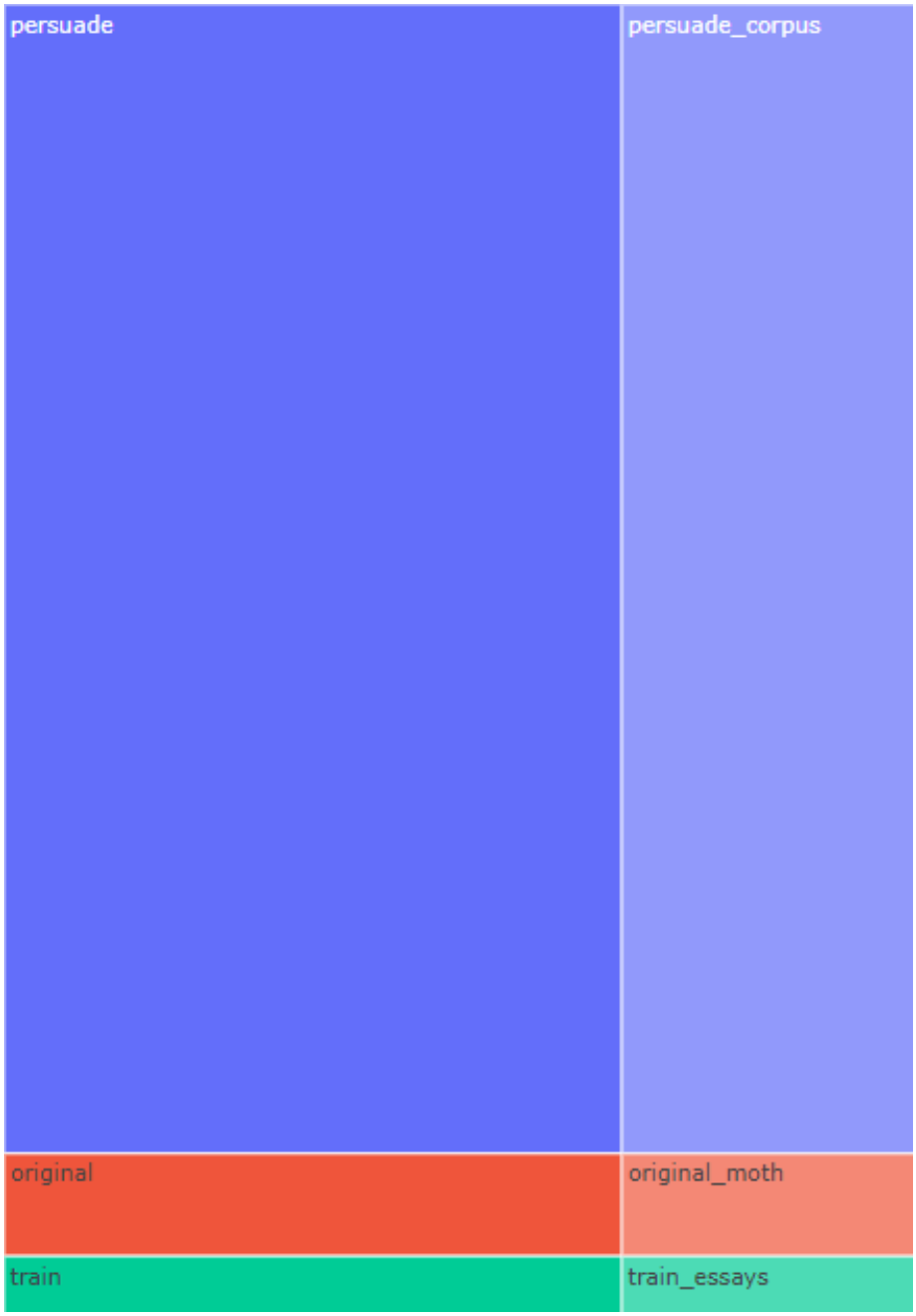


Figure A.5 Human Made Essays Distribution

A.6 Text Data Distribution Tabel

	count	unique	top	freq
text	31429	31429	Students would benefit from this a lot because...	1
source	29114	12	persuade_corpus	13312
category	31429	2	generated	15715

Figure A.6 Text Data Distribution

A.7 Text Data Box Plot

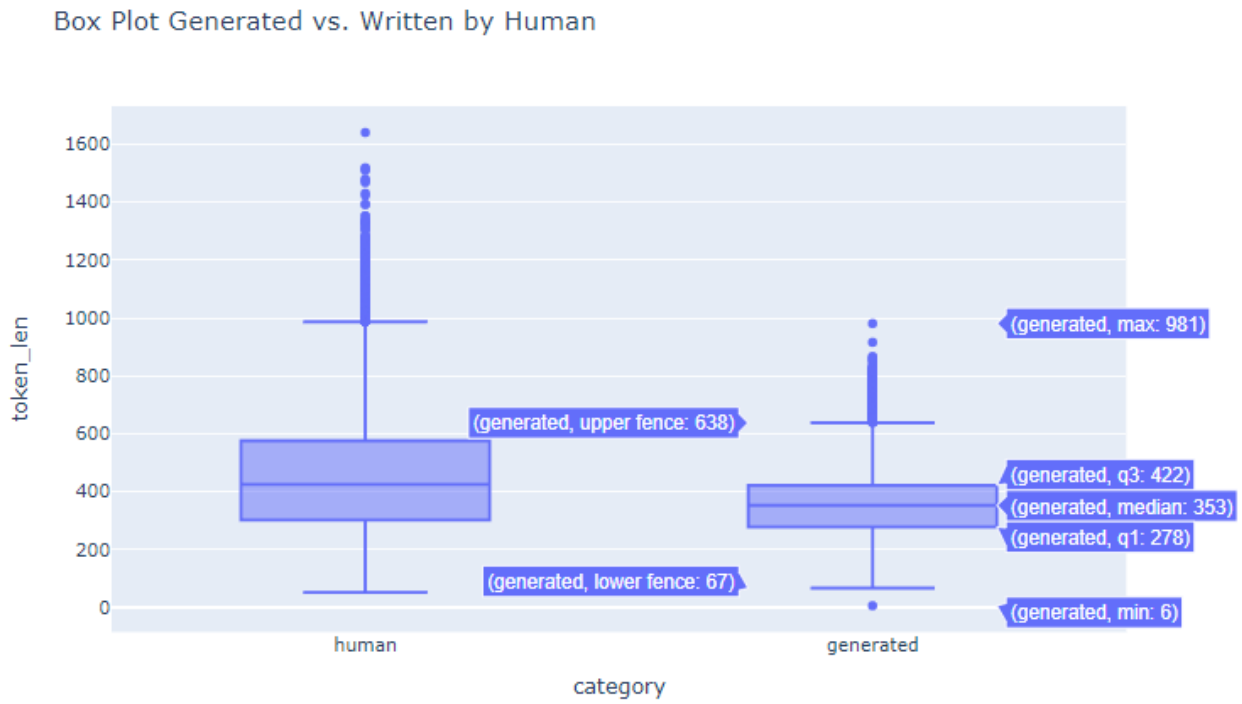


Figure A.7 Text Data Box

A.8 Most used Word in Human Essays

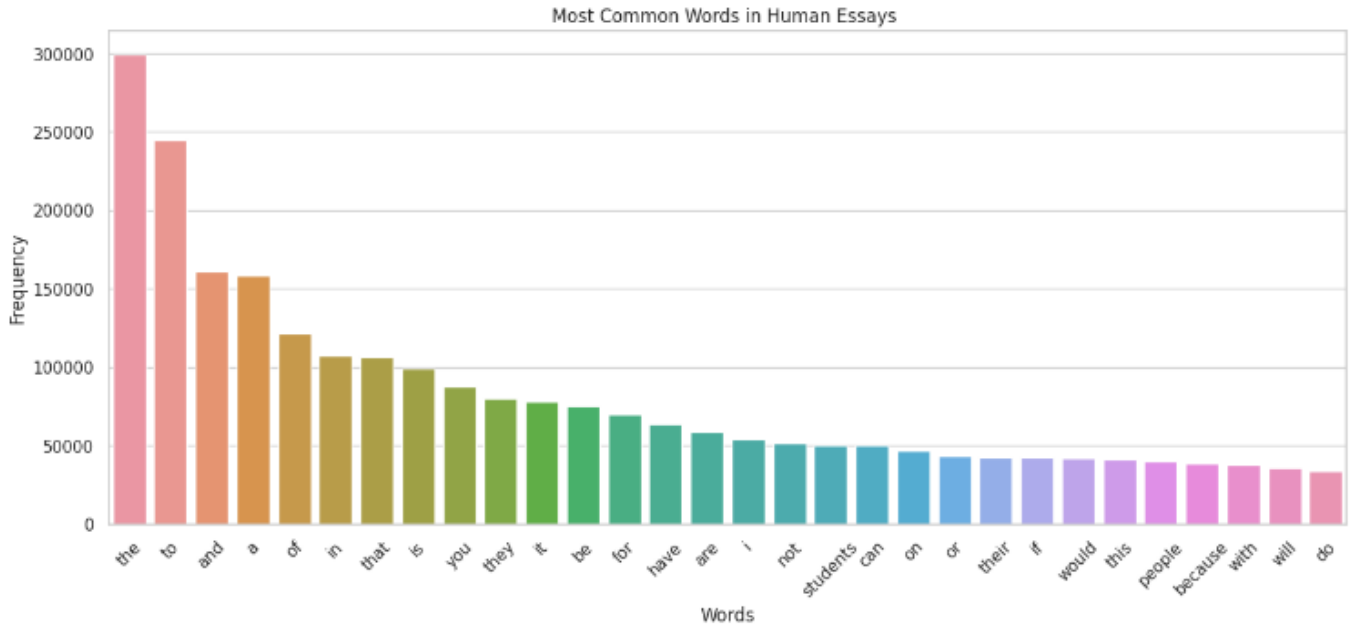


Figure A.8 Most used Word in Human Essays

APPENDIX B: DATA SAMPLES

B.1 Audio Data Samples

File Name	Duration	Sample Rate	Label	Format
C_07536_10_A	3.3680625	16000	fake	wav
C_06419_0_A	2.51575	16000	fake	wav
C_17869_5_A	2.3443125	16000	fake	wav
C_14606_05_A	2.468125	16000	fake	wav
C_08619_20_A	2.51675	16000	fake	wav
C_16896_0_A	2.1716875	16000	fake	wav
C_21398_5_A	2.9851875	16000	fake	wav
C_30328_0_D	3.48	16000	fake	wav
C_26338_0_B	5.82	16000	fake	wav
C_10619_10_A	3.4968125	16000	fake	wav
C_18038_5_A	4.262125	16000	fake	wav
C_11382_20_A	2.433875	16000	fake	wav
C_10683_0_A	4.220625	16000	fake	wav
C_20295_10_A	3.5375625	16000	fake	wav
C_25495_10_B	3.61	16000	fake	wav
C_23324_05_B	2.88	16000	fake	wav

Figure B.1 Audio Data Samples

B.2 Audio File Sample

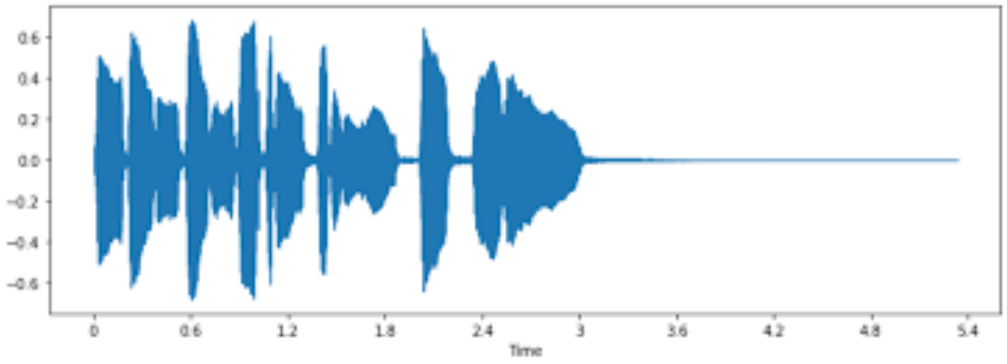


Figure B.2 Audio Sample

B.3 Image Real Data Samples

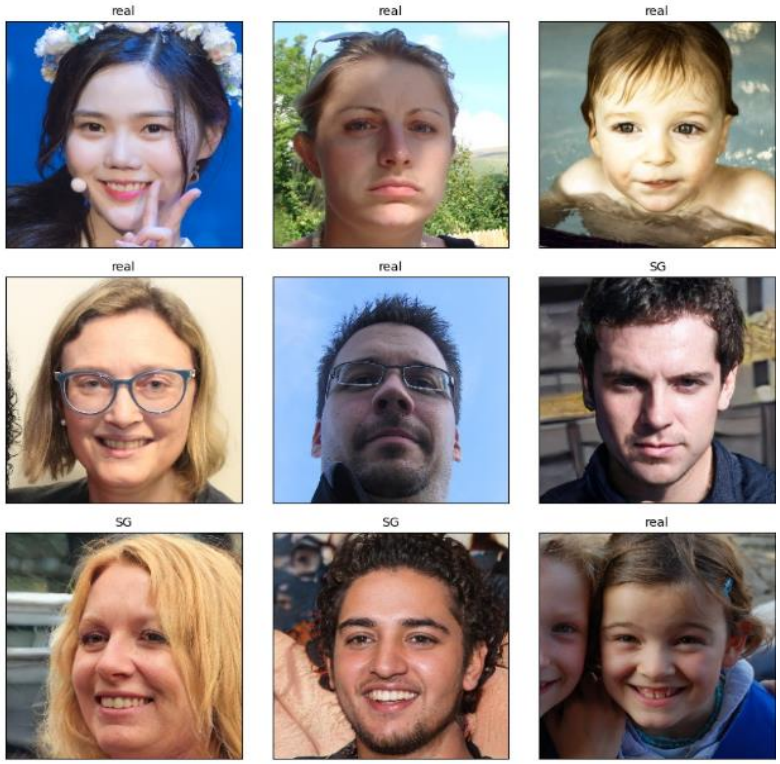


Figure B.3 Image Real Data Samples

B.4 Image Generated Data Samples

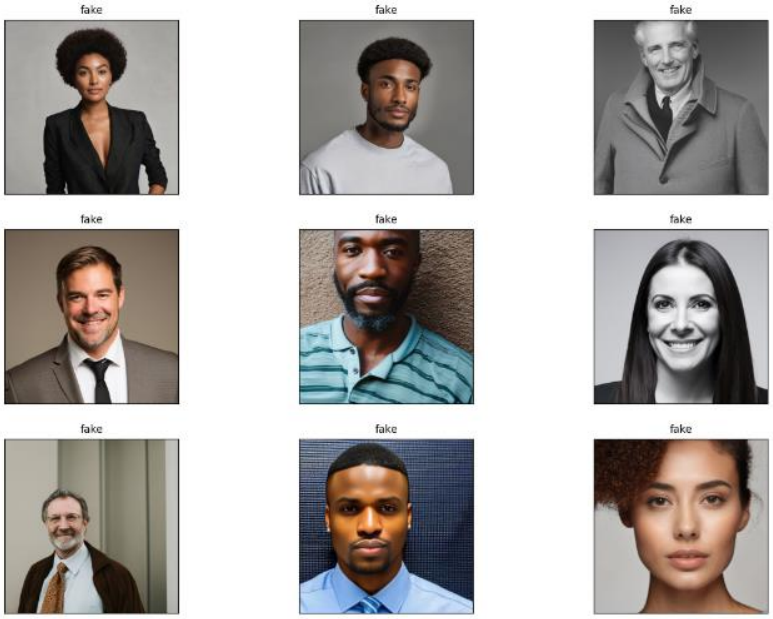


Figure Image Generated Data Samples

B.5 Text Token Length Box Plot

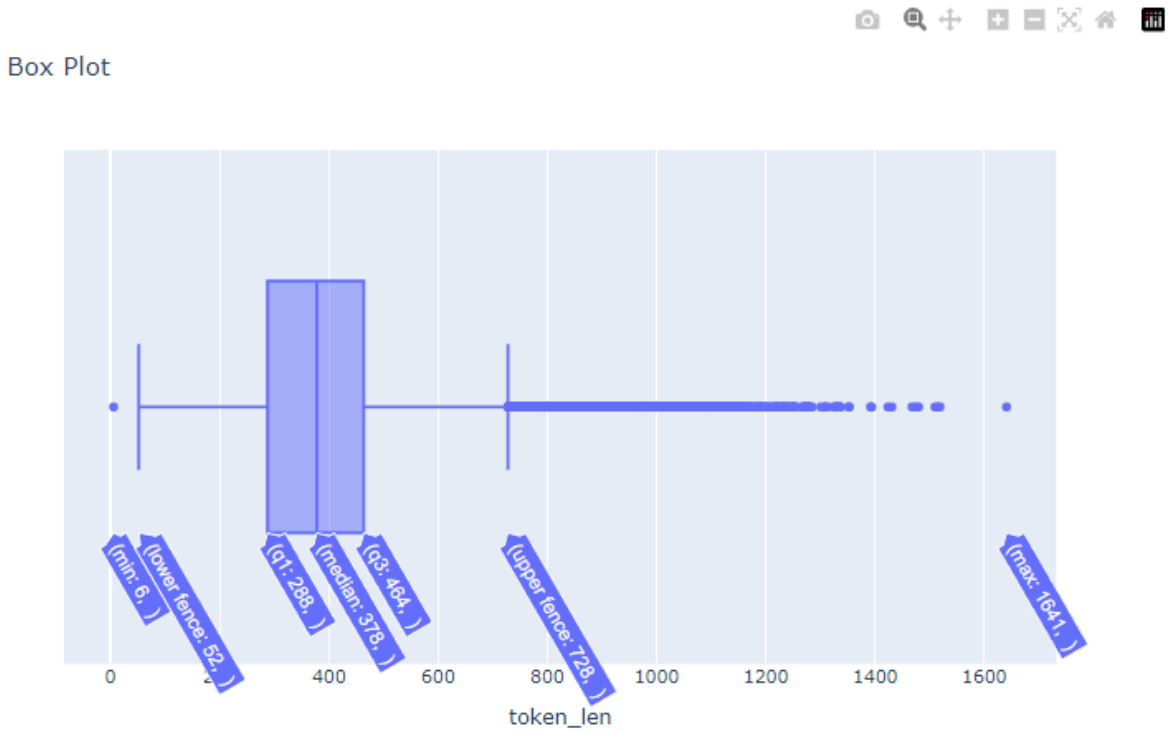


Figure B.5 Text Token Length Box Plot

APPENDIX C: MODELS ARCHITECTURES

C.1 WAV2VEC2.0 Model Architecture

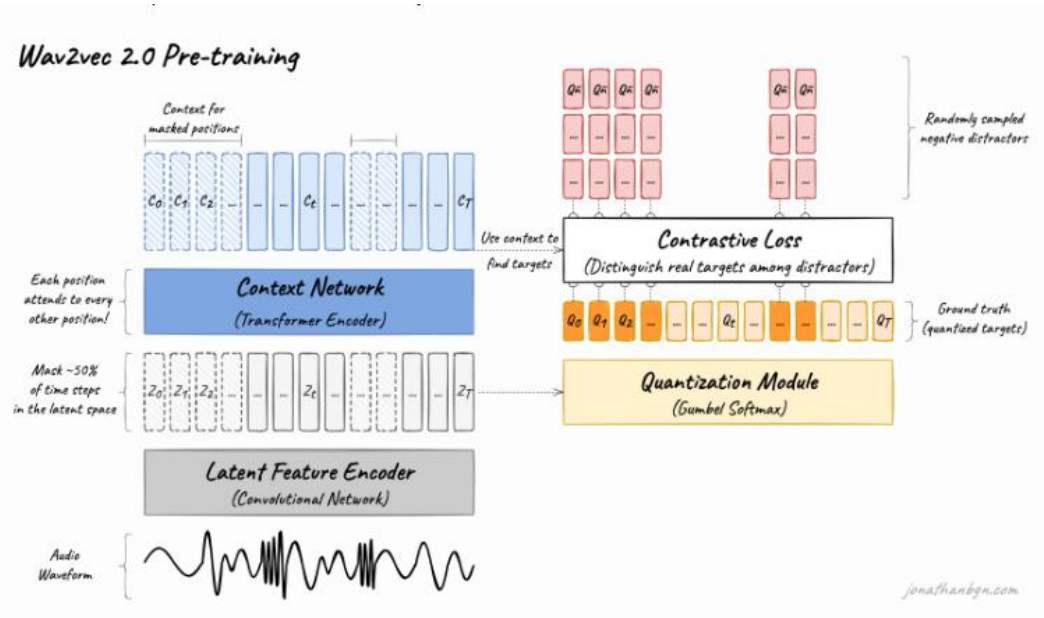


Figure C.1 WAV2VEC2.0 Architecture

C.2 BERT base Architecture

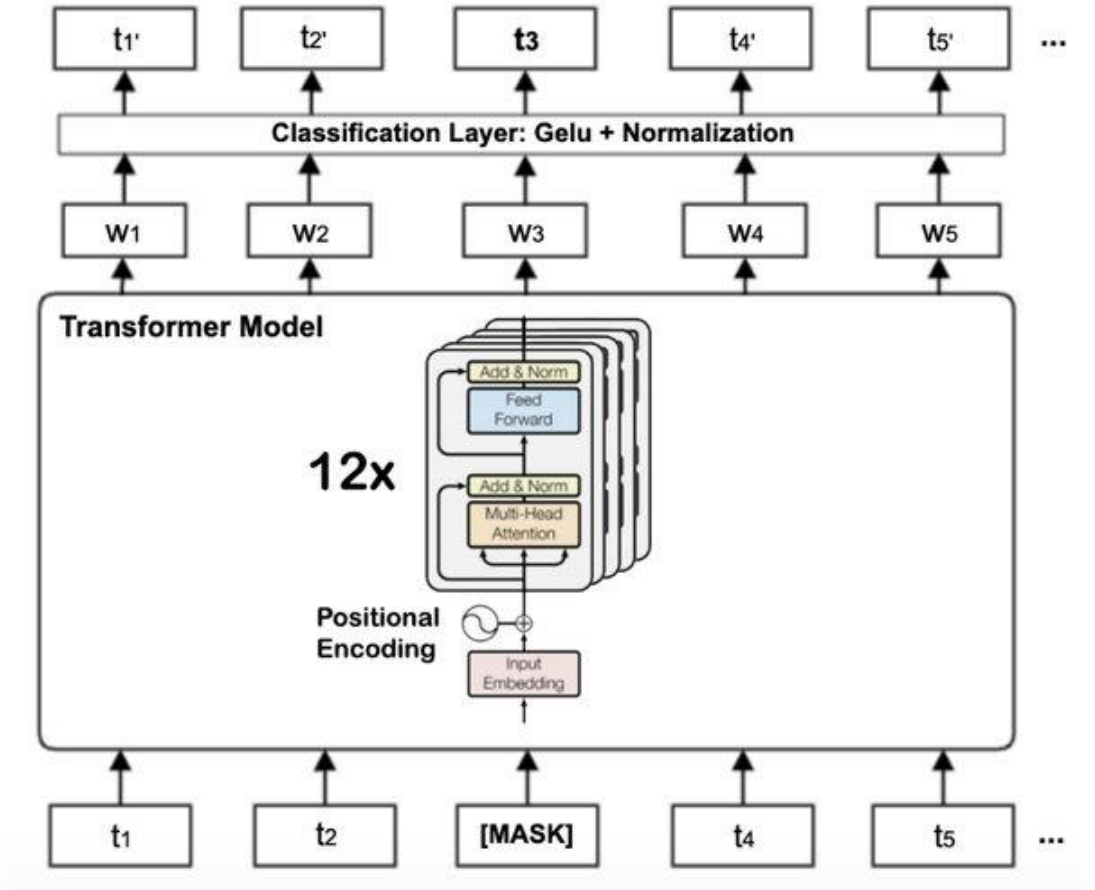


Figure C.2 BERT Architecture

C.3 RoBERTa Model Architecture

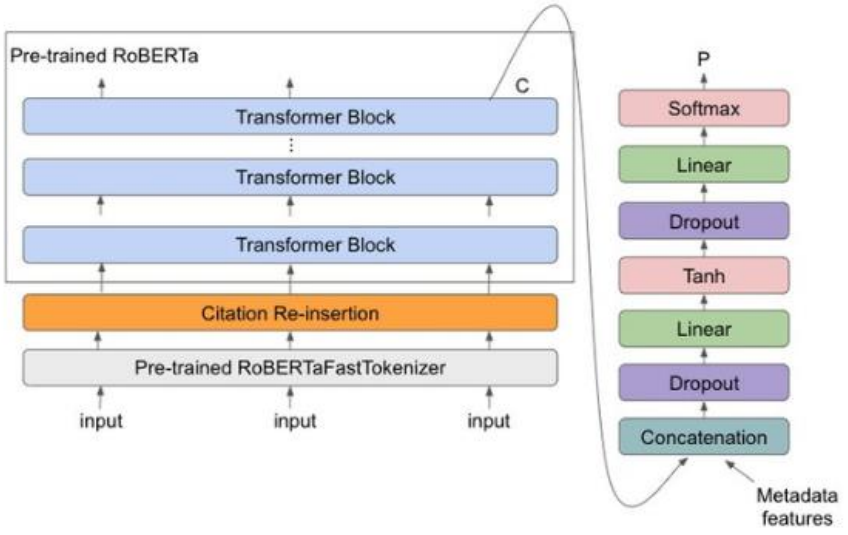


Figure C3 RoBERTa Model Architecture

C.4 DeBERTa Used Architecture

```
DebertaForSequenceClassification(
  (deberta): DebertaModel(
    (embeddings): DebertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=0)
      (LayerNorm): DebertaLayerNorm()
      (dropout): StableDropout()
    )
    (encoder): DebertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x DebertaLayer(
          (attention): DebertaAttention(
            (self): DisentangledSelfAttention(
              (in_proj): Linear(in_features=768, out_features=2304, bias=False)
              (pos_dropout): StableDropout()
              (pos_proj): Linear(in_features=768, out_features=768, bias=False)
              (pos_q_proj): Linear(in_features=768, out_features=768, bias=True)
              (dropout): StableDropout()
            )
            (output): DebertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): DebertaLayerNorm()
              (dropout): StableDropout()
            )
          )
          (intermediate): DebertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): DebertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): DebertaLayerNorm()
            (dropout): StableDropout()
          )
        )
      )
      (rel_embeddings): Embedding(1024, 768)
    )
  )
  (pooler): ContextPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): StableDropout()
  )
  (classifier): Linear(in_features=768, out_features=2, bias=True)
  (dropout): StableDropout()
)
```

Figure C.4 DeBERTa Architecture

C.5 BERT Vs RoBERTa Comparison

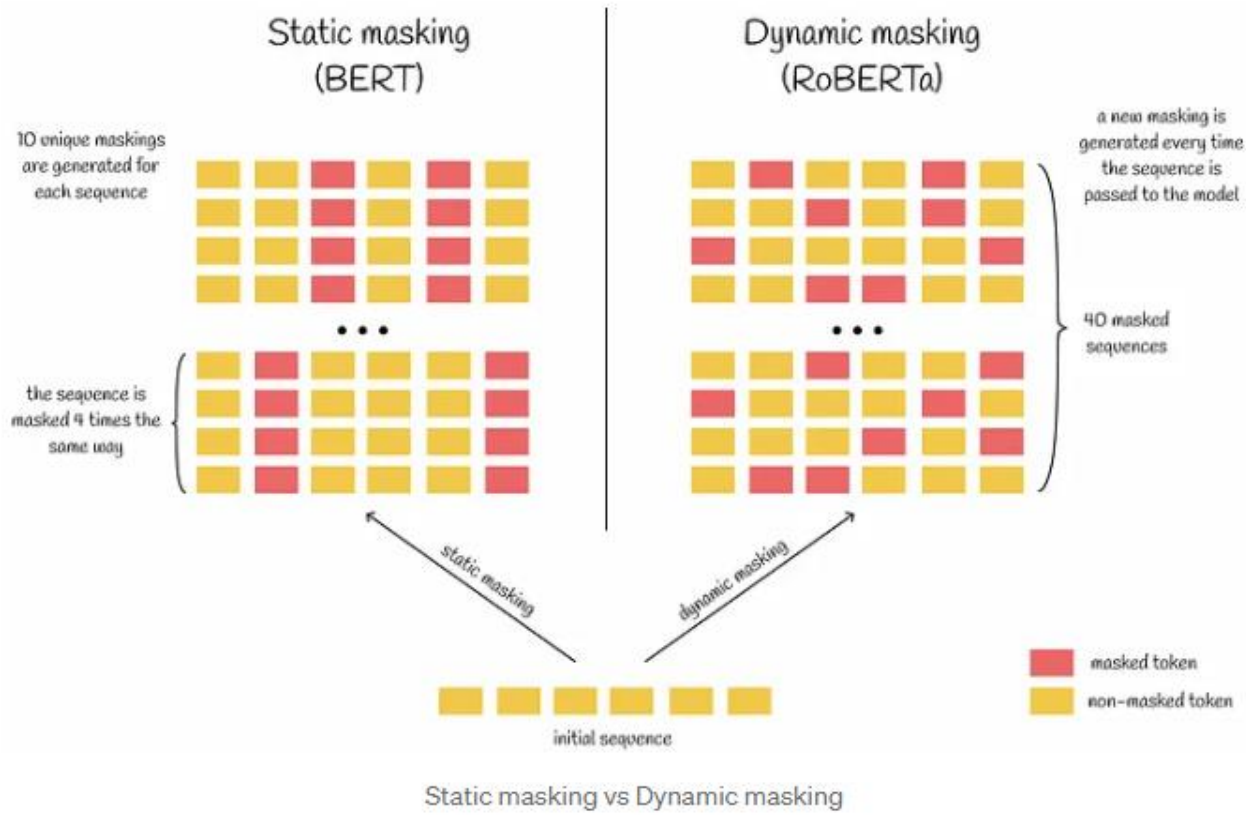


Figure C.5 BERT VS RoBERTa

C.6 ViT Model Architecture

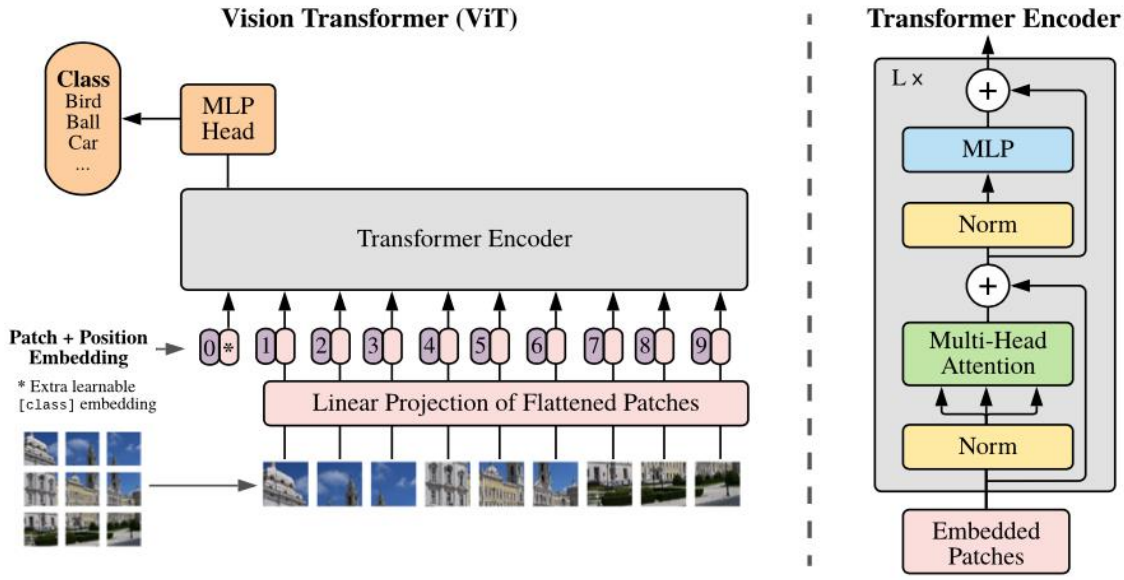


Figure C.6 ViT Model Architecture

C.7 Efficient Net Model Architecture

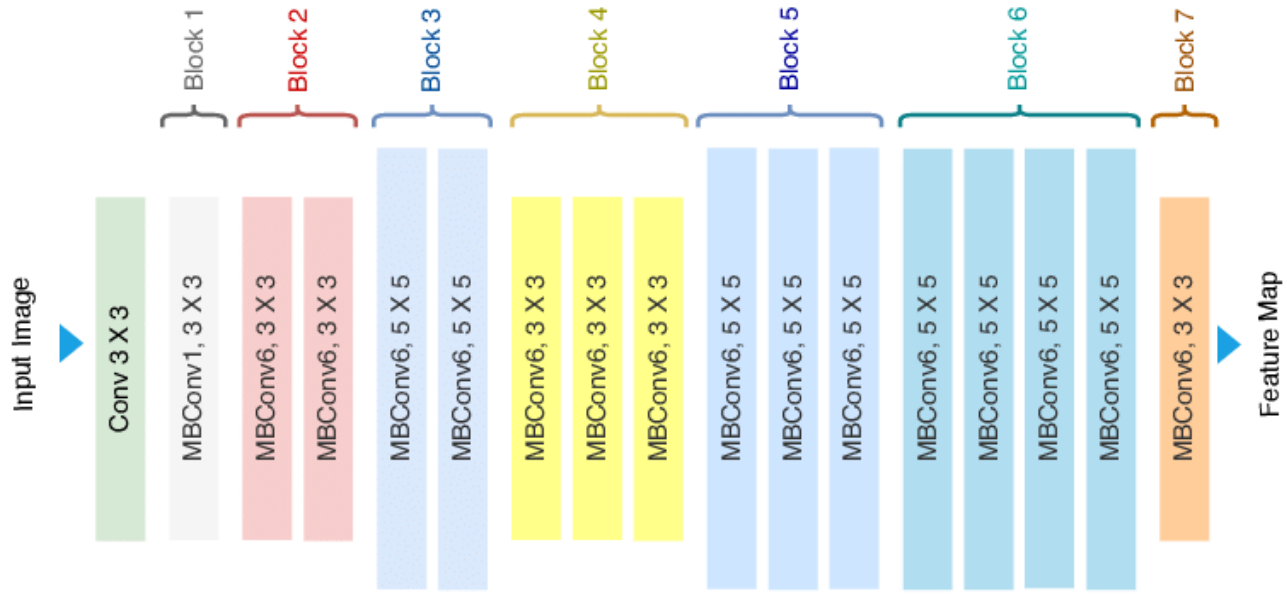


Figure 0C.7 Efficient Net Model Architecture

C.8 Final Text Model Illustration

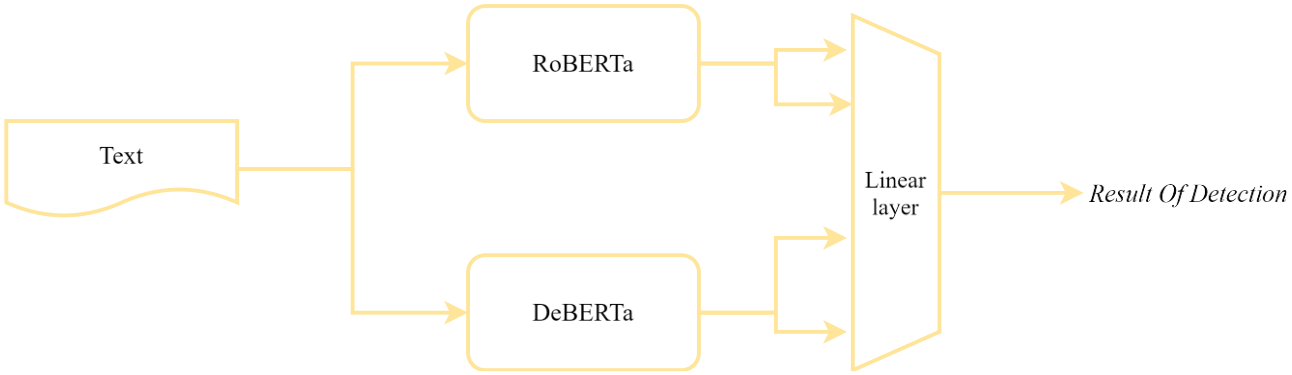


Figure C.8 Final Text Model Illustration

APPENDIX D: OUTPUT SAMPLES

D.1 Image Model Output Sample

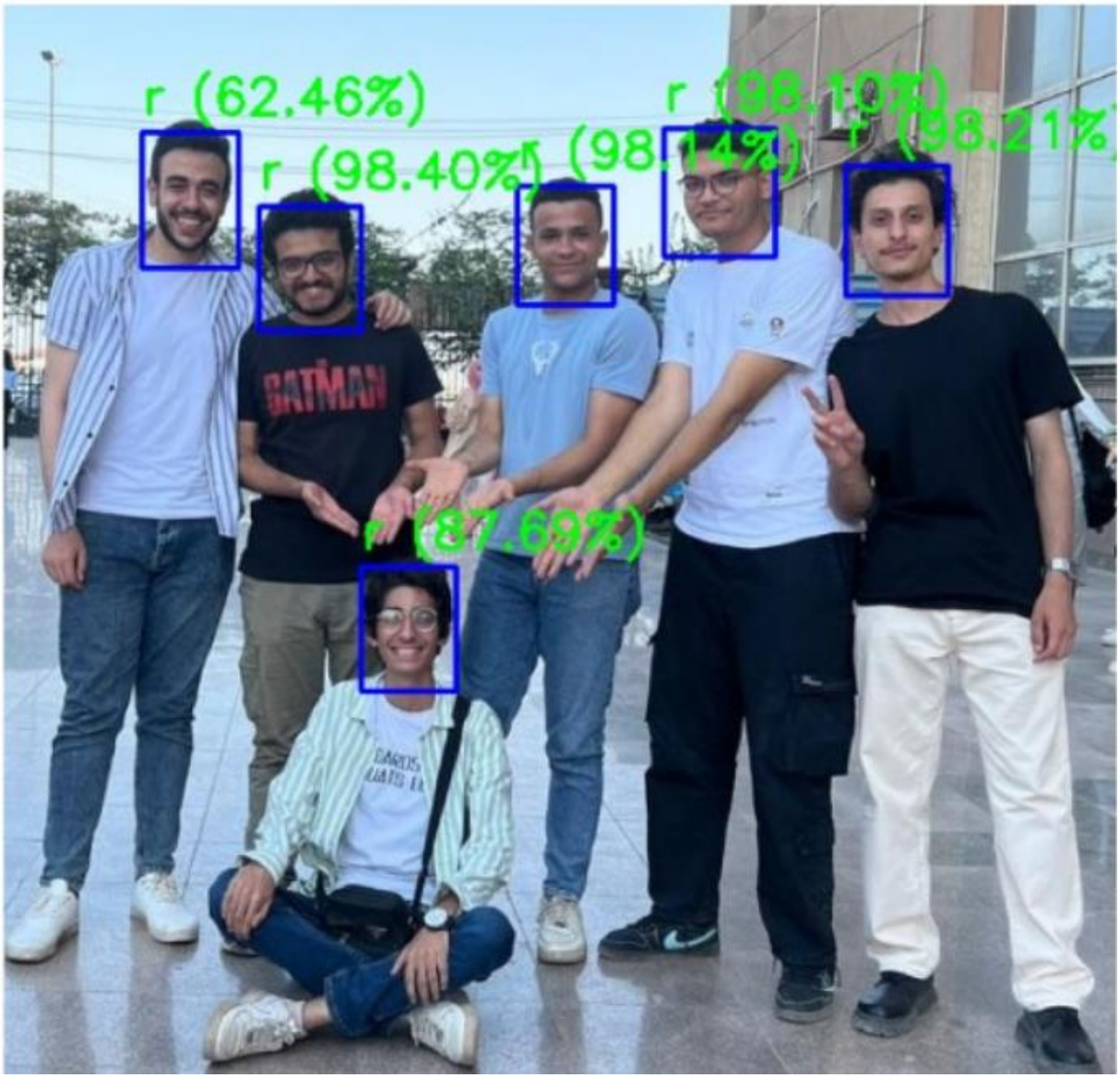


Figure D.1 Image Model Output Sample

D.2 Audio Model Output Sample

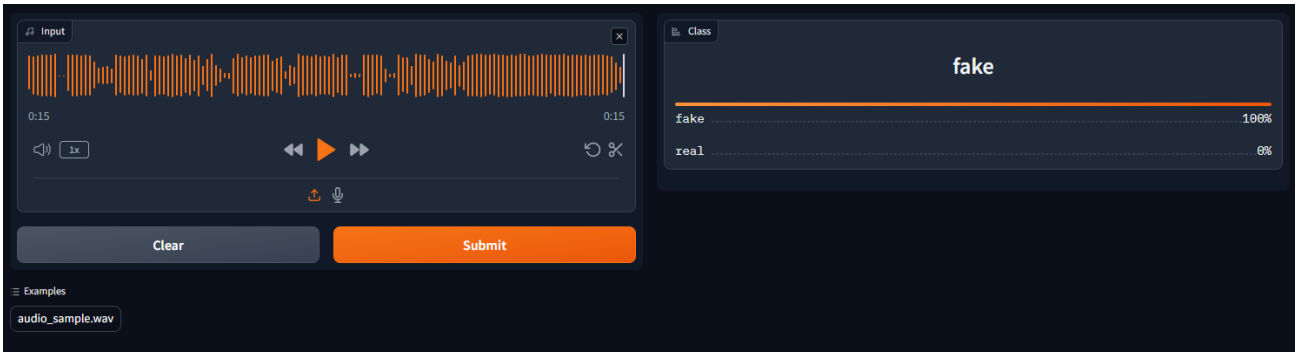


Figure D.2 Audio Output Sample

D.3 Text Model Heatmap Across All Attention Heads

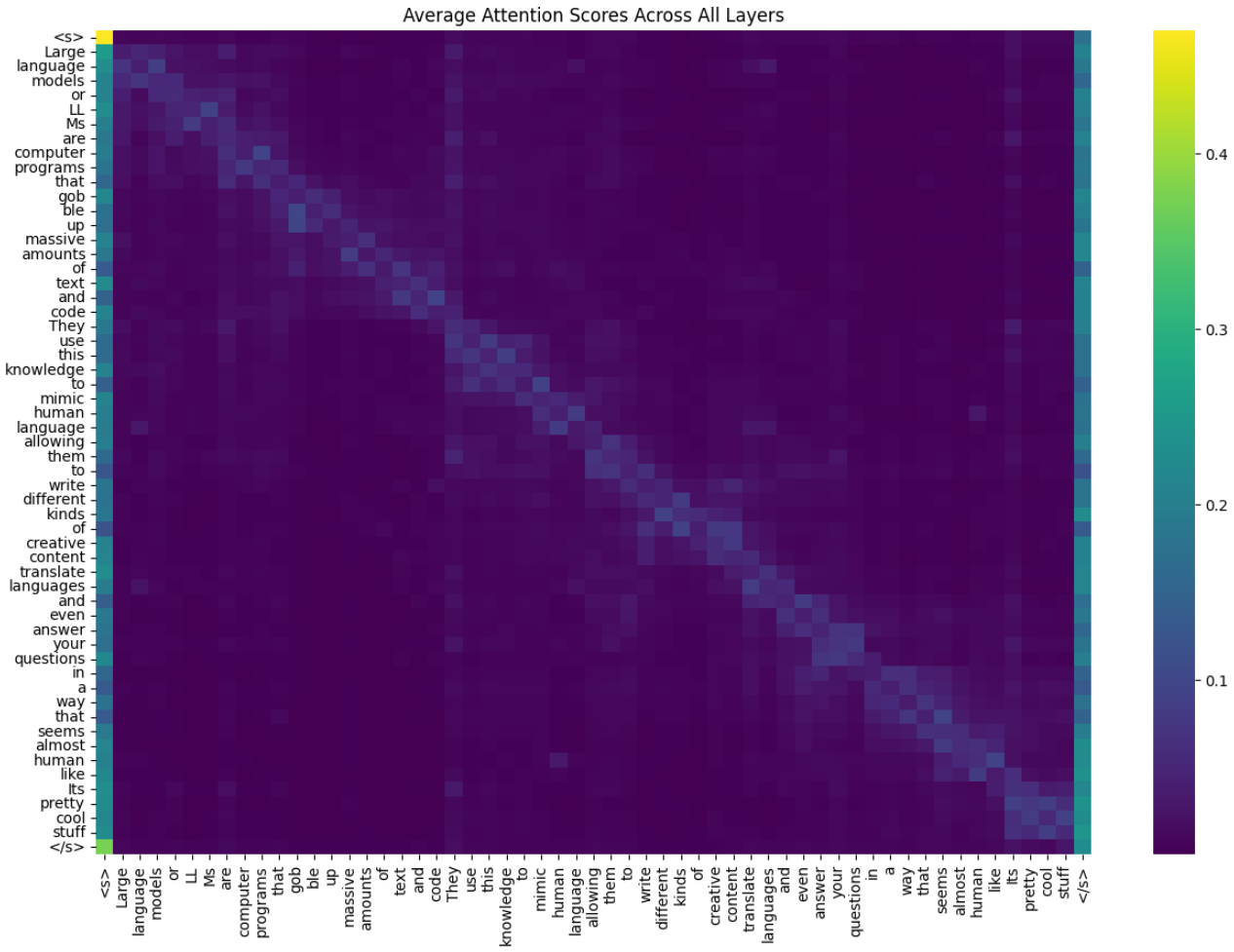


Figure D.3 Text Heatmap

D.4 Image Heatmap

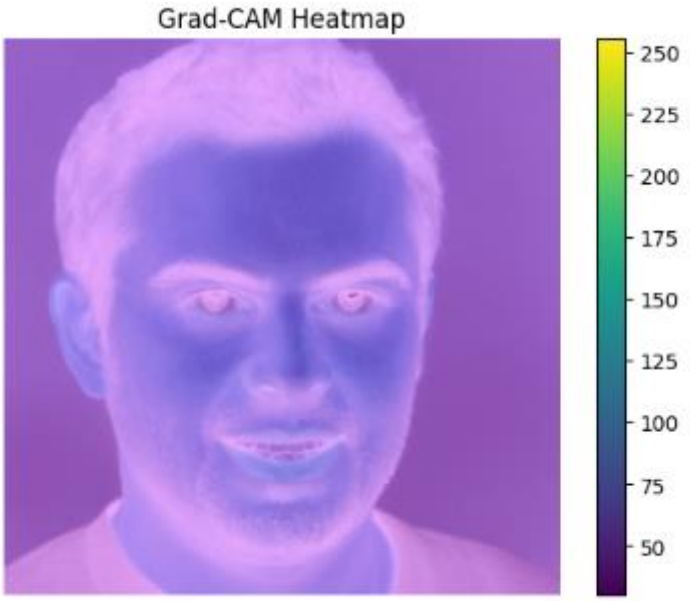


Figure D.4 Image Model Heat Map

APPENDIX E: CODE SNIPPETS

E.1 Text Query API Function

```
def api_huggingface(self,text, retries=5, wait_time=7):
        """
        Queries the Hugging Face API endpoints for DeBERTa and RoBERTa models.
        Parameters:
            retries (int): Number of retry attempts if errors occur during API query (default is 5).
            wait_time  (int): Time to wait (in seconds) between retry attempts (default is 7).
        Returns:
            tuple: Tuple containing the API responses for DeBERTa and RoBERTa models.
        """
        for _ in range(5):
            # DeBERTa
            deberta_text = str(TextCleaner(text))
            deberta = self.query({"inputs": deberta_text}, api_url=self.api_url_deberta)
            if 'error' in deberta and 'loading' in deberta['error']:
                time.sleep(wait_time)
            # RoBERTa
            roberta_text = str(TextCleaner(text, roberta_clean=True))
            roberta = self.query({"inputs": roberta_text}, api_url=self.api_url_roberta)
            if 'error' in roberta and 'loading' in roberta['error']:
                time.sleep(wait_time)

        return roberta, deberta
```

Figure E.1 Text Model API

E.2 Get Output for Text Script

```python
from DAIGT import DAIGT
token = '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' # Temporary token for testing
ff_path = r'D:\Graduation-project\Models\Text Detector Model\Second Term\Final Model Script\model_scripted.pt'  # set Your Path

API_URL_DeBERTa = "https://api-inference.huggingface.co/models/zeyadusf/deberta-DAIGT-MODELS"
API_URL_RoBERTa = "https://api-inference.huggingface.co/models/zeyadusf/roberta-DAIGT-kaggle"


daigt = DAIGT(token, API_URL_DeBERTa, API_URL_RoBERTa, ff_path)

def detect_text(text):
    output = daigt.detect_text(text)

    if output >= 0.5 :
        return f'This Text is AI-Generated'
    else :
        return f'This Text is Human-Written'
```

Figure E.2 Text Get Results

E.3 Audio Model Get Results API

```python
    def query_huggingface_api(self, file_path: str, retries: int = 5, wait_time: int = 7) -> List[Dict[str, float]]:
        headers = {
            "Authorization": f"Bearer {self.api_token}"
        }
        with open(file_path, 'rb') as f:
            data = f.read()

        for attempt in range(retries):
            response = requests.post(self.api_url, headers=headers, data=data)
            result = response.json()

            if 'error' in result and 'loading' in result['error']:
                print(f"Model is loading. Retrying in {wait_time} seconds...")
                time.sleep(wait_time)
            else:
                return result
        raise RuntimeError(f"Failed to get a response from the model after {retries} retries.")

    def get_label_with_max_score(self, response: List[Dict[str, float]]) -> str:
        return max(response, key=lambda x: x['score'])['label']

    def process_and_classify(self, input_path: str) -> str:
        try:
            output_wav = os.path.join('/tmp', f"{os.path.splitext(os.path.basename(input_path))[0]}.wav")

            # Convert the file to WAV format
            self.convert_to_wav(input_path, output_wav)

            # Query the Hugging Face API
            response = self.query_huggingface_api(output_wav)
            print(f"API Response: {response}")

            # Get the label with the highest score
            label = self.get_label_with_max_score(response)
            print(f"Label with max score: {label}")
            return label

        except Exception as e:
            print(f"An error occurred: {e}")
            return ""
```

Figure E.3 Audio Model API

E.4 Audio Model Get Results Script

```
api_url = "https://api-inference.huggingface.co/models/motheecreator/wav2vec2-base-finetuned-finetuned"
api_token = "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"

processor = AudioProcessor(api_url, api_token)


input_file = 'input file path'  # Update with your input file path

label = processor.process_and_classify(input_file)
print(f"Final Label: {label}")
```

Figure E.4 Audio Get Results Script

E.5 The Verification Function for Loging in

```
handleFormSubmit = async (event) => {
        event.preventDefault();

        const { username, password } = this.state;

        // Example URL for login API
        const loginUrl = `${BASE_DOMAIN_URL}/users/login/`;

        try {
            // Example POST request using fetch
            const response = await fetch(loginUrl, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ username, password }),
            });

            if (!response.ok) {
                throw new Error('Login failed.');
            }

            const data = await response.json();
            console.log('Login successful!', data);

            // Example handling of successful login (redirect, etc.)
            localStorage.setItem('token', data.token);
            window.location.href = "/"; // Redirect to home page
        } catch (error) {
            console.error('Error logging in:', error);

            // Example handling of incorrect password (change input field style)
            if (error.message === 'Login failed.') {
                document.getElementById('password').style.borderColor = 'red';
                document.getElementById('password').placeholder = 'Wrong Password';
            }
        }
    };
```

Figure E.5 Log in verification

E.6 Reset Password Method

```
const fetchResetPassword = async (new_password) => {
    const body = JSON.stringify({ new_password, email });
    const url = BASE_DOMAIN_URL + '/users/reset_password/';
    console.log(body)
    await fetch(url, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`,
        },
        body: body
    }).then(response => {
        if (response.ok) {
            return response.json();
        } else {
            throw new Error('Request failed!');
        }
    }).then(data => {
        if (data.status === 1) {
            // remove token from local storage
            navigate('/SuccessfullyResetPassword');
            localStorage.removeItem('token');
        } else {
            console.log(data.message);
        }
    }).catch(error => {
        console.error("Error:", error);
    });
}
```

Figure E.6 Reset Password

APPENDIX F: FRAMES FROM THE FRONTEND

F.1 Home Page in Light Mode

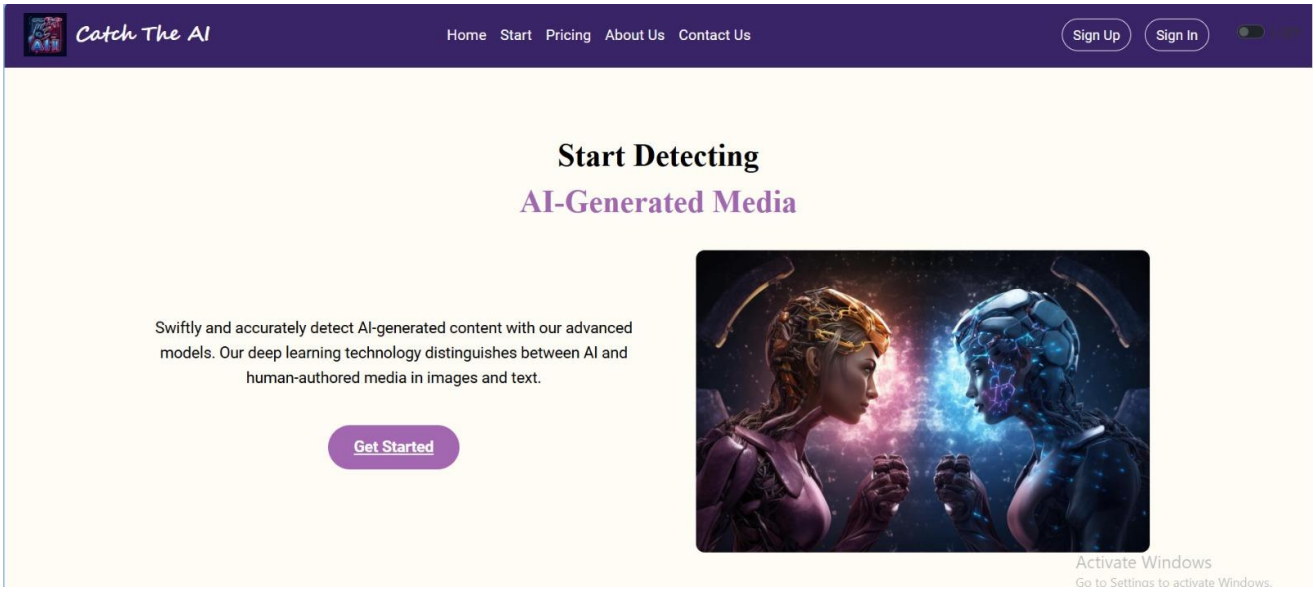


Figure F.1 Light Mode

F.2 Home Page in Dark Mode

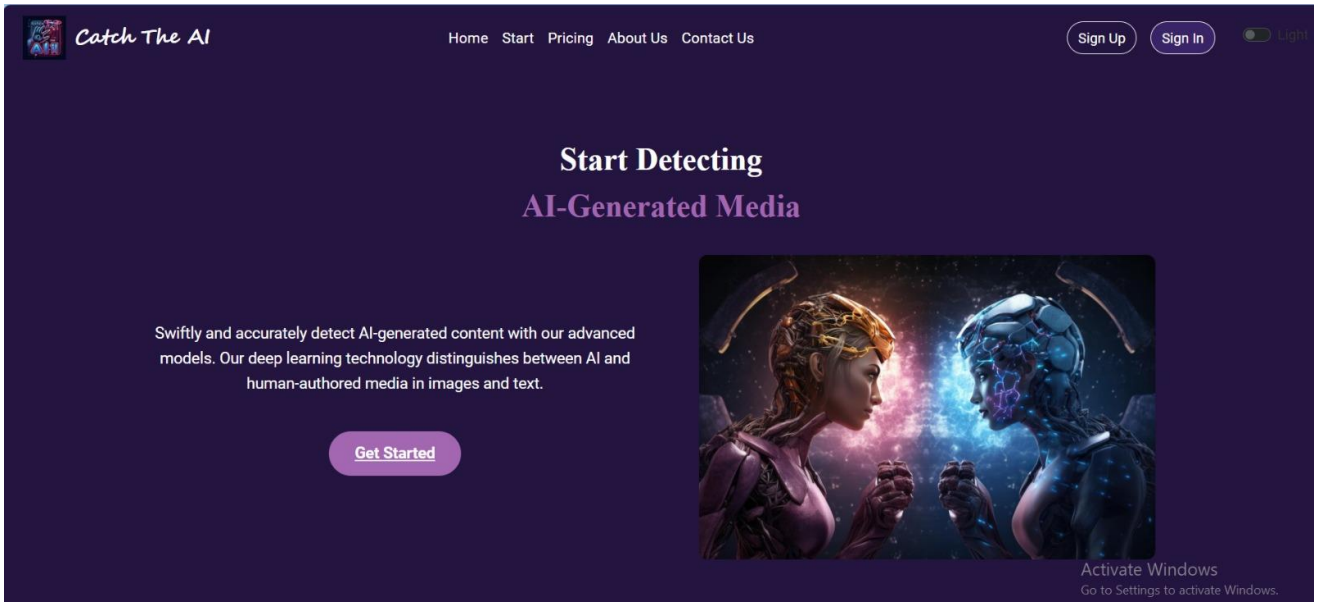


Figure F.2 Dark Mode

F.3 Select Media Page

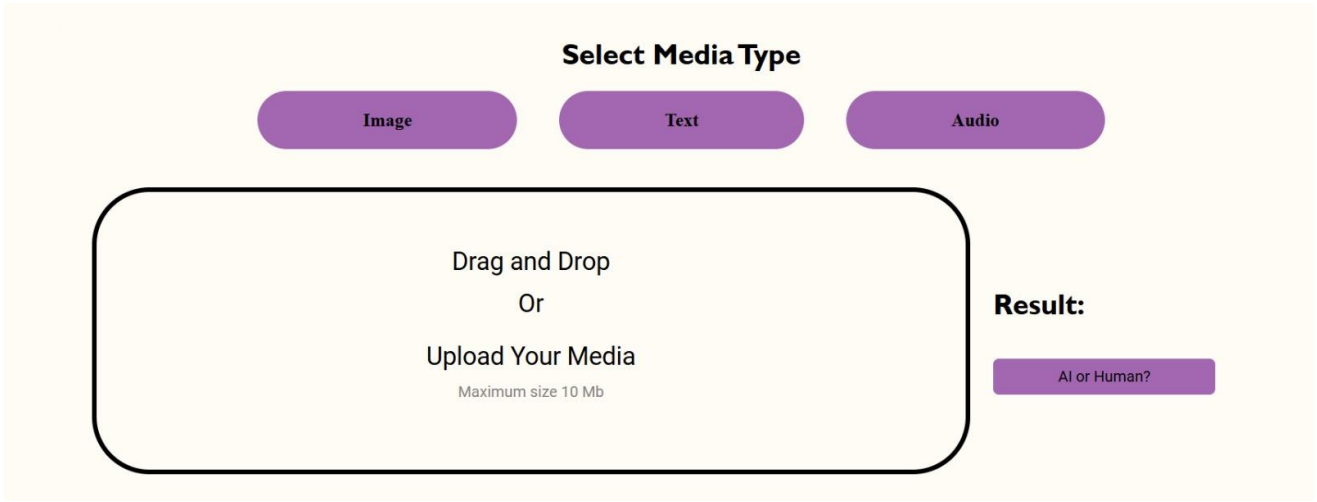


Figure F.3 Select Media

F.4 Contact Us

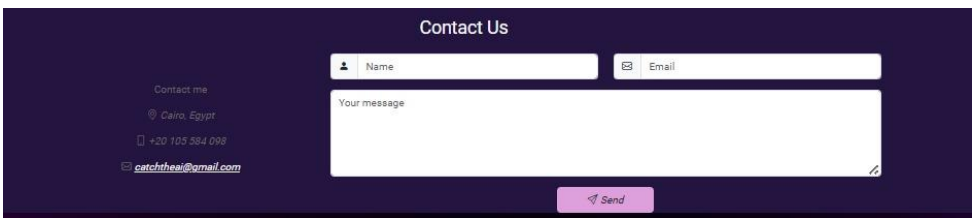


Figure F.4 Contact Us

F.5 Sign Up Page

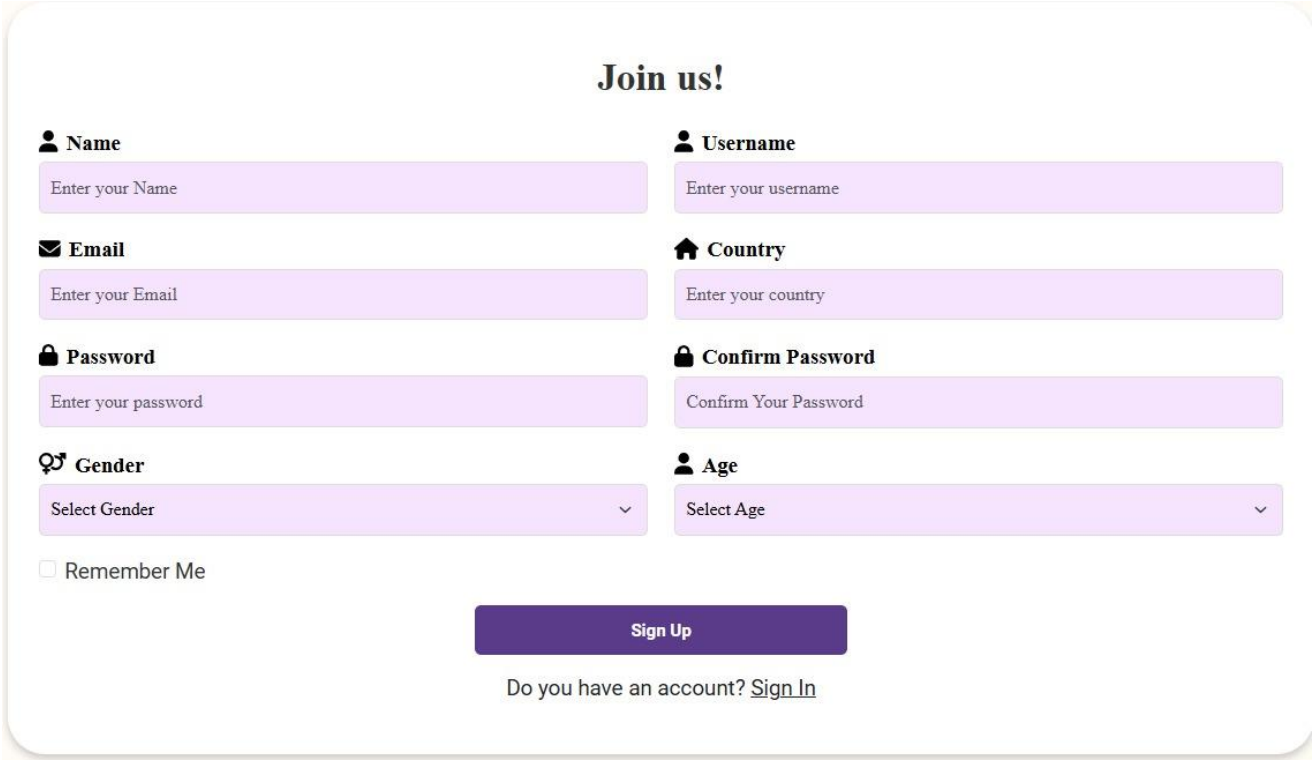


Figure F.5 Sign Up

F.6 Sign In Page

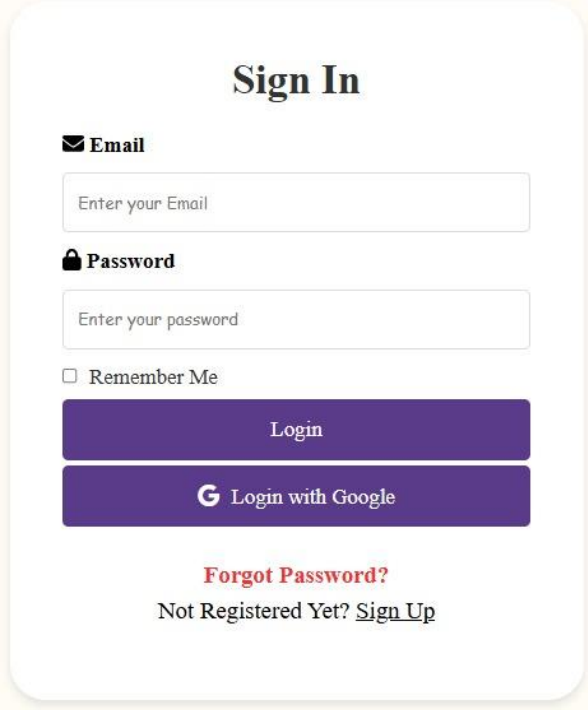


Figure F.6 Sign In

F.7 Frequently Asked Questions Page

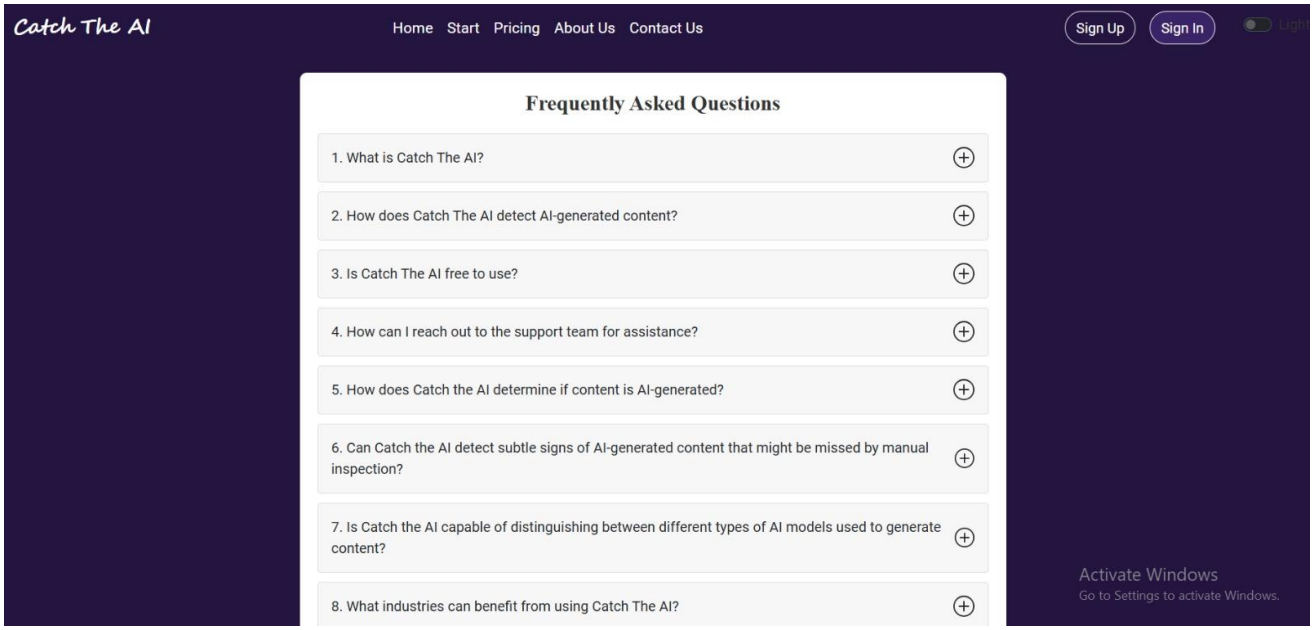


Figure F.7 FAQs

F.8 Terms of Service

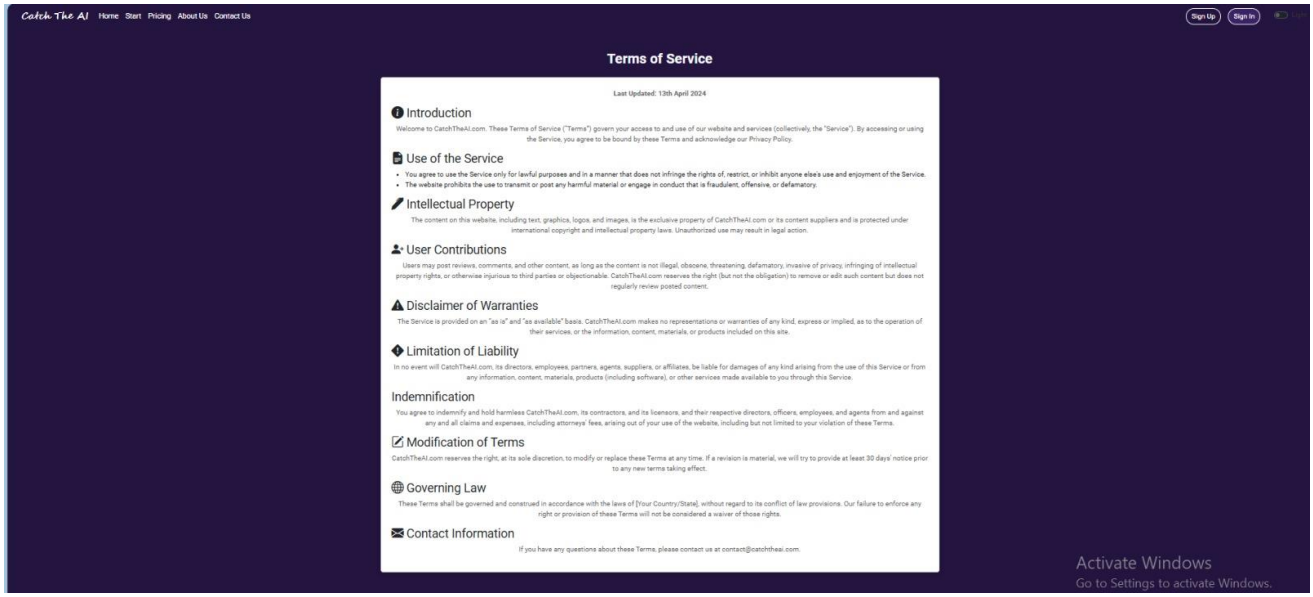


Figure F.8 Terms of Service

REFERENCES

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, et al., "Language Models are Few-Shot Learners," in Proc. 34th Adv. Neural Inf. Process. Syst., Dec. 2020, pp. 1877-1901. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] D. Crevier, "AI: The Tumultuous Search for Artificial Intelligence," Basic Books, 1993.

[3] Y. Wang, Q. She, M. G. Moghaddam, and X. Liu, "DeepFake Detection: A Comprehensive Survey," IEEE Trans. Pattern Anal. Mach. Intell., 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9844203

[4] Y. Li, M. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Apr. 2018, pp. 3267-3271. [Online]. Available: https://arxiv.org/abs/1806.02877

[5] M. M. Lucas, and M. V. Afanasyev, "Detecting Fake News in Social Media Networks Using Machine Learning," IEEE Trans. Comput. Soc. Syst., vol. 6, no. 3, pp. 544-556, Sep. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8704120

[6] K. K. Singh, V. Nangia, and S. Belkhir, "Deep Learning for AI-Generated Media Detection: A Survey," IEEE Access, vol. 9, pp. 123210-123224, Nov. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9641266

[7] J. K. Hwang, "Audio Deepfakes: Analysis, Detection, and Mitigation," in Proc. 2021 IEEE Int. Conf. Acoust. Speech Signal Process., Jun. 2021, pp. 2469-2473. [Online]. Available: https://ieeexplore.ieee.org/document/9413964

[8] R. Verma, A. Thakur, and M. Singh, "An Overview of Data Collection and Preprocessing Techniques for Machine Learning," in Proc. 2019 IEEE Int. Conf. Big Data, Dec. 2019, pp. 4157-4161. [Online]. Available: https://ieeexplore.ieee.org/document/898156

[9] Y. Lu, Y. Zhang, and L. Song, "Survey of AI and Privacy Protection," IEEE Commun. Surv. Tutor., vol. 23, no. 4, pp. 2255-2281, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9449312

[10] J. Daiber, and M. P. Thomas, "AI-Driven Detection Technologies for Law Enforcement: A Survey," IEEE Secur. Priv., vol. 19, no. 3, pp. 67-76, May 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9418891

[11] M. H. N. Tuan, "AI in Academia: Challenges and Opportunities in the Detection of AI-Generated Content," Comput. Educ. Artif. Intell., vol. 2, no. 1, Jun. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666920X21000144

[12] K. Fukushima, and N. Kawaguchi, "Deep Learning Methodologies and Applications: A Review," IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 7, pp. 1953-1974, Jul. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8954164

[13] A. Van Deursen, and J. Visser, "UML-Based Specification for Interactive Systems: A Review," Softw. Syst. Model., vol. 19, no. 3, pp. 361-380, Mar. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10270-019-00771-7

[14] L. Downey, "Guidelines for Effective Documentation in Software Engineering," IEEE Softw., vol. 36, no. 5, pp. 43-50, Sep. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8803917

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, et al., “Language Models are Unsupervised Multitask Learners,” OpenAI, Feb. 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist., Jun. 2019, pp. 4171-4186. [Online]. Available: https://arxiv.org/abs/1810.04805

[17] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” in Proc. 34th Adv. Neural Inf. Process. Syst., Dec. 2020, pp. 12449-12460. [Online]. Available: https://arxiv.org/abs/2006.11477

[18] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” MIT Press, 2016.

[19] L. Pinto, L. Almeida, and P. Mendes, “Challenges and Opportunities in AI-Generated Content Detection,” IEEE Access, vol. 8, pp. 139091-139102, Aug. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9142976

[20] R. Girshick, “Fast R-CNN,” in Proc. 2015 IEEE Int. Conf. Comput. Vis., Dec. 2015, pp. 1440-1448. [Online]. Available: https://arxiv.org/abs/1504.08083

[21] F. Chollet, “On the Measure of Intelligence,” arXiv Prepr. arXiv1911.01547, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1911.01547

[22] T. H. Davenport, and R. Ronanki, “Artificial Intelligence for the Real World,” Harvard Business Review, vol. 96, no. 1, pp. 108-116, Jan. 2018. [Online]. Available: https://hbr.org/2018/01/artificial-intelligence-for-the-real-world

[23] J. Spillner, “Patterns and Practices for Software Containers: Architecture, Methodology, Portability,” in Proc. 2017 IEEE Int. Conf. Software Architecture Workshops, Apr. 2017, pp. 57-64. [Online]. Available: https://ieeexplore.ieee.org/document/7949164

[24] F. Jiang, L. Chen, and X. Y. Wang, “An Overview of Multimodal Deep Learning in AI Systems,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 7, pp. 1-13, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9283424

[25] M. Rao, “Effective API Development with Django and Django REST Framework,” Packt Publishing, 2020. [Online]. Available: https://www.packtpub.com/product/effective-django-rest-framework/9781800560096

[26] L. Baresi, and L. Pasquale, “A UML Use Case Driven Approach to Requirements Engineering,” in Proc. 2006 IEEE Int. Conf. on Software Engineering and Knowledge Engineering, Jul. 2006, pp. 423-430. [Online]. Available: https://ieeexplore.ieee.org/document/1627604

[27] M. D. P. Papazoglou, and W. J. Van Den Heuvel, “Service-Oriented Design and Development Methodology,” J. Database Manage., vol. 16, no. 4, pp. 45-70, Oct. 2005. [Online]. Available: https://www.igi-global.com/article/service-oriented-design-development-methodology/3331

[28] N. Leavitt, “Security in the Cloud: Protecting Data and Applications,” IEEE Cloud Comput., vol. 3, no. 2, pp. 44-50, Mar. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7460194

[29] P. Zave, “Multimodal Systems: Integration of Different Modalities in System Design,” IEEE Softw., vol. 26, no. 2, pp. 63-70, Mar. 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4814280

[30] M. Fowler, “UML Distilled: A Brief Guide to the Standard Object Modeling Language,” 3rd ed., Addison-Wesley, 2003. [Online]. Available: https://www.oreilly.com/library/view/uml-distilled/0321193687/

[31] M. Bender, “SQL & NoSQL Databases: Models, Languages, Consistency Options, and Architectures for Big Data Management,” Springer, 2018. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-61837-5

[32] T. B. Wen, D. Bogdanov, and X. Serra, "An Audio-Visual Dataset for Fake and Real Speech Detection," arXiv preprint arXiv:1905.04515, 2019. [Online]. Available: https://arxiv.org/abs/1905.04515

[33] G. Goswami, "Scenefake: A Dataset of Deepfake Audio Clips," Proc. 2021 IEEE Conf. Acoustics, Speech and Signal Processing (ICASSP), pp. 7528-7532, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9414200

[34] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Proc. 31st AAAI Conf. Artificial Intelligence (AAAI-17), pp. 4278-4284, 2017. [Online]. Available: https://arxiv.org/abs/1602.07261

[35] T. Kinnunen, et al., "ASVspoof 2019: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan," arXiv preprint arXiv:1904.10320, 2019. [Online]. Available: https://arxiv.org/abs/1904.10320

[36] H. Delgado, et al., "ASVspoof 2021: Automatic Speaker Verification Spoofing and Countermeasures Challenge," Proc. Interspeech 2021, pp. 4319-4323, 2021. [Online]. Available: https://www.isca-speech.org/archive/pdfs/interspeech_2021/1112.pdf

[37] S. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," arXiv preprint arXiv:1710.10196, 2018. [Online]. Available: https://arxiv.org/abs/1710.10196

[38] Z. Liu, et al., "Deep Learning Face Attributes in the Wild," Proc. 2015 IEEE Int. Conf. Computer Vision (ICCV), pp. 3730-3738, 2015. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[39] S. Karras, et al., "Analyzing and Improving the Image Quality of StyleGAN," Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 8107-8116, 2020. [Online]. Available: https://arxiv.org/abs/1912.04958

[40] R. Rombach, et al., "High-Resolution Image Synthesis with Latent Diffusion Models," arXiv preprint arXiv:2112.10752, 2021. [Online]. Available: https://arxiv.org/abs/2112.10752

[41] L. von Platen, et al., "Diffusion Models: A Comprehensive Survey of Methods and Applications," arXiv preprint arXiv:2301.09675, 2023. [Online]. Available: https://arxiv.org/abs/2301.09675

[42] T. A. Salimans, et al., "Improved Techniques for Training GANs," Proc. 2016 Conf. Neural Information Processing Systems (NeurIPS), pp. 2234-2242, 2016. [Online]. Available: https://arxiv.org/abs/1606.03498

[43] E. Chan, et al., "Efficient Geometry-aware 3D Face Generation with Multi-Scale Learning," arXiv preprint arXiv:2303.07247, 2023. [Online]. Available: https://arxiv.org/abs/2303.07247

[44] A. T. Liu, et al., "Large Language Models: A Review of Capabilities, Applications, and Challenges," arXiv preprint arXiv:2403.03201, 2024. [Online]. Available: https://arxiv.org/abs/2403.03201

[45] C. Potthast, et al., "No preprocessing? No problem! Text categorization using raw text," Proc. 2018 ACM Conf. Research and Development in Information Retrieval (SIGIR), pp. 1503-1506, 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3209978.3210181

[46] T. Pollet, and H. Steeneken, "Speech Signal Sampling Rate: Impact on Recognition Accuracy," IEEE Trans. Speech Audio Process., vol. 10, no. 8, pp. 650-656, 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1168392

[47] J. Barker, et al., "Truncation in Speech Processing: Benefits and Trade-offs," IEEE Signal Process. Lett., vol. 12, no. 6, pp. 492-495, 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1457608

[48] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Proc. 2012 Conf. Neural Information Processing Systems (NeurIPS), pp. 1097-1105, 2012. [Online]. Available: https://dl.acm.org/doi/10.5555/2999134.2999257

[49] Y. Liu, et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[50] P. He, et al., "DeBERTa: Decoding-enhanced BERT with Disentangled Attention," Proc. 2021 Int. Conf. Learning Representations (ICLR), pp. 1-10, 2021. [Online]. Available: https://arxiv.org/abs/2006.03654

[51] A. Baevski, et al., "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," Proc. 2020 Conf. Neural Information Processing Systems (NeurIPS), pp. 12449-12460, 2020. [Online]. Available: https://arxiv.org/abs/2006.11477

[52] M. Tan, and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Proc. 2019 Int. Conf. Machine Learning (ICML), pp. 6105-6114, 2019. [Online]. Available: https://arxiv.org/abs/1905.11946

[53] K. He, et al., "Deep Residual Learning for Image Recognition," Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016. [Online]. Available: https://arxiv.org/abs/1512.03385

[54] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1251-1258, 2017. [Online]. Available: https://arxiv.org/abs/1610.02357

[55] D. D. Lewis, "Evaluating and Optimizing Autonomous Learning Algorithms," Machine Learning Review, vol. 24, no. 4, pp. 411-437, 2007. [Online]. Available: https://link.springer.com/article/10.1007/s10994-007-5038-3

[56] L. Verdoliva, "Media Forensics and Deepfakes: An Overview," IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 910-932, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9121205

[57] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, Univ. California, Irvine, 2000. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.html

[58] R. Wieruch, The Road to React, Leanpub, 2021. [Online]. Available: https://www.roadtoreact.com/

[59] "Introduction," Bootstrap Documentation, 2024. [Online]. Available: https://getbootstrap.com/docs/5.3/getting-started/introduction/

[60] "Getting Started with Postman," Postman Learning Center, 2024. [Online]. Available: https://learning.postman.com/

[61] "Getting Started," NPM Documentation, 2024. [Online]. Available: https://docs.npmjs.com/about-npm

[62] "Introduction," Django REST Framework Documentation, 2024. [Online]. Available: https://www.django-rest-framework.org/