

Deep Learning based Videos, Audios and Image Forgery Detection

A thesis submitted by

Mariam Fatima **(2021-BSE-020)**

Supervisor

Dr. Mukhtiar Bano

Submitted in the partial fulfillment of the requirements for the degree of
BSc. in Software Engineering

DEPARTMENT OF SOFTWARE ENGINEERING
FATIMA JINNAH WOMEN UNIVERSITY THE MALL ROAD RAWALPINDI

July 2025

DEPARTMENT OF SOFTWARE ENGINEERING



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

“Starting with the name of Allah, who is the Most Merciful and the Most Beneficial”

ACKNOWLEDGEMENT

I express my gratitude to Allah first, and then I want to appreciate the individuals Without whom this would not have been possible. I am deeply thankful to my supervisor, Dr. Mukhtiar Bano, for her invaluable help and guidance throughout the course of my Final Year Project. Her instructions, encouragement, and inspiration were crucial in accomplishing the project's goals. I also extend my heartfelt appreciation to my parents and other family members. Their contributions, assistance, and motivation have served as an unending source of strength throughout this endeavor. I appreciate you for consistently being present for me.

DEDICATION

To Allah Almighty

I dedicate this project and all my efforts to Almighty Allah. I am deeply grateful for His boundless help and mercy.

To My Parents

To my guiding parents, who instilled in me the values of faith in Allah, the importance of hard work, and the belief that anything can be achieved with dedication and effort.

To My Supervisor (Dr. Mukhtiar Bano)

To my mentor and supporter, who provided invaluable encouragement and guidance throughout this endeavor.

Table of Contents

List of Tables	viii
List of Figures.....	ix
List of Abbreviations	2
INTRODUCTION.....	4
1. INTRODUCTION.....	5
1.1. BACKGROUND	5
1.2. PROBLEM STATEMENT	6
1.3. AIMS AND OBJECTIVES	6
1.4. THESIS LAYOUT.....	7
LITERATURE REVIEW	8
2. LITERATURE REVIEW	9
MATERIAL AND METHODS	15
3. MATERIALS AND METHODS	16
3.1. REQUIREMENTS.....	16
3.1.1. FUNCTIONAL REQUIREMENTS	16
3.1.2. NON-FUNCTIONAL REQUIREMENTS	17
3.2. SYSTEM DESIGN	18
3.2.1. USE CASE DIAGRAM.....	18
3.2.2. CLASS DIAGRAM	20
3.2.3. SEQUENCE DIAGRAM.....	22
3.3. METHODOLOGY DEVELOPED	25
3.3.1. SOFTWARE TOOLS	25
3.3.2. PROGRAMMING LANGUAGES.....	26
3.3.3. LIBRARIES AND PACKAGES	26
3.3.4. SERVER-SIDE TOOLS	27
3.4. IMPLEMENTATION DETAIL	27
3.4.1. DATASET DETAILS.....	27
3.4.2. DATA PREPARATION	28
3.4.3. LOADING OF AUDIO FILES.....	28
3.4.4. FEATURE EXTRACTION	29

3.4.5.	LABEL ENCODING	30
3.4.6.	METHOD DESCRIPTION	31
3.4.7.	REFERENCE FEATURE CREATION	31
3.4.8.	PROCESSING OF UPLOADED AUDIO.....	31
3.4.9.	CLASSIFICATION THROUGH SIMILARITY MEASUREMENT	32
3.4.10.	OUTPUT DISPLAY	33
3.4.11.	DATABASE DETAILS.....	33
3.5.	IMPLEMENTATION DETAILS	34
3.5.1.	FRONTEND DEVELOPMENT/UI DEVELOPMENT	34
3.5.2.	DATABASE DEVELOPMENT.....	35
3.5.3.	BACKEND DEVELOPMENT.....	35
RESULTS AND DISCUSSION		37
4.	RESULTS AND DISCUSSION	38
4.1.	PROTOTYPE DEVELOPMENT	38
4.1.1.	HOME PAGE	38
4.1.2.	LOGIN PAGE.....	39
4.1.3.	REGISTRATION PAGE	39
4.1.4.	UPLOAD PAGE.....	40
4.1.5.	RESULT PAGE	41
4.1.6.	ABOUT PAGE	42
4.1.7.	DATABASE	43
4.2.	TESTING.....	43
4.2.1.	TEST CASE FOR INVALID LOGIN ATTEMPT	44
4.2.2.	TEST CASE FOR USER REGISTRATION.....	45
4.2.3.	TEST CASE FOR DUPLICATE EMAIL REGISTRATION	45
4.2.4.	TEST CASE FOR PASSWORD MISMATCH IN REGISTRATION	46
4.2.5.	TEST CASE FOR UPLOADING VALID AUDIO FILE.....	46
4.2.6.	TEST CASE FOR UPLOADING INVALID FILE TYPE.....	47
4.2.7.	TEST CASE FOR AUDIO FILE WITH NO VOICE CONTENT.....	47
4.2.8.	TEST CASE FOR UPLOAD WITHOUT LOGIN.....	48
4.2.9.	TEST CASE FOR FAKE AUDIO DETECTION	49

4.2.10.	TEST CASE FOR REAL AUDIO DETECTION	49
4.2.11.	TEST CASE FOR AUDIO PROCESSING TIME.....	50
4.2.12.	TEST CASE FOR BROWSER COMPATIBILITY.....	51
4.2.13.	TEST CASE FOR LOGGING OUT	51
4.2.14.	TEST CASE FOR SESSION EXPIRY AFTER LOGOUT	52
4.3.	IMPLEMENTATION RESULTS	52
4.3.1.	VISUAL ANALYSIS OF AUDIO FEATURES	53
4.3.2.	WAVEFORMS	53
4.3.3.	SPECTOGRAM ANALYSIS	55
4.3.4.	MEL SPECTOGRAM	57
4.3.5.	CHROMAGRAM	59
4.3.6.	MFCCs.....	61
4.4.	DISCUSSION	63
CONCLUSION AND FUTURE WORK		64
5.	CONCLUSION AND FUTURE WORK	65
5.1.	CONCLUSION.....	66
5.2.	FUTURE WORK.....	66

List of Tables

Table 2.1: Literature Review	12
Table 3.1: Use Case	19
Table 3.2: Class Diagram.....	21
Table 3.3: Sequence Diagram for Registration.....	22
Table 3.4: Sequence Diagram for Login.....	23
Table 3.5: Sequence Diagram for Audio Detection.....	24
Table 4.1: Login (TC-01).....	44
Table 4.2: Invalid login attempt (TC-02).....	44
Table 4.3: User registration (TC-03)	45
Table 4.4: Duplicate email registration (TC-04).....	45
Table 4.5: Password mismatch in registration (TC-05)	46
Table 4.6: Uploading valid audio file (TC-06)	46
Table 4.7: Invalid audio file (TC-07).....	47
Table 4.8: Audio with no voice content (TC-08).....	47
Table 4.9: Upload without login (TC-09)	48
Table 4.10: Fake audio (TC-010).....	49
Table 4.11: Real audio (TC-011)	49
Table 4.12: Audio processing time (TC-012)	50
Table 4.13: Browser Compatibility (TC-013)	51
Table 4.14: Logging out (TC-014).....	51
Table 4.15: Session expiry after logout (TC-015)	52

List of Figures

Figure 2.1: Speech to Speech Conversion	11
Figure 3.1: Use Case Diagram	18
Figure 3.2: Class Diagram	20
Figure 3.3: Sequence Diagram for Registration	22
Figure 3.4: Sequence Diagram for Login	23
Figure 3.5: Sequence Diagram for Audio Detection	24
Figure 4.1: Home page.....	38
Figure 4.2: Login page.....	39
Figure 4.3: Registration page.....	39
Figure 4.4: Upload page.....	40
Figure 4.5: After Uploading Audio.....	41
Figure 4.6: Fake Audio	41
Figure 4.7: Real Audio.....	42
Figure 4.8: About Page	42
Figure 4.9: Database	43
Figure 4.10: Waveform for real audio	53
Figure 4.11: Waveform for fake audio	54
Figure 4.12: Spectrogram for real audio	55
Figure 4.13: Spectrogram for fake audio	56
Figure 4.14: Mel spectrogram for real audio	57
Figure 4.15: Mel spectrogram for fake audio	58
Figure 4.16: Chromagram for real audio	59
Figure 4.17: Chromagram for fake audio	60
Figure 4.18: MFCCs for real audio.....	61
Figure 4.19: MFCCs for fake audio.....	62

List of Abbreviations

MFCC	Mel-frequency Cepstral Coefficients
Chroma	Chroma feature analysis
Mel Spectrogram	Mel-scaled spectrogram
Tonnetz	Tonal centroid features
Euclidean Distance	Distance metric
Cosine Similarity	Similarity metric
MongoDB	NoSQL database
Librosa	Python audio library
PyMongo	Python MongoDB library
Flask	Python web framework
WAV	Audio file format

ABSTRACT

In today's digital age, the production and dissemination of deceitful digital content is a serious issue that endangers individual privacy, cybersecurity, as well as the general reliability of online information. With evolving artificial intelligence, it has become feasible to produce exceedingly realistic deceptive content that can deceive individuals, propagate lies, and even fuel criminal activities like impersonation and fraud. To fight this emerging threat, our system, Vox Guard, presents a deep learning-based mechanism to identify and categorize counterfeit digital content through powerful computational methods.

The system evaluates significant feature patterns derived from uploaded digital files based on techniques including Mel Frequency Cepstral Coefficients (MFCCs), spectrogram-based analysis, chroma features, spectral centroid, zero-crossing rate, and spectral flatness. These characteristics are transformed into numerical vectors and matched against a pre-tagged database with real and phony entries. Using distance-based algorithms of similarity—like Euclidean distance—the system determines the nearest match and labels the uploaded content as such. It also returns a percentage confidence, giving users a perception of how trustworthy the detection is.

Vox Guard prioritizes simplicity, precision, and user anonymity. It stores the user credentials safely in a MongoDB database and does not keep any of the processed or uploaded files. The software is made to continually develop by changing to new methods of forgery and adding future improvements such as identity verification and real-time detection.

This project will help advance digital safety through the provision of a reliable tool for authenticating digital content against forgery. Vox Guard would prove particularly useful to individuals, companies, and institutions that depend on the authenticity of digital information and are keen to safeguard themselves against deception and manipulation online.

Chapter-1

INTRODUCTION

1. INTRODUCTION

This chapter explains the background and problem statement of Vox Guard, focusing on the problems that users face in detecting deepfake audio and the need for an inclusive solution. The system builds trust by accurately detecting audio and consistently improves performance while offering an easy-to-use platform to users. This chapter establishes key principles that explain why Vox Guard seeks to redefine the way by which user's authenticate audio content.

1.1. BACKGROUND

The development of digital communication requires immediate attention to audio authenticity because deepfake technologies are increasingly spreading. Traditional systems do not have efficient artificial audio detection methods which results in misinformation along with security threats and trust breakdowns. The available software solutions face limited success when trying to solve these issues despite their modest attempt at addressing them. Vox Guard distinguishes itself from others because it provides a user-friendly platform that makes audio verification process easier than ever for all users.

The intelligent audio authentication platform named Vox Guard helps users in detecting deepfake content when users need to verify its authenticity through streamlined efficient analysis. Users benefit from the Vox Guard website because it maintains a central system for audio uploads and analysis which excludes the need of complex tools and reduces the barrier of technical expertise. The platform offers an easy-to-use interface that enables quick and certain authentication of audio clips for users.

Vox Guard enhances its audio feature extraction with MFCCs and Mel spectrogram as well as chroma and spectral characteristics. The algorithm contrasts these features with previous verified data records by performing distance-based operations to define manipulation features. The system generates exact results that enable users to make decisions for personal needs and professional and legal requirements.

Users must register and login to upload the audio files for detection. Due to the absence of the feedback function the system focuses on eliminating complexity while ensuring high performance and complete transparency. The evolution of Vox Guard security software

depends on user feedback to establish effective audio analysis features that keep digital users confident in modern data protection systems.

1.2. PROBLEM STATEMENT

The ability to determine real versus fake audio files proves difficult to users because deepfake technologies have improved along with their accessibility becoming wider. Users face issues in audio detection mainly because they lack reliable tools and they find it hard to analyze features while they are burdened by complex technology and there is no simple detection platform available. The unreliable methods used for detection and unclear result outputs defeat users' confidence in real and fake audio identification. The absence of proper authentication methods alongside slow manual checks creates security threats for people and institutions at the same time. The current limitations in speedy misinformation response require the creation of an easy-to-use system that verifies audio authenticity precisely.

1.3. AIMS AND OBJECTIVES

This study aims to assess the impact of audio feature extraction and distance-based comparison methods in detecting fake audios using the Vox Guard system, which is designed for accurate classification of real and fake audios. To fulfill the aim, following objectives were set:

- To develop a user-authenticated Flask-based web application for uploading and analyzing audio files for deepfake detection.
- To extract meaningful audio features such as MFCCs, Mel spectrogram, chroma, zero-crossing rate, spectral centroid, and flatness using Librosa.
- To implement a distance calculation mechanism comparing uploaded audio features with an existing dataset to classify the audio as real or fake.
- To evaluate the detection result based on similarity scores and present the classification outcome with associated certainty in the system interface.

1.4. THESIS LAYOUT

Chapter 1: Introduction

Start with an introduction, outlining the growing risks of audio manipulation and the motivation behind Vox Guard, a deepfake detection software.

Chapter 2: Literature on Review

The literature review explores various deepfake detection methods and audio verification techniques, analyzing their effectiveness and gaps.

Chapter 3: Material and Methods

Material and methods chapter presents the functional and non functional requirements of the system and also a detailed view of Vox Guard design structure.

Chapter 4: Methodologies

Methodologies chapter presents the applied techniques including feature extraction with Librosa and distance-based classification for detection.

Chapter 5: Prototype and System Testing

Prototype and system testing documents the development of the Vox Guard interface and evaluates performance through multiple test cases.

Chapter 6: Results and discussion

Results and discussion includes all the results obtained, graphs and their explanation and a section of discussion.

Chapter 7: Conclusion and Future Work

In conclusion, and future work reflects on Vox Guard's impact and outlines future enhancements for improved accuracy and reach.

Chapter-2

LITERATURE REVIEW

2. LITERATURE REVIEW

This Chapter presents a review of existing research on deepfake audio detection. This part conclude key studies that have explored some techniques, models and challenges in this field.

Hamza et al. (2022) [1] reviewed a Deepfake audio detection system by using MFCC features and machine learning methods like Random Forest, SVM, and XGBoost. They assess different sub-dataset with preprocessing dataset and get better accuracy, particularly along with VGG-16. This system approach is merged many feature extraction method and employed transfer learning for deep learning models. This method demonstrate remarkable refinement across previous work in deepfake detection.

Elgibreen and Almutairi (2022) [2] proposed advance Deepfake audio detection methods, classifying them into machine learning and deep learning models. They evaluated the effectiveness of these method opposed to imitation as well as fake audio attacks. Their work highlights the difficulties of detecting deepfakes in noisy, camorous environments and non-English languages. They emphasized the need for more robust and generalized detection models.

Shaaban et al. (2023) [3] proposed latest Deepfake generation and detection techniques, providing methods like TTS and VC for fake audio creation. They examin handcrafted and deep learning-based detection methods such as CNNs, RNNs, and LCNNs. Their study demonstrate that SVM get the highest detection accuracy of 99%. They highlight challenges in noisy environments and the need for detection models.

Xie et al. (2025) [4] reviewed the Codecfake dataset targeting ALM-based deepfake audio. They demostrate failures of current detection models against codec-based audio generation methods. The CSAM strategy they reviewed better model adaptability across domains. Their experiments demonstrate that vocoder-trained models performed poorly, with a remarkable reduction in error rates.

Di Pietro, Bakiras and Rabhi (2024) [5] demonstrated the vulnerability of audio deepfake classifiers to adversarial attacks. They highlights that state-of-the-art models, like Deep4SNet, can be significantly fooled by adversarial attacks, reducing detection accuracy from 98.5% to nearly 0%. To notice this, they reviewed a generalizable, lightweight defense mechanism that

can be merged into existing audio-deepfake detectors, expanding and enhancing resilience opposed to attacks.

Phat Lam and alo Lam Pham (2024) [6] reviewed a deep learning system for deepfake audio detection using spectrogramvariations and auditory filters. They trained CNN, RNN, and C-RNN models on spectrograms, achieving an Equal Error Rate (EER) of 0.03 on the ASVspoof 2019 dataset. They examin transfer learning from computer vision models and pre-trained embeddings. The consequences demonstrate the potential of merging spectrograms with deep learning.

Chintha et al. (2020) [7] proposed a convolutional bidirectional recurrent network for detecting audio and video deepfake. They attain new benchmarks on FaceForensics++, Celeb-DF, and ASVSpooof 2019 datasets. Their techniques explored entropy-based loss functions, improving detection accuracy. This method highlighted robustness over various deepfake generation techniques.

Khochare et al. (2022) [8] reviewed a deep learning framework for audio deepfake detection using the Fake or Real dataset. They merged feature-based and image-based approaches with TCN and STN models, achieving 92% accuracy. Their TCN model outperformed traditional machine learning algorithms. They emphasized the effectiveness of deep learning for robust detection.

Yang and Zhou (2024) [9] built a joint learning framework that uses a Temporal-Spatial Encoder, Multimodal Joint-Decoder, and Cross-Modal Classifier. This framework captures temporal-spatial features and cross-modal inconsistencies between audio and video. They tested it on multimodal deepfake datasets and achieved high accuracy. This technique was effective but showed drawbacks with datasets having low inter-modal manipulation variance.

Song and Lee (2024) [10] reviewed an unsupervised method for deepfake audio detection using GANomaly and f-AnoGAN models. Their technique preprocess audio with mel-spectrograms and MFCCs to detect anomalies. According to this research paper they attain F1-score of 0.93, giving an effective solution for detecting new deepfake audio. This method required no additional training data.

Ba and Xia (2024) [11] introduced Avoid-DF, that integrated temporal-spatial encoding, multimodal joint-decoder, and cross-modal classifier to enhance detection of deepfake videos. Their methods capture audio-visual inconsistencies for accurate detection. Although, the technique was minimum by the availability of diverse multimodal deepfake datasets for robust training.

Alsoubi and also Mubarak (2023) [12] provided a survey on deepfakes in audio, visual, and text formats. In this research paper they mentioned the risks deepfakes pose to trust, security, and information integrity. Their review stressed the need for unified detection solutions and

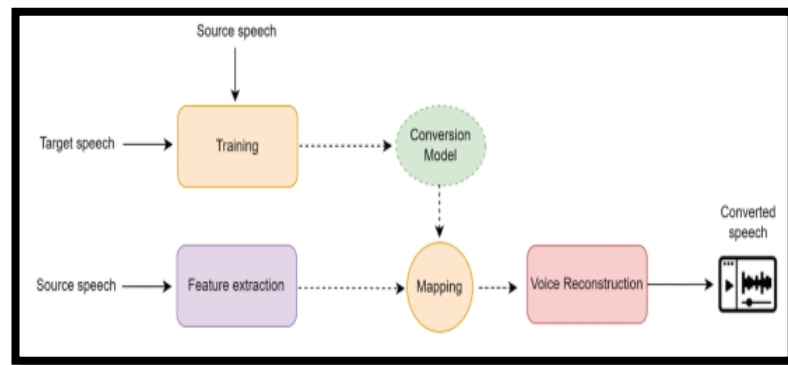


Figure 2.1: Speech to Speech Conversion

public awareness. This work serves as a key resource for understanding deepfake impacts and solutions.

Wang and Huang (2024) [13] proposed ART-AVDF, a narrative approach for audio-visual deepfake detection. In this paper discuss about the model that utilizes articulatory representations by extracting audio features related to speech movements and combining them with lip motion data. A multimodal joint fusion module enhances the detection of deepfakes by leveraging both audio and visual cues. Extensive experiments on datasets like DFDC and FakeAVCeleb show ART-AVDF significantly outperforms existing models.

Alshehri and Almalki (2024) [14] presented Sonic Sleuth, a deep learning model for detecting audio deepfakes with a custom CNN architecture. This model get 98.27% accuracy and a 0.016 EER on a primary dataset. It demonstrated an 84.92% accuracy and a 0.085 EER on an external dataset. The model's ability to generalize across diverse audio inputs demonstrated its real-world application potential.

Lee et al. (2025) [15] presented a dual-channel model for deepfake audio detection using both direct and reverberant waveforms. Their model addressed the drawbacks of single-channel

methods and showed improvements across multiple datasets. Authors presented a new Sports Press Conference dataset to enhance detection performance. Their approach led to better detection results for audio deepfakes.

Table 2.1: Literature Review

Author	Approaches	Features	Limitations
Ameer Hamza, Abdul Rehman Javaid[1]	A hybrid ML/DL approach using MFCC features with classifiers like SVM, VGG-16	Mel-Frequency Cepstral Coefficients(MFCC), spectral features, raw signal features and signal energy	Machine learning models struggle with complex audio changes in original dataset. Deep models require high computational resources and time.
Zaynab Almutairi, Heban Elgibreen [2]	The paper reviews Machine and Deep learning based audio Deepfake detection methods.	Features include MFCC, LFCC, CQCC and spectrogram-based limited.	Current methods often overfit and require excessive preprocessing or language-limited.
Ousama A. Shaaban, Remzi Yildirim [3]	A survey of handcrafted and deep learning methods including CNNs, RNNs and hybrid models for detecting synthetic audio.	Acoustic features like MFCC, spectrograms and log power spectra are used for detection.	Detection accuracy drops in noisy environments and with unseen speaker data.
Yuankun Xie, Yi Lu [4]	Introduced the large-scale codec fake dataset and CSAM strategy for robust, generalized detection of Deepfake.	Neural codec waveform features tailored for detecting ALM-based Deepfake audio.	Existing ADD models perform poorly on ALM-generated audio due to domain bias and overfitting to vocoded data.
Mouna Rabhi [5]	Proposed a lightweight,	Use 2D audio representations with	Deepfake detectors like Deep4SNet are

	general-purpose defense mechanism against adversarial audio attacks.	CNN-based Deep4SNet for fake detection.	highly vulnerable to GAN-based adversarial attacks.
Lam pham , Phat LAM [6]	Ensemble of CNN, RNN, transfer learning models and audio embedding's from pre-trained models.	Spectrograms from STFT, CQT, and WT with auditory filters like Mel, Gammatone, and DCT.	Performance may vary across different spoofing types and depends on spectrogram selection.
Akash Chintha, Bao Thai [7]	Merge CNN and bidirectional RNN with entropy-based loss for multimodal Deepfake detection.	Latent audio/video features extracted via CNNs and executed with bidirectional RNNs.	Potential domain generalization problems despite strong benchmark performance.
Janavi Khochare, Chaitalu Joshi [8]	Uses feature-based Machine learning and image-based Deep learning.	Spectral features and Mel spectrograms derived from audio samples.	ML models underperform compared to deep learning on sequential data like audio.
Wenyuan Yang, Xiaoyu Zhou	Uses a joint learning framework with a Temporal-spatial Encoder, Multimodal Joint-Decoder and Cross-Modal Classifier.	Temporal-spatial and cross-modal features capturing audio-visual inconsistencies.	Performance may degrade on datasets with limited inter-modal manipulation variance.
Daeun Song, Nayoung Lee [10]	Uses unsupervised GAN-based anomaly detection (GANomaly, f-AnoGAN) trained only on real audio.	Mel-spectrogram and MFCC used to model actual audio patterns.	Struggles with highly sophisticated Deepfake that mimic real audio

			distribution closely.
Zhongjie Ba, Zhihua Xia [11]	Avoid-DF merges Temporal-spatial Encoding, Multi-Modal joint-Decoder, and Cross-Modal Classifier for improves detection.	Avoid-DF leverages multimodal audio-visual inconsistency for detecting Deepfake videos.	Limited by the availability of multi-modal Deepfake datasets for comprehensive training.
Rami Mubarak, Tariq Alsboui [12]	Examine Deepfake types, detection techniques and their social political and economic impacts.	Detail Survey on detection and impacts of visual, audio and textual Deepfake	Lack of a unified, real-time solution for detecting all types of Deepfake.
Yujia Wang, Hua Huang [13]	Merges audio and visual articulatory features with self-supervised learning for improves Deepfake detection	ART_AVDF uses articulatory representation learning for enhances audio-visual Deepfake detection.	The approach relies on accurate articulatory features, which may be difficult to capture in noisy real-world scenarios.
Anfal Alshehri, Danah Almalki [14]	Serves a custom CNN architecture for detecting audio deep fake with excellent performance on various datasets.	Sonic Sleuth is a deep learning model using CNNs to detect audio Deepfake with high accuracy.	The model may struggle with very novel or unknown Deepfake generation techniques not included in the training data.
Lee et al.[15]	Uses direct and reverberant sounds to improve Deepfake audio detection	Dual-channel audio detection	Potential real-time challenges.

Chapter-3

MATERIAL AND METHODS

3. MATERIALS AND METHODS

This Chapter covers the material and methods employed in this thesis. It provides comprehensive information on the implementation process, covering aspects such as dataset details, data preparation and the steps involved in both front-end and backend development.

3.1. REQUIREMENTS

Requirements specify the functional and non-functional requirements that the system is expected to meet. These requirements define what the system will do and how it will perform.

3.1.1. FUNCTIONAL REQUIREMENTS

These are the functional requirements that describe what the system should do.

- **User Registration:** Users can create an account by entering their username, email, and password. The system checks for any duplicate emails and securely saves the user's credentials in MongoDB after hashing the password.
- **User Login:** Registered users can log in with their email and password. The system verifies the credentials and allows access to the upload page upon successful login.
- **Audio Upload and Detection:** Once logged in, users can upload audio files (mp3, wav, flac, aiff, opus, amr, aac). The system extracts features like MFCC, Mel spectrogram, chroma, zero-crossing rate, spectral centroid, and flatness, compares them with the dataset, and calculates similarity to determine if the audio is real or fake.
- **Detection Result Display:** The results are displayed immediately after processing. The system shows messages such as "Fake Audio detected with 82.315% certainty" or "Real Audio detected with 91.028% certainty."
- **Upload Page:** Logged-in users are directed to the upload page where they can select and submit audio files. They can upload multiple files one at a time and view the results for each.
- **Profile Management:** During registration, users provide their username, email, and password. They can log out and register again with a different email.
- **MongoDB Integration:** MongoDB is utilized to store and manage user credentials (username, email, hashed password). Detection results are not saved in the database.

- **Logout:** Users can log out of the system. Logging out permanently deletes their account from the MongoDB database, requiring them to register again to regain access.
- **Security:** Passwords are stored in a hashed format using pbkdf2:sha256 for secure authentication. Unauthorized users are prevented from accessing the upload or detection features.

3.1.2. NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality standards and behavior of the Vox Guard system. These include:

- **Accuracy:** The system should provide trustworthy classification of audio as real or fake with a confidence percentage using internal feature comparison.
- **Security:** User credentials must be securely stored in MongoDB using password hashing. Only users that are logged in can access the detection feature, and no audio data is saved after detection.
- **Usability:** The interface must be clean, simple, and easy to use, allowing users to register, log in, upload audio, and view results with minimal effort.
- **Performance:** The system should quickly process uploaded audio and return the detection result without obvious delay.
- **Portability:** The application should work on desktops, laptops, and modern web browsers without any need of special installations.
- **Reliability:** The system should operate reliably by avoiding breakdowns that occur during standard file uploading and processing operations.
- **Maintainability:** The application must be built in a modular and clean code structure so future improvements or bug fixes can be made easily.

3.2. SYSTEM DESIGN

This section presents the high-level architecture and design of the system. System Design uses diagram to illustrate the system's components and their interactions.

3.2.1. USE CASE DIAGRAM

This section illustrates the interaction between users and the systems. It depicts the system's functionality from the user's perspective.

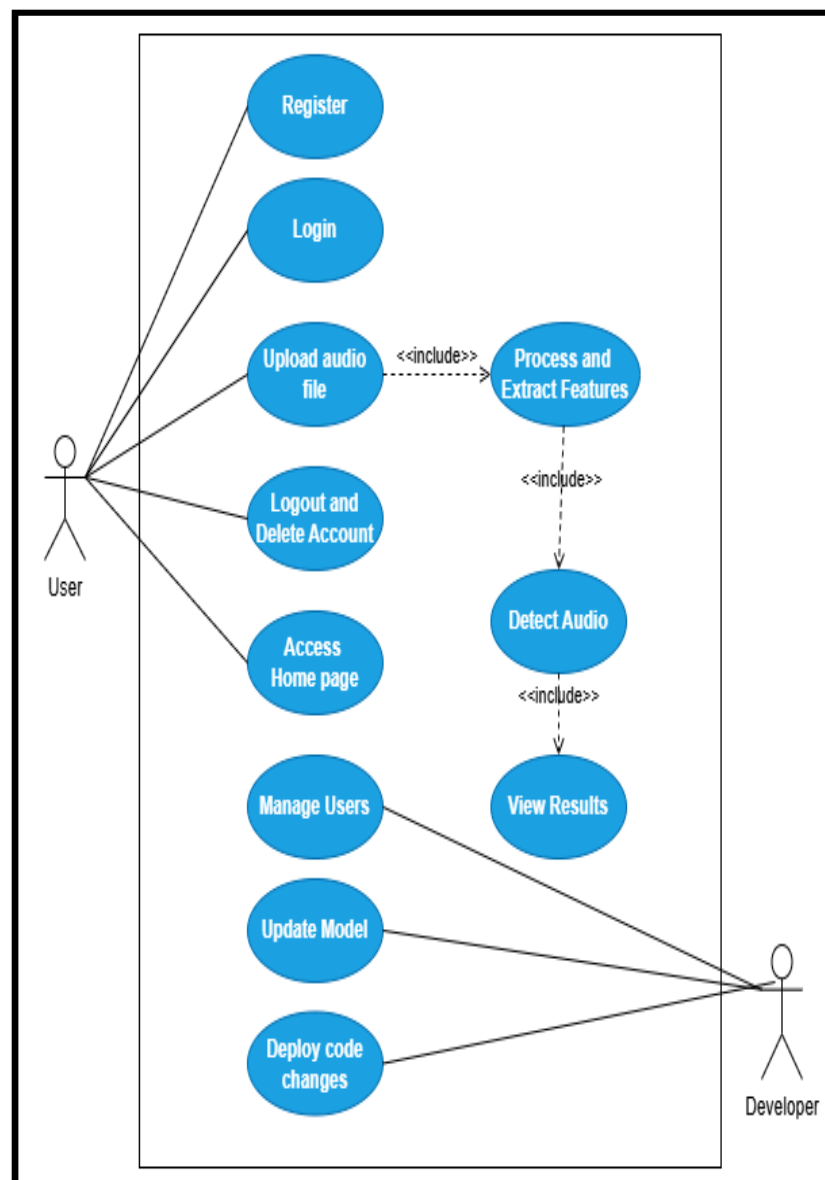


Figure 3.1: Use Case Diagram

Table 3.1: Use Case

Use Case	Actor	Description	Relationship
Register	User	Allows a new user to create an account.	–
Login	User	Authenticates user access with credentials.	–
Upload Audio	User	Enables the user to upload audio for analysis.	Includes → Process Audio
Process Audio	System	Extracts features from uploaded audio.	Includes → Classify Audio
Classify Audio	System	Compares features and classifies audio as real or fake.	–
Display Results	System	Shows the classification result and confidence score.	Extends → Upload Audio
Back to Upload	User	Allows the user to upload another audio file.	Extends → Display Results
Logout	User	Ends the current user session.	–

3.2.2. CLASS DIAGRAM

This section explains the structure of the system by showing its classes, attributes and the relationships between them. It provides a static view of the system.

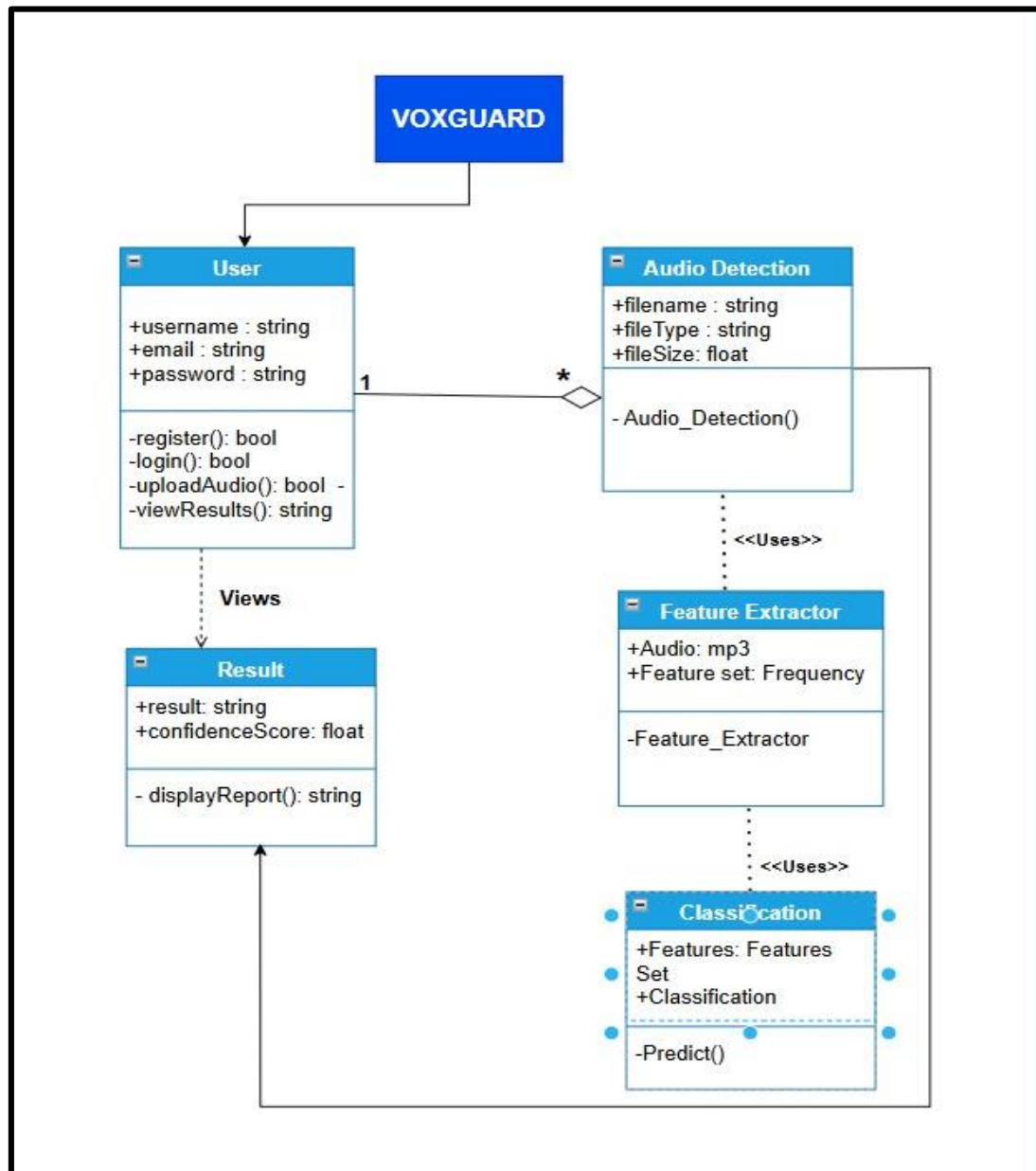


Figure 3.2: Class Diagram

Table 3.2: Class Diagram

Classes	Attribute	Operation
User	Username, email, password	This class allows the user to register, login and upload audio files for analysis. It also enables users to view the results of Deepfake detection process.
Audio Detection	File_name, File_type, File_size	This class manages the audio file including type and size. It initiate the detection process by calling the feature extractor.
Feature Extraction	Audio (mp3, wav), Feature set	This class processes the uploaded audio file and extracts frequency-based features. The extracted features are then passed for classification.
Classification	Features (Feature Set), Classification	This class receives extracted features and does classification. It predicts whether the audio is real or fake based on learned patterns.
Result	Result, Confidence Score	This class stores the detection result along with the prediction confidence score.

3.2.3. SEQUENCE DIAGRAM

This section illustrates the sequence of interactions between objects in time-ordered manner. Its depicts how objects interact to perform specific tasks.

For Registration:

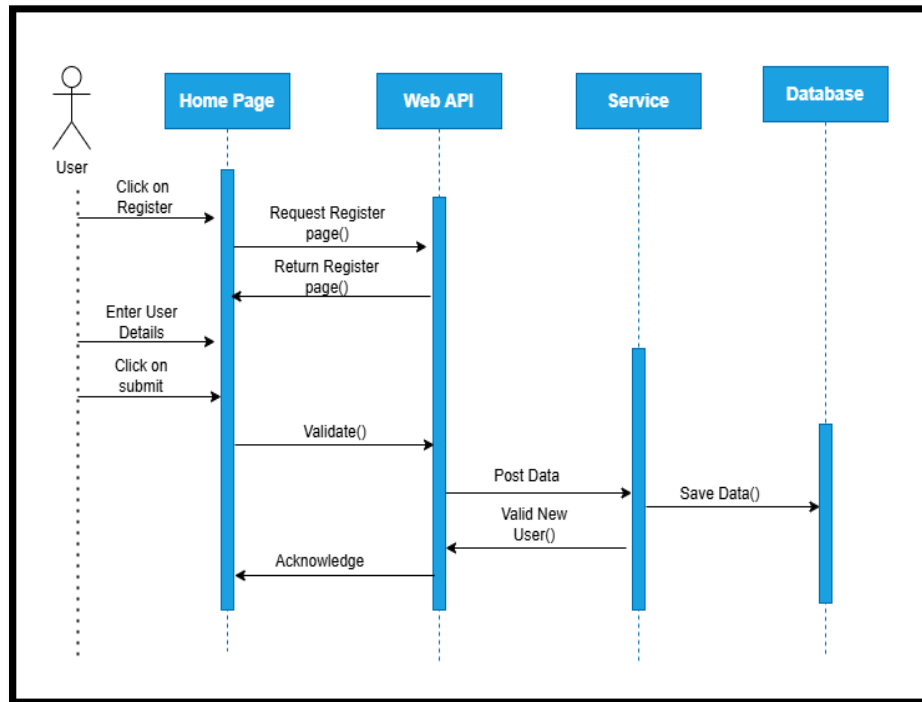


Figure 3.3: Sequence Diagram for Registration

Table 3.3: Sequence Diagram for Registration

Identification	Register
Purpose	This component lets the user register.
Type	A Module
Function	When users use the VoxGuard then firstly they will register by entering their details such as contact info. The user has to authenticate themselves for using this Website.
Subordinates	Validated account log in information.
Dependencies	This component depends on database record and login modules.
Interface	The user interface provides a button with text “Register here”.

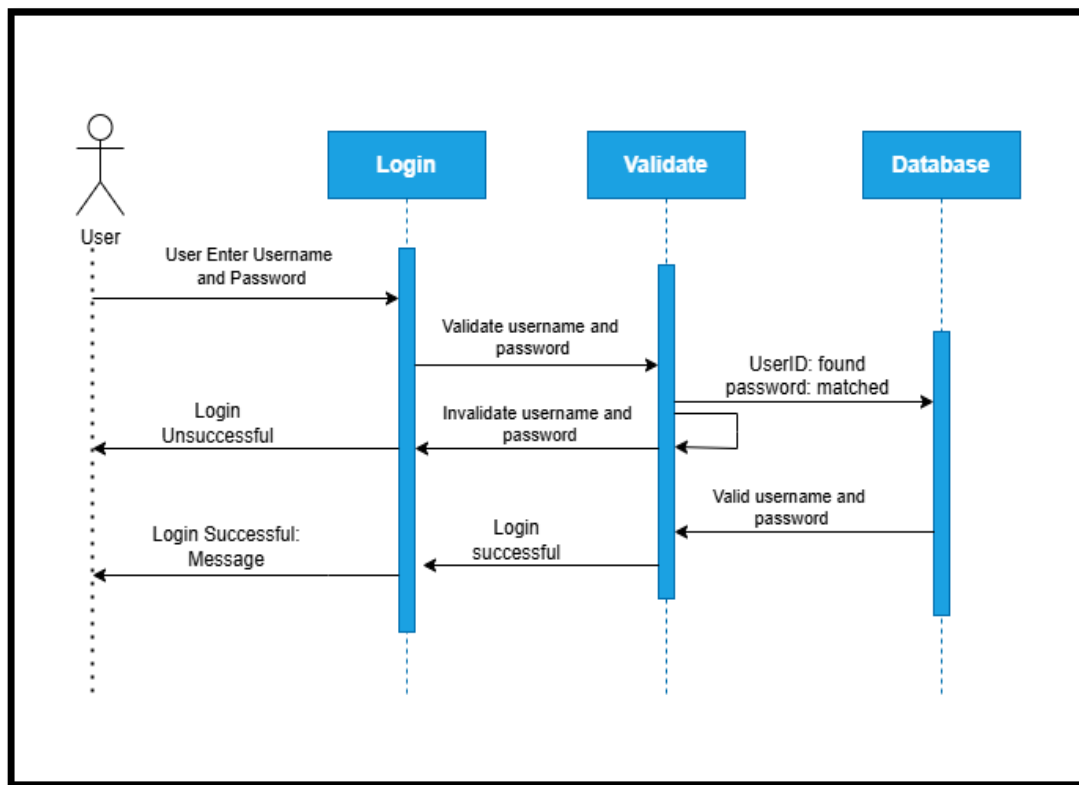
For Login:

Figure 3.4: Sequence Diagram for Login

Table 3.4: Sequence Diagram for Login

Identification	Login
Purpose	This component allows the user to login to VoxGuard by providing their username and password.
Type	A Module
Function	When a user enters their credentials, the system validates the information against stored records in the database. If the credentials match, the login is successful and the user is redirected to the main interface; otherwise a login unsuccessful message is shown.
Subordinates	Username and password validation.
Dependencies	This depends on the Validate module and the Database to retrieve and confirm user credentials.
Interface	The interface provides input fields for the username and password, and a "Login" button.

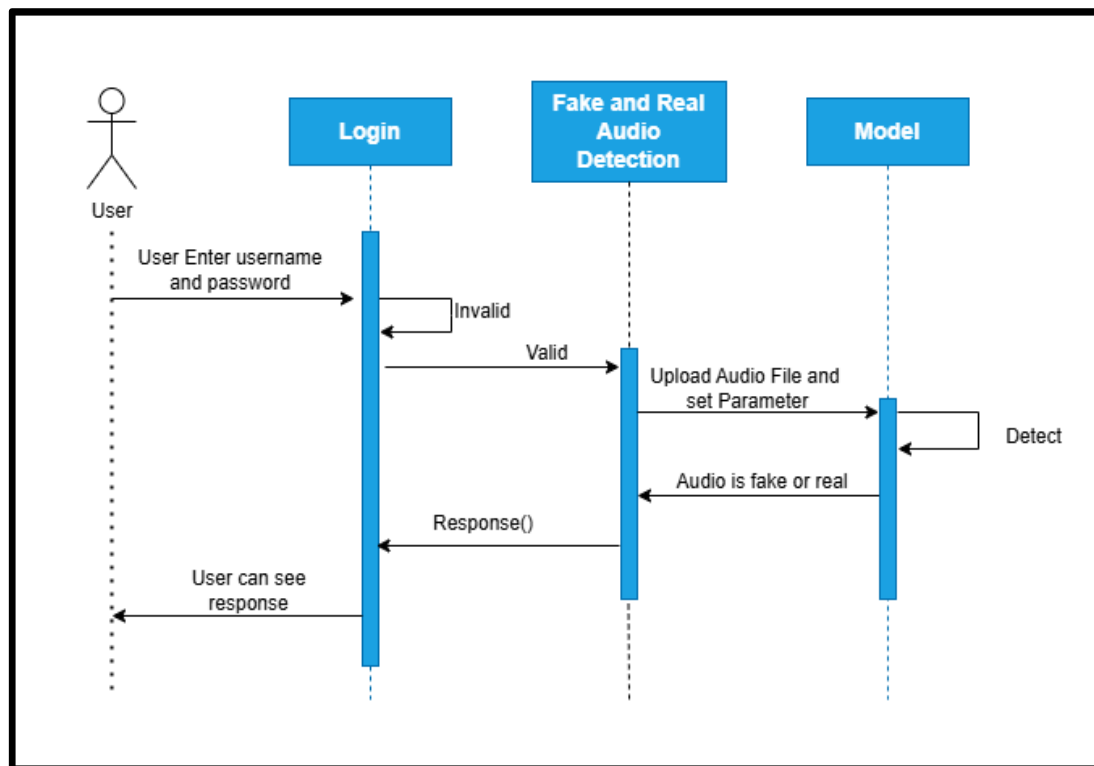
For Audio Detection:

Figure 3.5: Sequence Diagram for Audio Detection

Table 3.5: Sequence Diagram for Audio Detection

Identification	Fake and Real Audio Detection
Purpose	This component handles the core logic of detecting whether an uploaded audio file is real and fake.
Type	A Module
Function	After successful login, the user uploads an audio file. This component sets necessary parameters and sends the audio file for detection. It responds back with whether the audio is fake or real.
Subordinates	Parameters setting, result handling, communication with detection model.
Dependencies	Depends on successful login audio file input from the user, and the model module for detection.
Interface	The user interface provides an option to upload an audio file and view the result is Real or Fake.

3.3. METHODOLOGY DEVELOPED

This section covers the specific tools, languages, and libraries used in the development process. It explains the technical approach taken.

3.3.1. SOFTWARE TOOLS

These are the following tool that is used to develop the website.

Development Environment

- **Visual Studio Code:** Used as the primary local development environment to build and manage the Flask web application. It provides features like syntax highlighting, debugging tools, and terminal integration.
- **Google Colab:** Utilized for testing and extracting audio features using Librosa. It provides a convenient cloud-based environment for experimenting with audio processing. Audio files were uploaded here to extract features like MFCC, mel spectrogram, and chroma, which were later used for real vs. fake classification in the Flask app.

Server and Database

- **Flask:** A lightweight Python web framework used to develop the backend of the web application, manage routing, handle file uploads, user authentication, and render HTML templates. In Vox Guard, Flask was responsible for managing user registration and login using MongoDB, handling audio file uploads for detection, triggering the feature extraction and classification process, and displaying detection results along with confidence percentages through dynamically rendered web pages.
- **MongoDB:** A NoSQL database used to store user information such as email, username, and passwords. It is accessed using the PyMongo library in Python. In the Vox Guard project, MongoDB stores all registered user credentials securely, enabling login and registration functionality. It does not store audio files or detection results, focusing only on managing user authentication data efficiently.
- **MongoDB Compass:** A GUI tool used to view, manage, and verify the contents of the MongoDB database during development and testing. In Vox Guard, it was utilized to visually confirm the successful storage of user registration data such as usernames, emails, and hashed passwords, ensuring that the authentication system was working correctly.

3.3.2. PROGRAMMING LANGUAGES

These programming languages specifies the languages used to write the software code.

- **Python:** The Backend development of the Vox Guard project uses Python as its main programming language. The program collects audio features from Librosa while performing classification by means of distance evaluation. Flask integration in Python facilitates routing as well as file upload management which enables mongoDB connection through PyMongo and result display with calculated confidence percentage.
- **HTML:** The web pages used HTML to develop their structure through five critical elements: login, registration, upload, result and about functionality. This approach helps users understand the material in an organized manner. Each screen in Vox Guard received its design through HTML which allowed users to access system functions by logging in and uploading files while viewing results in an easy-to-understand style.
- **CSS:** The web pages obtained their visual design through CSS while enabling minimal refinements to keep everything look well organized. The application of CSS throughout Vox Guard contributed to improved looks of its login, registration, upload, about and result pages to provide an interface that remains equally friendly for everyone.

3.3.3. LIBRARIES AND PACKAGES

This section describes the external code used to facilitate development.

- **Librosa:** A Python library that is used to extract audio features consisting of MFCCs, Mel spectrogram, Chroma, Zero Crossing Rate, Spectral Centroid and Spectral Flatness from the uploaded audio files. In Vox Guard, these features play an essential role in analyzing audio acoustic data which allows comparison compared to reference data for establishing authenticity.
- **Numpy and Pandas:** It is used for performing numerical computations and managing dataset operations, especially in computing distance measures. Vox Guard uses NumPy for carrying out essential computations to analyze extracted audio features by determining the comparison distance through Euclidean or cosine measures between reference data and input features to identify authentic or simulated audio.

- **Werkzeug:** The Werkzeug library enables secure password hashing and provides functions for audio uploading and file management in Flask applications. During user registration and authentication sessions in Vox Guard Werkzeug provides password security through hashing the user passwords. Werkzeug helps run audio file processing procedures while securing the effective upload method for audio files that the application later uses for classification and feature extraction purposes.
- **PyMongo:** Used for connecting and performing operations on the MongoDB database from within the Flask application. In Vox Guard, PyMongo is used to handle user registration and login functionality by securely storing and obtaining user credentials (e.g., email, username, and password) in the MongoDB database, allowing effortless user authentication within the web app.

3.3.4. SERVER-SIDE TOOLS

This section describes the technologies used for server-side development and deployment.

- **Flask:** Main framework that is responsible for managing backend logic including user login/registration, session handling, audio upload, and classification functionality.
- **Jinja2:** Templating engine used within Flask to render dynamic HTML content.

3.4. IMPLEMENTATION DETAIL

This section provides in-depth information about how the system was built. This includes details about the data, its preparation, and the specific implementation of the core methods.

3.4.1. DATASET DETAILS

The dataset used in this study consists of specifically marked "real" and "fake" audio samples. Each audio sample uses .wav format storage and receives a label that indicates its authenticity against deepfake technology synthesis.

The .csv file supports efficient data organization and convenient access since it contains two main fields.

- **File Path:** The directory location of audio files is clearly identified in File Path so users can effortlessly find audio files during processing time.
- **Label:** The dataset contains two classes which are identified through the "real" label for authentic recordings while "fake" stands for artificially generated content.

Each audio recording extends from a few seconds to about ten seconds while uniformly using a format designed to extract speech features. Feature consistency supports the accurate extraction of MFCCs and Mel spectrograms and chroma vectors without need of any additional preprocessing modifications. This dataset structure along with its defined labels enables high suitability for audio classification algorithm training validation and evaluation processes.

3.4.2. DATA PREPARATION

The dataset needs multiple processing operations before being suitable for classification purposes. The data preprocessing step converts audio raw data into numerical representations which are further required for classification. The system first loads audio files before it runs normalization to standardize volume levels. The audio is divided into parts through segmentation procedures. The most important audio characteristics which are required to check if the audio is real or fake are extracted using techniques which include both Mel-frequency cepstral coefficients (MFCCs) and Mel spectrograms. After processing the feature vectors, the system prepares them for classification which enables accurate predictions through analyzed patterns.

3.4.3. LOADING OF AUDIO FILES

The Vox Guard system utilizes the Python-based Librosa library for audio file processing because it stands as a popular tool for audio analysis and feature extraction tasks. The Vox Guard system works differently from standard techniques because it maintains the original sampling frequencies of audio files so the acoustic properties stay same. The system properly preserves the original format of uploaded samples because this step helps protect essential speech characteristics that are needed to identify synthetic audio from authentic sources.

Such processing method ensures high efficiency with added protection for extracted feature quality. The approach protects against possible degradation because it refrains from unnecessary resampling which enables the system to obtain full speech-related attributes. The feature extraction methods used consistently on all samples produce dependable MFCCs and

Mel spectrograms and chroma vectors that help achieve accurate audio classifications inside the Vox Guard system.

3.4.4. FEATURE EXTRACTION

To accurately detect the often-subtle differences between real and fake audio samples, the Vox Guard system performs thorough feature extraction from each uploaded audio file. This process is vital because deepfake audios can closely mimic real speech patterns, requiring advanced methods to capture slight anomalies that human ears might miss. Various audio features are extracted, each designed to emphasize specific features of the speech signal. Together, these features offer a detailed understanding of both the spectral (frequency-related) and tonal (pitch-related) properties of the sound.

The key features that are extracted from the audios include:

- **Mel-Frequency Cepstral Coefficients (MFCCs):** MFCCs are one of the most important and widely used features in audio analysis, especially in the field of speech recognition. In Vox Guard, MFCCs are extracted from the audios capturing the essential timbral and textural properties of the speech signal. These coefficients help in identifying the unique qualities of a speaker's voice, such as the way vocal tract shapes the sound, making them crucial for distinguishing between real and synthetic speech.
- **Chroma Features:** Chroma features capture the distribution of energy across the 12 distinct pitch classes of the musical octave, regardless of the specific frequency or loudness. In speech, pitch plays a significant role, and subtle inconsistencies in pitch patterns can indicate synthetic manipulation. By analyzing chroma features, the system can detect unnatural tonal structures that are often present in fake audio.
- **Mel Spectrogram:** A Mel spectrogram converts the audio signal into a visual representation based on both time and frequency, scaled according to the Mel scale, which reflects how the human ear perceives sound frequencies. This representation highlights how different frequencies change over time, allowing the system to spot unnatural transitions or artifacts that are common in deepfake audios but rare in genuine speech recordings.
- **Spectral Contrast:** Spectral contrast measures the difference between spectral peaks and valleys across different frequency bands. Natural speech typically exhibits sharp contrasts due to the richness and variation of vocal expressions. In contrast, synthetic audio often lacks this variation, resulting in smoother, less distinct spectral patterns. By analyzing

spectral contrast, the system can better detect the absence of natural sharpness and clarity, which often reveals the presence of deepfake synthesis.

- **Tonnetz Features:** Tonnetz, short for tonal network, captures the harmonic relationships between different frequencies. These features are commonly used in music information retrieval, but they also offer valuable insights into speech, particularly in identifying the harmonic structure of a voice. Deepfake audio may fail to replicate the complex, natural harmonic patterns of real human speech. Thus, Tonnetz features help in building a harmonic profile that can further aid in the classification of real versus fake speech.

After individually computing all of these features for a given audio sample, the extracted data are joined into a comprehensive single feature vector. This vector, representing all the captured properties of the audio, is then flattened into a one-dimensional array. This flattening ensures compatibility with the classification algorithm, making it easier for the system to process the input and make an informed decision regarding the authenticity of the audio. By combining these multiple feature types, Vox Guard achieves a robust and detailed representation of each sample, maximizing the chances of detecting even the most complex deepfake audios.

3.4.5. LABEL ENCODING

Since the labels are originally in string format ("real" and "fake"), they are converted into numerical values using LabelEncoder from the scikit-learn library. This step is essential for machine learning, which requires numeric class labels for the classification process. In Vox Guard system, the encoding is as follows:

- Real \rightarrow 0
- Fake \rightarrow 1

This conversion allows the classification algorithm to work efficiently by mapping the audio labels to numerical values, making it easier to calculate the distance metrics during feature comparison and ultimately determining whether the audio is genuine or synthetic based on the extracted features.

3.4.6. METHOD DESCRIPTION

The proposed system named Vox Guard depends on a classification method to validate audio input authenticity. The system methodology uses newly uploaded audio tests against pre-processed reference samples which exist in the system storage. The system makes originality assessments through feature comparison analysis between the input sample and reserved example data to figure out if the uploaded audio represents genuine human speech or simulated or manipulated material. The method starts with complete feature extraction which uses MFCC and Mel spectrogram and chroma and ends with distance-based classification. The similarity analysis measures the relationship between input features and reference features by evaluating their distance through Euclidean distance or cosine similarity calculation to establish precise classifications based on feature set resemblances. The method provides secure detection of deepfake audio through its strong framework.

3.4.7. REFERENCE FEATURE CREATION

The Vox Guard system starts by extracting and getting ready reference data from existing labeled datasets. A feature extraction pipeline operates on each audio file from the dataset by generating multiple acoustic features which include MFCCs along with chroma and Mel spectrogram and spectral contrast and tonal centroid features (tonnetz). All audio-acoustic features are integrated together into a singular vector format that represents distinctive audio elements of each sample.

The generated vectors receive storage along with their real or fake labels. The labeled feature vectors serve as the base to create the reference set used for classification. User-uploaded audio samples undergo comparison within the reference set of labeled examples to decide between real and fake authenticity of the input audio.

During system initialization the extraction process occurs together with storage of features to build a reference set that becomes functional when users upload audios for detection purposes.

3.4.8. PROCESSING OF UPLOADED AUDIO

Users transmit audio samples to Vox Guard which initiates several processing operations to make the samples ready for classification. Librosa library enables memory storage of the uploaded .wav file for feature extraction before detection. The system maintains audio

processing at the original format while resampling is excluded for retention of natural sample characteristics. The system provides an efficient method to analyze and compare features that appear in both the uploaded audio file and the reference data saved in its database.

- Mel-frequency cepstral coefficients (MFCCs)
- Chroma feature analysis
- Mel-scaled spectrogram
- Spectral contrast
- Tonnetz harmonic features

Once these features are extracted, they are integrated into a single one-dimensional vector, creating a numerical representation of the uploaded audio. This vector represents the key acoustic characteristics of the sample and can be directly compared to the pre-processed reference vectors stored in the system.

This processing ensures that the uploaded sample is represented in the same format and feature space as the reference data, maintaining uniformity across all classification tasks. By using consistent feature extraction methods, the system can accurately classify whether the uploaded audio is real or fake based on the similarity to the reference dataset.

3.4.9. CLASSIFICATION THROUGH SIMILARITY MEASUREMENT

Upon creating the feature vector from uploaded audio Vox Guard employs the Euclidean distance metric to conduct classification operations. The calculation determines Euclidean distances between feature vectors from the uploaded sample and all vectors of the reference set.

The process includes the following steps:

- Calculate the distance between the uploaded sample and every reference sample.
- Identify the reference sample with the smallest distance — i.e., the most similar one.
- Retrieve the label (real or fake) of the closest reference sample.
- Assign this label to the uploaded audio as its classification result.

In addition to the predicted label, the system calculates a classification confidence score. This score is computed based on the closeness of the input vector to the nearest reference samples,

and is expressed as a percentage. A smaller distance results in a higher confidence percentage, indicating the system's certainty in its prediction.

This approach assumes that similar types of audios have similar numerical features, and thus lie closer in the feature space. The use of nearest neighbor classification technique generates fast results that practitioners can easily understand.

3.4.10. OUTPUT DISPLAY

The classification work from Vox Guard appears to users through an easy-to-understand visual interface. The output includes:

- The user interface displays a straightforward result either as "Audio is Real" or "Audio is Fake."
- The classification outcome presents a confidence percentage displayed as "This audio is Real with 96.5% confidence" next to the result.
- The visually appealing output uses an audio article style interface that enhances the presentation of results.

The output format works best for users who do not have technical skills. Users can download the audio, set playback speed and adjust volume. The confidence percentage informs users about how strongly the system classifies (audio) while adding clarity to the decision-making process.

3.4.11. DATABASE DETAILS

The Vox Guard system incorporates a user authentication and management feature that depends on MongoDB database services for information storage. The NoSQL database MongoDB stores JSON-like documents in its database while offering high flexibility and scalability together with quick data fetching capabilities.

Within the Deepfake database there exists a collection named Deepfake which contains complete details of all the registered user accounts. Each document inside the collection holds information about a registered user with these particular fields.

- **User name:** Users of the system can find their name as the username assigned to their account during registration.

- **Email:** Email acts as the single identifying factor for users to access their accounts through the system.
- **Password:** User passwords appear in a protected format secured by PBKDF2 hashing algorithm with the purpose of protecting against data theft.

Through its pymongo library connection the Flask application executes secure transactions that allow managing user records within the MongoDB system. All passwords use password security tools from Werkzeug for hash encryption prior to database storage.

The steps included in the authentication process are as follow:

- During registration, the system confirms user inputs and assures the email is unique.
- A secure password hash is generated and stored in the MongoDB database.
- During login, the entered password is checked against the stored hash using `check_password_hash`.
- Upon successful login, a session is created, allowing the user to access the audio upload and detection functionality.

Logout functionality also includes a database operation that deletes the user record from MongoDB to prevent account persistence unless re-registered. This helps keep the system lightweight and secure for demonstrations or prototype usage.

By leveraging MongoDB, the Vox Guard system provides fast data management, and security without requiring complex relational database schemas. The flexibility of NoSQL storage aligns well with the lightweight and modular structure of the application.

3.5. IMPLEMENTATION DETAILS

This section outlines the key steps taken to implement the system. It focuses on the main development phases.

3.5.1. FRONTEND DEVELOPMENT/UI DEVELOPMENT

This section describes the front end development. Details the technologies and processes used to create the user interface.

- **HTML Structure:** The basic structure of web pages was created using HTML. Pages such as home, login, registration, upload, about and result display are set up to guide the user experience with appropriate forms and links. These pages were designed to enable smooth navigation, allowing users to easily access and interact with the system, whether for logging in, registering, uploading audio, or viewing detection results.
- **CSS Styling:** Webpages obtain their responsive design through Custom CSS which creates a clean interface for users. The designed pages deliver an attractive visually appealing style which supports their essential operational capabilities. The website design separates different zones including sign-in, upload section and outcome display keeping users able to easily move throughout the system while supporting all device types and dimensions.

3.5.2. DATABASE DEVELOPMENT

This section describes the database development. It explains how the database was designed and implemented.

- **Mongo DB Setup:** User authentication information including email and username together with password data is stored in Mongo DB database. Users authenticate through the PyMongo library into a database structure that contains a database named Deepfake together with a collection named Deepfake. A system design with this logic allows optimized storage along with future-proofing capability for user credentials storage.
- **Database Design:** User credentials include hashed passwords and email addresses that serve as authentication elements in the Deepfake collection. All data storage occurs at a minimum level and contains user-related information alone which ensures privacy standards and security measures. The database structure exists to reduce data storage of unimportant information without violating protected user credentials from the login and registration sequence.

3.5.3. BACKEND DEVELOPMENT

This section describes the back end development. It explain the server-side logic and functionality of the system.

- **Flask Framework:** The backend development relies on Flask Framework as its web framework. Users can reach routes for registration and login and file upload functionalities

through the tool. The implementation uses Flask to establish the framework that manages both request pathways and response delivery systems.

- **Librosa for Audio Processing:** The Librosa library is integrated in backend to process audio files that are uploaded by users. Audio features such as MFCCs, mel spectrogram, chroma, zero-crossing rate, spectral centroid, and spectral flatness are extracted from the audio files that are uploaded by the user to detect whether they are fake or not.
- **Distance Calculation for Classification:** After feature extraction, the system calculates the distance between the uploaded audio's features and those stored in a previously existing dataset. It compares features using Euclidean distance and classifies whether the uploaded audio is fake or real based on the closest match from the dataset.
- **File Upload:** Users can upload audio files in different formats (mp3, wav, flac, aiff, opus, amr, aac). The system authenticates the file types and saves the file securely in the upload folder.
- **Session Management:** The session management features of Flask are used to maintain user authentication records. All users are needed to perform login authentication before uploading audio files to the system for detection. Before being granted access to the upload page Flask ensures that their session contains proper authentication.

Chapter-4

RESULTS AND DISCUSSION

4. RESULTS AND DISCUSSION

This chapter presents the outcomes of the prototype development and testing phases. It details the various components of the prototype and provides a comprehensive analysis of the testing results.

4.1. PROTOTYPE DEVELOPMENT

The user interface of Vox Guard is made to be visually attractive, easy to use and user-friendly so that everyone who uses it can have a simple and enjoyable experience. This is how it works: The main screen of our app is the login screen, where users will enter their username/email and password in order to log in to the system and detect audios.

4.1.1. HOME PAGE

This is the homepage of our website. When you open our website, you'll hear an audio message saying, 'Hi, welcome to VoxGuard'.

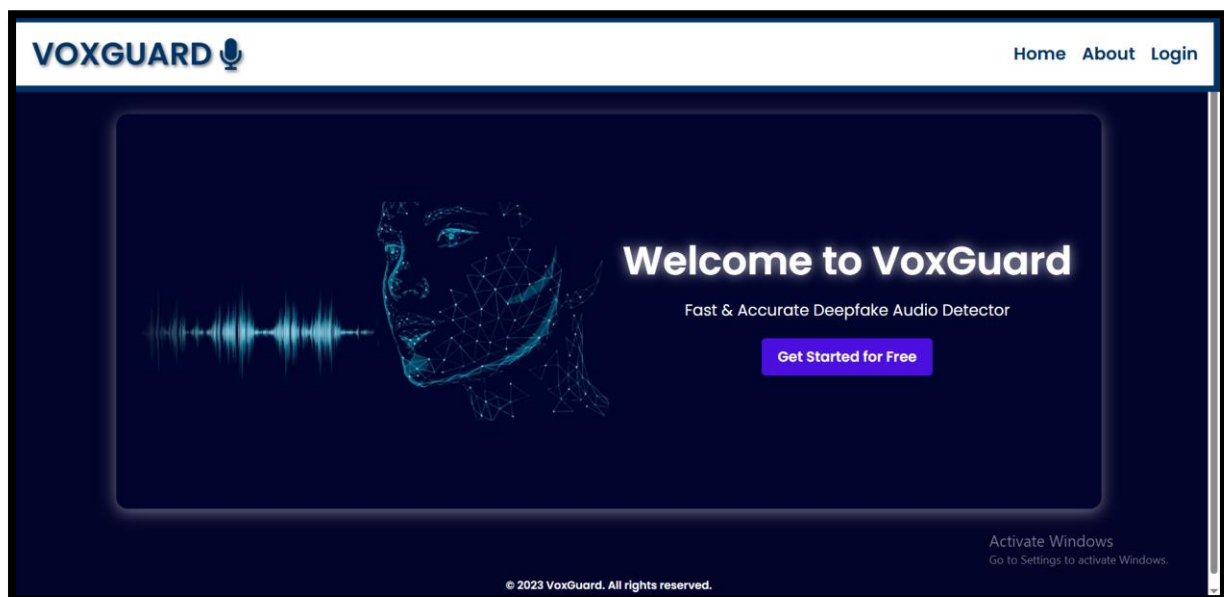
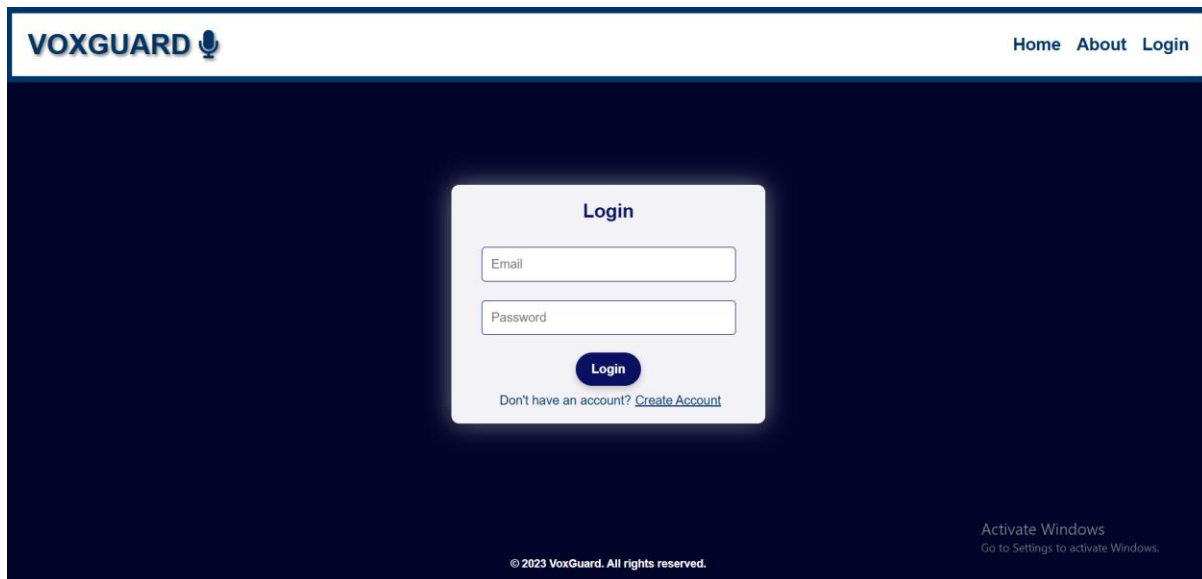


Figure 4.1: Home page

4.1.2. LOGIN PAGE

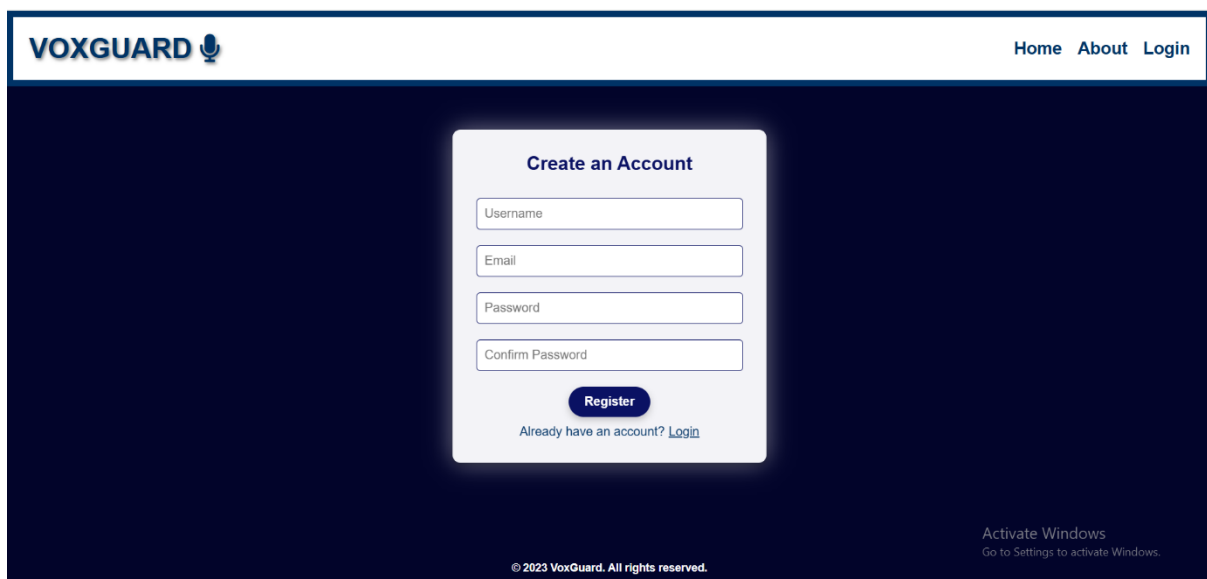


The screenshot shows the VoxGuard Login page. At the top, there is a header with the VoxGuard logo on the left and navigation links 'Home', 'About', and 'Login' on the right. The main content area has a dark blue background. In the center, there is a white login form titled 'Login'. The form contains two input fields: 'Email' and 'Password'. Below these fields is a blue 'Login' button. Under the button, there is a link that says 'Don't have an account? [Create Account](#)'. At the bottom right of the page, there is a small text area that says 'Activate Windows Go to Settings to activate Windows.' and a copyright notice '© 2023 VoxGuard. All rights reserved.'

Figure 4.2: Login page

If you are a new user then click on create account to create your account.

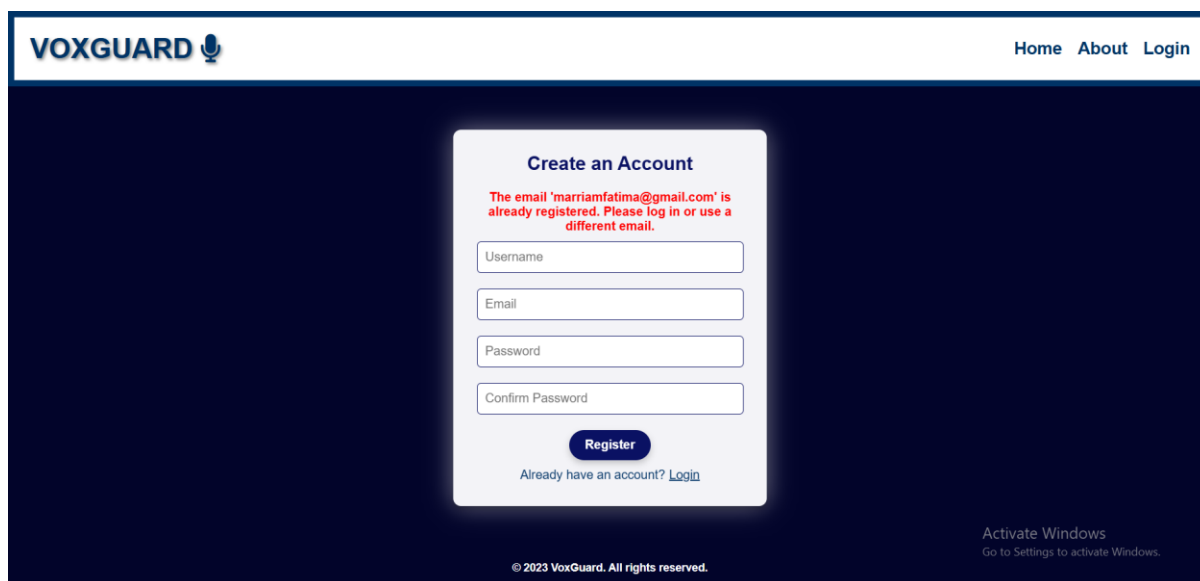
4.1.3. REGISTRATION PAGE



The screenshot shows the VoxGuard Registration page. At the top, there is a header with the VoxGuard logo on the left and navigation links 'Home', 'About', and 'Login' on the right. The main content area has a dark blue background. In the center, there is a white registration form titled 'Create an Account'. The form contains four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. Below these fields is a blue 'Register' button. Under the button, there is a link that says 'Already have an account? [Login](#)'. At the bottom right of the page, there is a small text area that says 'Activate Windows Go to Settings to activate Windows.' and a copyright notice '© 2023 VoxGuard. All rights reserved.'

Figure 4.3: Registration page

Fill in these fields to sign up. Enter your name, email and password for proper registration.

Errors:

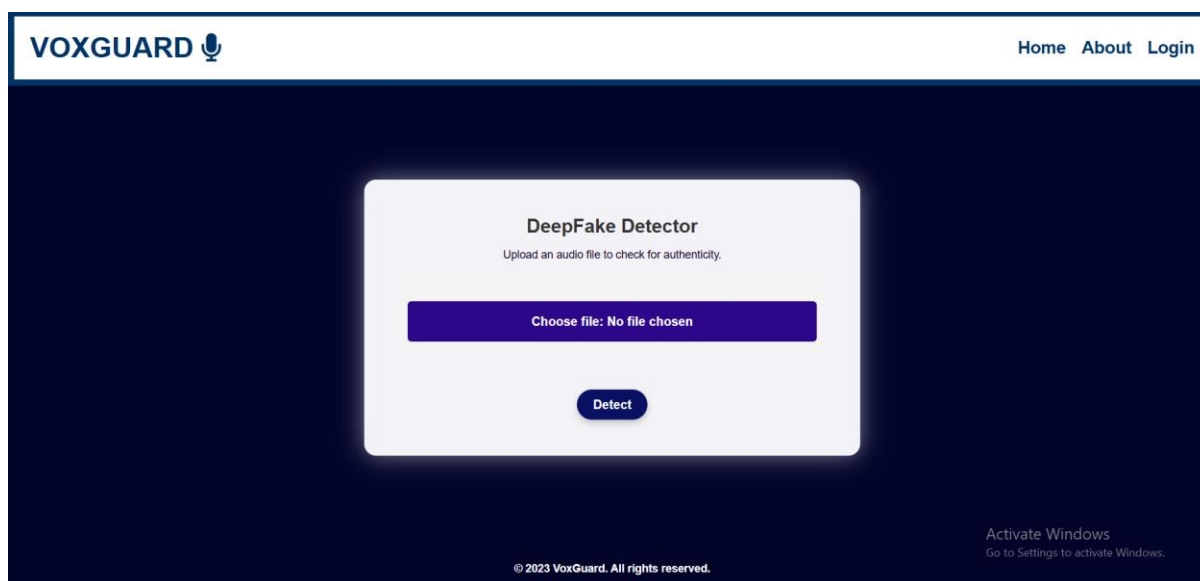
The screenshot shows the 'Create an Account' form on the VoxGuard website. The form has four input fields: Username, Email, Password, and Confirm Password. A red error message is displayed above the Email field: 'The email 'marriamfatima@gmail.com' is already registered. Please log in or use a different email.' Below the fields is a 'Register' button and a link for 'Already have an account? Login'. The page includes a header with the VoxGuard logo and navigation links (Home, About, Login), and a footer with copyright information and an 'Activate Windows' watermark.

Figure 4.1.4 Error message for already registered email

If user enters duplicate email, then this error occurs.

4.1.4. UPLOAD PAGE

This is the upload page. When you open it, an audio message plays saying, 'Please upload an audio file for detection.'



The screenshot shows the 'DeepFake Detector' page on the VoxGuard website. The page has a header with the VoxGuard logo and navigation links (Home, About, Login). The main content area features a 'DeepFake Detector' title, a subtitle 'Upload an audio file to check for authenticity.', and a file upload button labeled 'Choose file: No file chosen'. Below the button is a 'Detect' button. The page includes a footer with copyright information and an 'Activate Windows' watermark.

Figure 4.4: Upload page

After uploading audio, it looks like this:

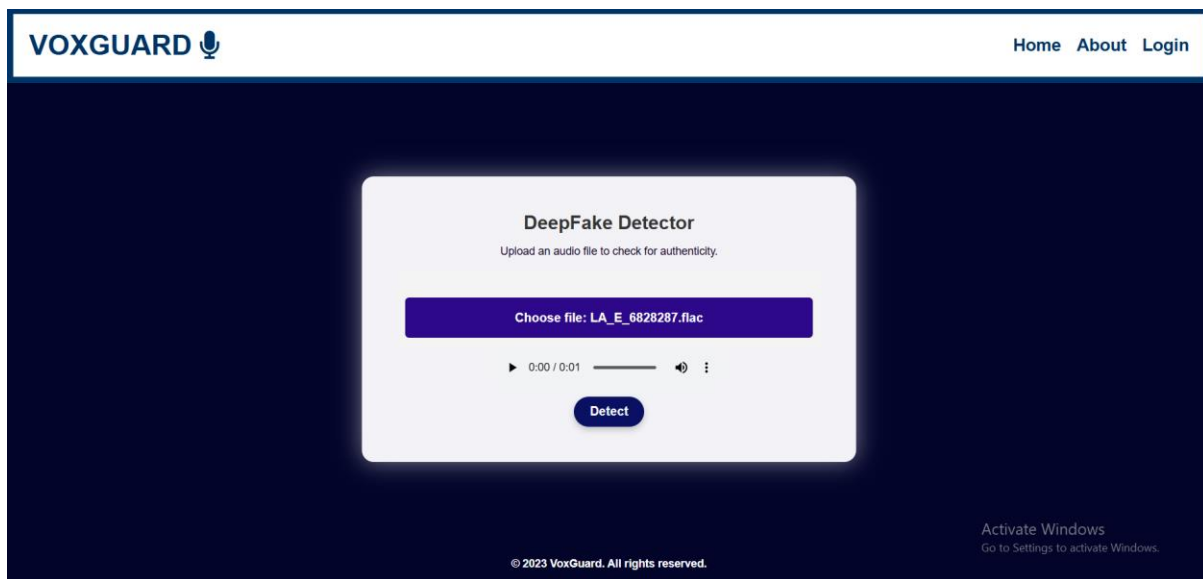


Figure 4.5: After Uploading Audio

4.1.5. RESULT PAGE

After clicking 'Detect' on the upload page, the result page opens. It shows the uploaded audio along with the detection result, indicating whether the audio is real or fake. An audio message also plays to state the result. If the audio is fake, the page looks like this:

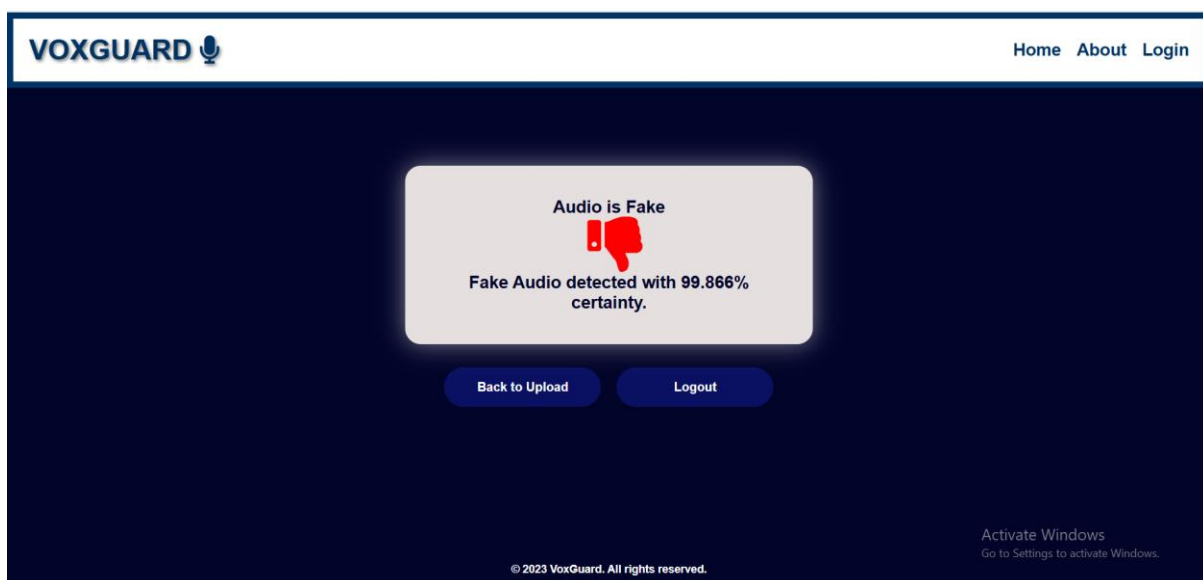


Figure 4.6: Fake Audio

And if the audio is real, the page looks like this:

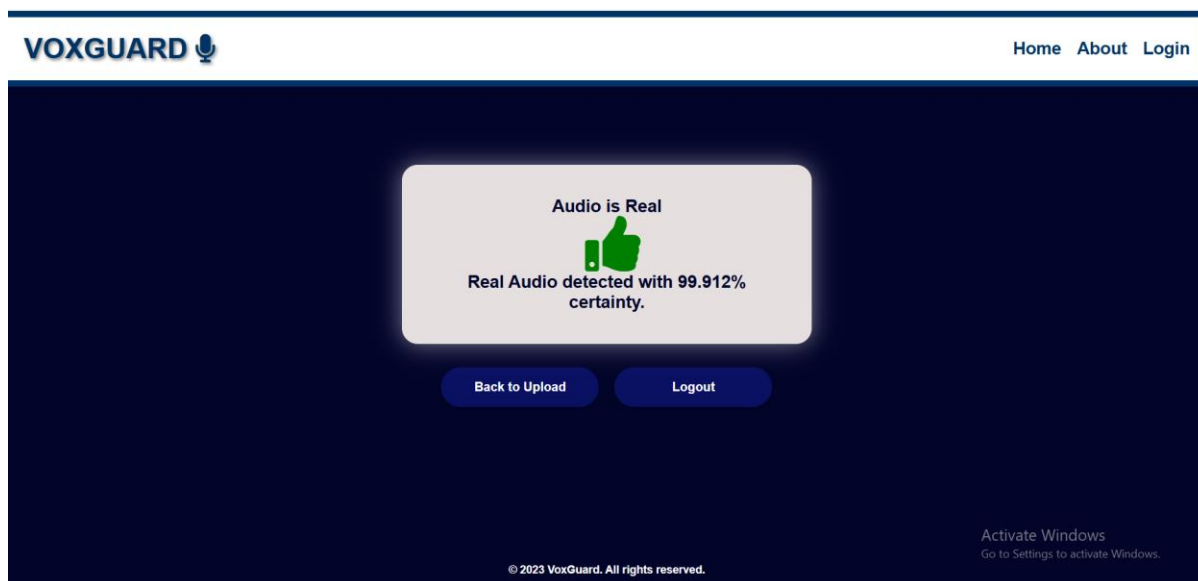


Figure 4.7: Real Audio

After detecting the audio, you can click 'Back to Upload' to submit another audio file for detection, or click 'Logout' to exit the system. If you choose to log out, your session will be deleted

4.1.6. ABOUT PAGE

This is the about page of our website.

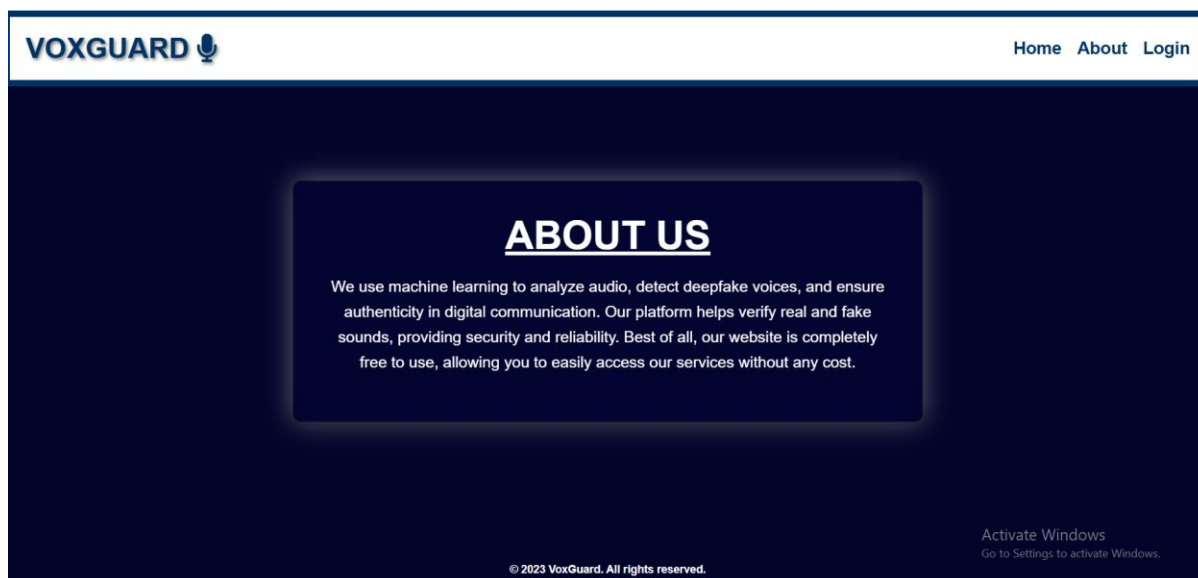


Figure 4.8: About Page

4.1.7. DATABASE

We created a database in MongoDB Compass named 'deepfake' to store user information such as username, email, and password during registration. When a user logs out, their data is deleted from the database. It looks like this:

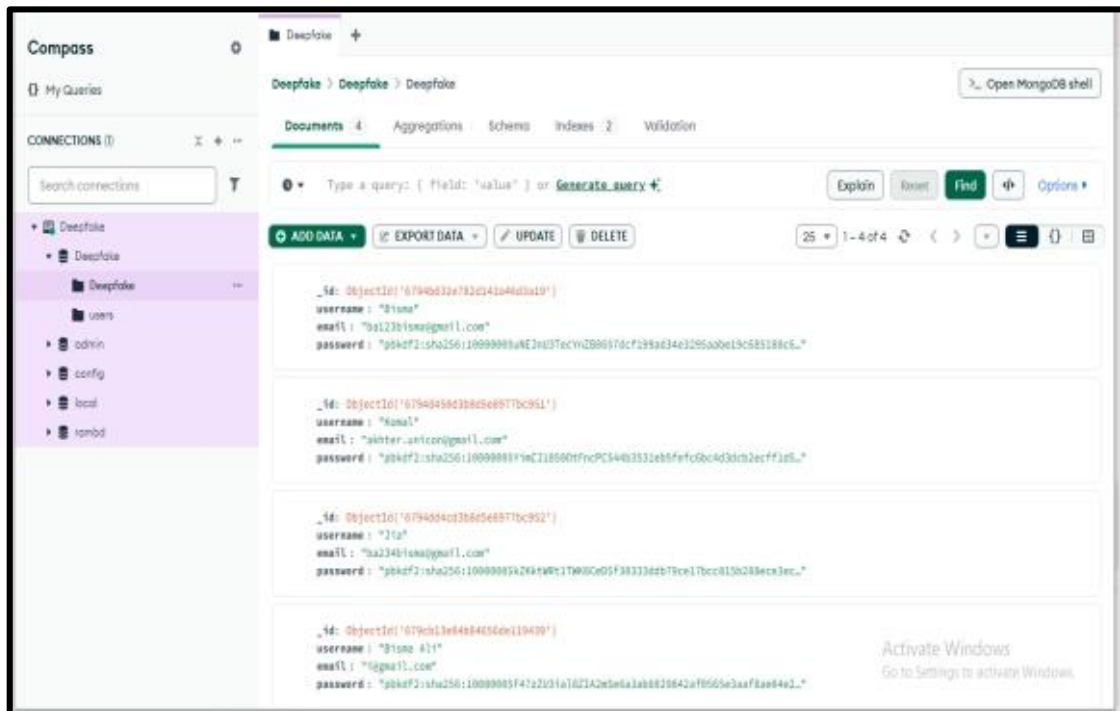


Figure 4.9: Database

4.2. TESTING

Conducted thorough testing and debugging of both frontend and backend components to ensure functionality and performance. Here are the test cases.

TEST CASE FOR LOGIN

Table 4.1: Login (TC-01)

Test Case ID	TC-01
Test Case Name	Test Case for Testing Login Screen
Test Description	Verify login functionality with valid credentials.
Pre-Condition	User is already registered.
Steps to Execute	1. Navigate to the login page. 2. Enter valid email and password. 3. Click the login button.
Post-Condition	User is logged in and redirected to upload page.
Expected Result	User is logged in successfully.
Actual Result	User is logged in successfully.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.1. TEST CASE FOR INVALID LOGIN ATTEMPT

Table 4.2: Invalid login attempt (TC-02)

Test Case ID	TC-02
Test Case Name	Test Case for Invalid Login Attempt
Test Description	Verify login functionality with invalid credentials.
Pre-Condition	User is not logged in.
Steps to Execute	1. Navigate to the login page. 2. Enter invalid email or password. 3. Click the login button.
Post-Condition	User remains on the login page.
Expected Result	Login fails with error message.
Actual Result	Login fails with error message.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.2. TEST CASE FOR USER REGISTRATION

Table 4.3: User registration (TC-03)

Test Case ID	TC-03
Test Case Name	Test Case for User Registration
Test Description	Verify registration functionality with valid details.
Pre-Condition	User is not registered.
Steps to Execute	1. Navigate to the registration page. 2. Fill in all the required fields. 3. Click the register button.
Post-Condition	User is registered in database.
Expected Result	User is registered successfully.
Actual Result	User is registered successfully.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.3. TEST CASE FOR DUPLICATE EMAIL REGISTRATION

Table 4.4: Duplicate email registration (TC-04)

Test Case ID	TC-04
Test Case Name	Test Case for Duplicate Email Registration
Test Description	Verify error handling when registering with an already registered email.
Pre-Condition	User with the email already exists.
Steps to Execute	1. Navigate to registration page. 2. Fill the form with existing email. 3. Click register button.
Post-Condition	User is not registered again.
Expected Result	Error message shown for duplicate email.
Actual Result	Error message shown for duplicate email.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.4. TEST CASE FOR PASSWORD MISMATCH IN REGISTRATION

Table 4.5: Password mismatch in registration (TC-05)

Test Case ID	TC-05
Test Case Name	Test Case for Password Mismatch in Registration
Test Description	Verify that password mismatch is handled during registration.
Pre-Condition	User not registered.
Steps to Execute	1. Navigate to registration page. 2. Enter mismatched passwords. 3. Click register.
Post-Condition	User is not registered.
Expected Result	Error shown for password mismatch.
Actual Result	Error shown for password mismatch.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.5. TEST CASE FOR UPLOADING VALID AUDIO FILE

Table 4.6: Uploading valid audio file (TC-06)

Test Case ID	TC-06
Test Case Name	Test Case for Uploading Valid Audio File
Test Description	Verify system behavior when a valid audio file is uploaded.
Pre-Condition	User is logged in.
Steps to Execute	1. Go to upload page. 2. Upload a valid audio file. 3. Submit the file.
Post-Condition	User gets a result of deepfake or real audio.
Expected Result	Audio is processed and classified correctly.
Actual Result	Audio is processed and classified correctly.

Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.6. TEST CASE FOR UPLOADING INVALID FILE TYPE

Table 4.7: Invalid audio file (TC-07)

Test Case ID	TC-07
Test Case Name	Test Case for Uploading Invalid File Type
Test Description	Check error handling for unsupported audio format.
Pre-Condition	User is logged in.
Steps to Execute	<ol style="list-style-type: none"> 1. Go to upload page. 2. Upload unsupported file format (e.g. .txt). 3. Submit the file.
Post-Condition	File is not accepted.
Expected Result	Error message displayed for invalid format.
Actual Result	Error message displayed for invalid format.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.7. TEST CASE FOR AUDIO FILE WITH NO VOICE CONTENT

Table 4.8: Audio with no voice content (TC-08)

Test Case ID	TC-08
Test Case Name	Test Case for Audio File with No Voice Content
Test Description	Verify behavior when silent or noisy audio file is uploaded.
Pre-Condition	User is logged in.

Steps to Execute	1. Go to upload page. 2. Upload an audio file with no speech. 3. Submit the file.
Post-Condition	System tries to process file.
Expected Result	System returns error or low confidence result.
Actual Result	Low confidence result returned.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.8. TEST CASE FOR UPLOAD WITHOUT LOGIN

Table 4.9: Upload without login (TC-09)

Test Case ID	TC-09
Test Case Name	Test Case for Upload Without Login
Test Description	Ensure access control on upload route.
Pre-Condition	User is not logged in.
Steps to Execute	1. Try to access /upload directly.
Post-Condition	User redirected to login page.
Expected Result	Redirected with flash message.
Actual Result	Redirected with flash message.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.9. TEST CASE FOR FAKE AUDIO DETECTION

Table 4.10: Fake audio (TC-010)

Test Case ID	TC-10
Test Case Name	Test Case for Deepfake Audio Detection
Test Description	Verify system classifies a known deepfake audio as fake.
Pre-Condition	User is logged in and has uploaded a fake audio.
Steps to Execute	1. Go to upload page. 2. Upload known fake audio file. 3. Submit the file for detection.
Post-Condition	Detection result is displayed.
Expected Result	System classifies the file as fake.
Actual Result	System classifies the file as fake.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.10. TEST CASE FOR REAL AUDIO DETECTION

Table 4.11: Real audio (TC-011)

Test Case ID	TC-11
Test Case Name	Test Case for Real Audio Detection
Test Description	Verify system classifies a known real audio correctly.
Pre-Condition	User is logged in and has uploaded a real audio.
Steps to Execute	1. Go to upload page. 2. Upload real voice recording. 3. Submit the file for detection.
Post-Condition	Detection result is displayed.
Expected Result	System classifies the file as real.
Actual Result	System classifies the file as real.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.11. TEST CASE FOR AUDIO PROCESSING TIME

Table 4.12: Audio processing time (TC-012)

Test Case ID	TC-12
Test Case Name	Test Case for Audio Processing Time
Test Description	Verify that system returns detection results within the expected time limit.
Pre-Condition	User is logged in and uploads any audio.
Steps to Execute	1. Upload an audio file. 2. Measure time from upload to result display.
Post-Condition	Result is shown.
Expected Result	Result displayed within 5 seconds.
Actual Result	Result displayed within 5 seconds.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.12. TEST CASE FOR BROWSER COMPATIBILITY

Table 4.13: Browser Compatibility (TC-013)

Test Case ID	TC-13
Test Case Name	Test Case for Browser Compatibility
Test Description	Verify that the application works across different browsers.
Pre-Condition	System is deployed and accessible.
Steps to Execute	1. Open app on Chrome, Firefox, Edge. 2. Login and perform upload.
Post-Condition	Detection works across all tested browsers.
Expected Result	System works without issue on all browsers.
Actual Result	System works without issue on all browsers.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.13. TEST CASE FOR LOGGING OUT

Table 4.14: Logging out (TC-014)

Test Case ID	TC-14
Test Case Name	Test Case for Logging Out
Test Description	Verify that user can logout successfully.
Pre-Condition	User is logged in.
Steps to Execute	1. Click on logout button. 2. Confirm the logout.
Post-Condition	Session is cleared.
Expected Result	User is logged out and redirected to home.
Actual Result	User is logged out and redirected to home.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.2.14. TEST CASE FOR SESSION EXPIRY AFTER LOGOUT

Table 4.15: Session expiry after logout (TC-015)

Test Case ID	TC-15
Test Case Name	Test Case for Session Expiry After Logout
Test Description	Verify user session is cleared after logout.
Pre-Condition	User is logged in.
Steps to Execute	1. Logout from application. 2. Try accessing upload page directly.
Post-Condition	User redirected to login.
Expected Result	Access denied without login.
Actual Result	Access denied without login.
Status	Pass
Test Conducted By	Bisma Ali
Test Date	April 17, 2025

4.3. IMPLEMENTATION RESULTS

The Vox Guard system allows registered users to upload audio files in various formats, including MP3, WAV, FLAC, AIFF, OPUS, AMR, and AAC. Upon upload, the system extracts critical audio features using the Librosa library:

- Mel Frequency Cepstral Coefficients (MFCC)
- Mel Spectrogram
- Chroma Features
- Zero-Crossing Rate
- Spectral Centroid
- Spectral Flatness

These features are concatenated into a feature vector representing the audio's characteristics. The system calculates the Euclidean distance between this vector and pre-labeled audio instances from the existing dataset. The label of the nearest neighbor is assigned to the uploaded file, and a confidence score is generated based on the closest distance compared to total distances. This score is then shown to the user, indicating whether the audio is classified as "Real" or "Fake."

For example, when a deepfake audio file was uploaded, the system successfully classified it as "Fake" with a confidence score of 94.2%. Similarly, real audio samples were detected accurately with high certainty.

4.3.1. VISUAL ANALYSIS OF AUDIO FEATURES

To gain deeper insights into the distinguishing characteristics of real and fake audio samples, various visualizations were employed using Matplotlib in the Colab notebook. These visualizations help in understanding how different audio features contribute to the classification process.

4.3.2. WAVEFORMS

A waveform is a graph that shows how the loudness (amplitude) of an audio signal changes over time.

For real audio:

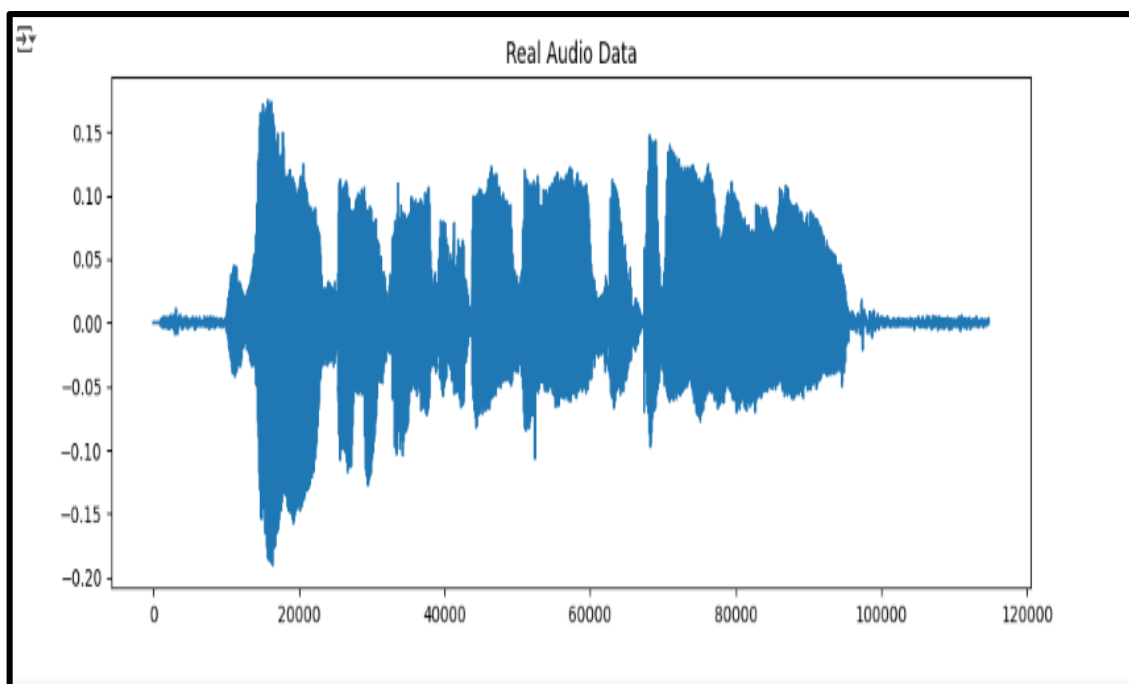


Figure 4.10: Waveform for real audio

The graph displayed is a waveform representation of a real audio signal. It visually shows how the amplitude (or loudness) of the sound varies over time.

- The horizontal axis (X-axis) represents time, showing how the audio progresses from start to end.
- The vertical axis (Y-axis) represents amplitude, which reflects the loudness or energy of the audio at each moment.

The ups and downs in the waveform indicate fluctuations in the sound. Different sound volumes correlate with spike height where loud sounds result from increased amplitude and silent or soft sounds generate flat regions in the waveform. The visual display provides essential insights into how an audio document organizes itself by illustrating both the speech patterns and the duration of pauses as well as the volume intensity over time.

Using waveform displays provides a fundamental tool for checking audio patterns during deepfake audio detection procedures. The natural variability present in genuine audio recordings usually contrasts with the artificial features of fake audio that should demonstrate smooth sounding tones and repeated patterns.

For fake audio:

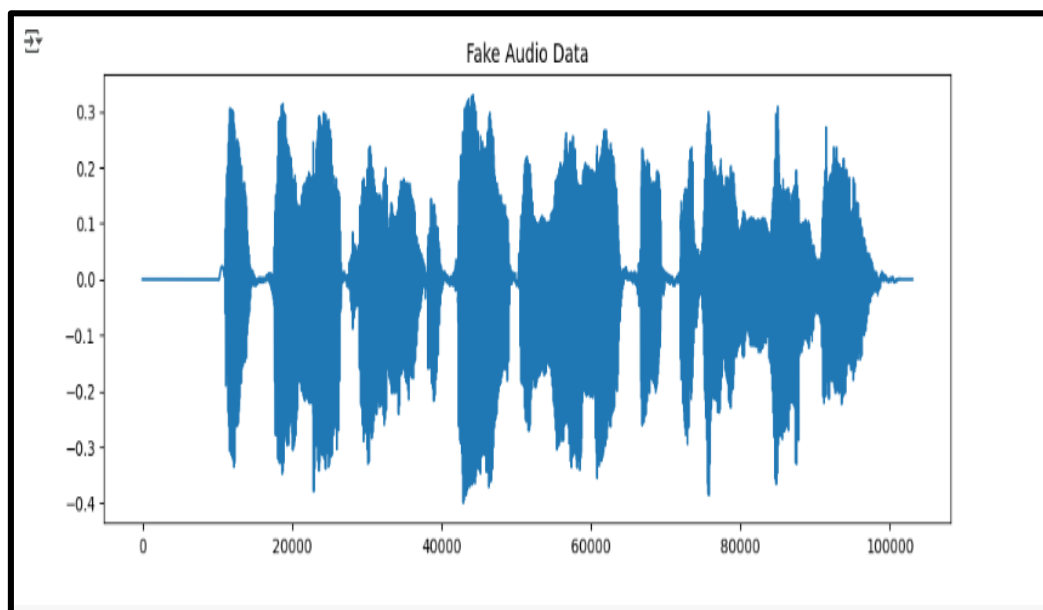


Figure 4.11: Waveform for fake audio

The graph displayed is a waveform representation of a fake or synthetically generated audio signal. It visually shows how the amplitude (or loudness) of the sound varies over time.

- The horizontal axis (X-axis) represents time, showing how the audio progresses from start to end.
- The vertical axis (Y-axis) represents amplitude, which reflects the loudness or energy of the audio at each moment.

The ups and downs in the waveform indicate fluctuations in the sound. In this fake audio, the waveform may exhibit more consistent or repetitive patterns compared to natural speech. Some sections may appear overly smooth or too uniform, and transitions might be more abrupt. The beginning or ending of the audio might also look unnaturally flat, which is uncommon in real recordings.

In the context of audio detection, this waveform can provide a visual clue for identifying synthetic speech. While it's not always conclusive, fake audio often lacks the subtle variations and imperfections found in real human speech such as irregular pauses, background noise, or slight tonal shifts making its waveform appear less natural overall.

4.3.3. SPECTROGRAM ANALYSIS

For real audio:

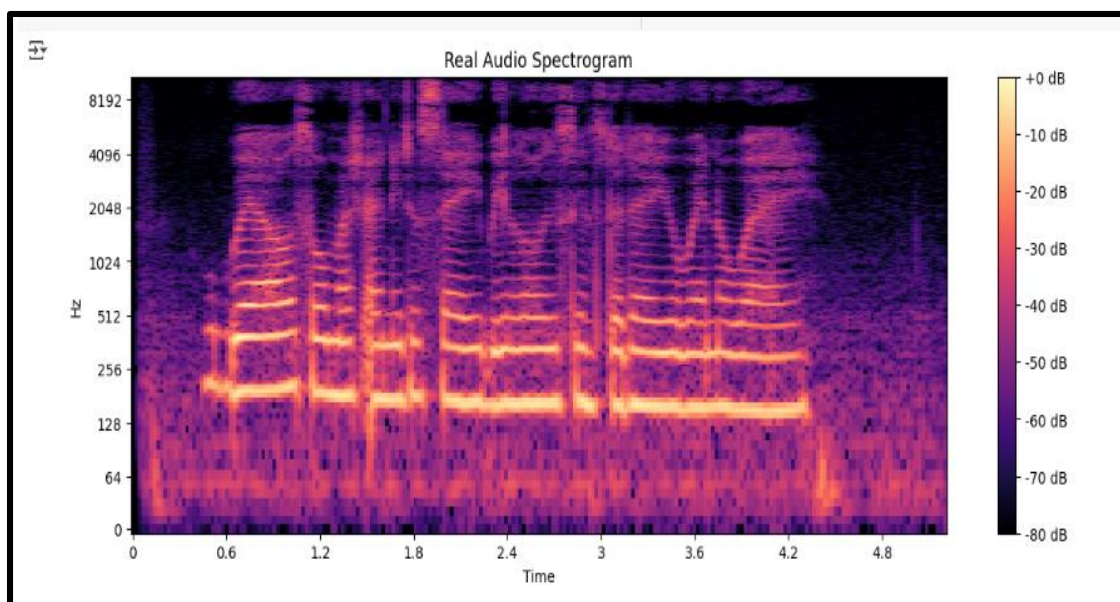


Figure 4.12: Spectrogram for real audio

The graph displayed is a spectrogram representation of a real audio signal. Unlike a waveform that only displays amplitude over time, a spectrogram provides a much deeper view by visualizing how the frequency content of the audio signal evolves over time.

- The horizontal axis (X-axis) represents time, showing how the audio progresses from start to end.
- The vertical axis (Y-axis) represents frequency in Hertz (Hz), ranging from low (bottom) to high (top) frequencies.

The color intensity (from dark purple to yellow) indicates the amplitude or strength of the frequencies at each time point, measured in decibels (dB). Brighter areas (yellow/white) show stronger or louder frequencies, while darker areas (purple/black) indicate weaker signals.

In this real audio spectrogram, you can observe rich and natural patterns of harmonics and frequency transitions. These patterns often reflect natural speech characteristics, such as formants (resonant frequencies of the vocal tract) and intonation changes. The presence of complex, non-repetitive frequency bands over time typically suggests the audio is naturally recorded rather than artificially generated.

In deepfake audio detection, a spectrogram is especially useful for identifying subtle anomalies in synthetic audio that might not be visible in waveforms. Real audio tends to have smooth and naturally varying spectral features.

For fake audio:

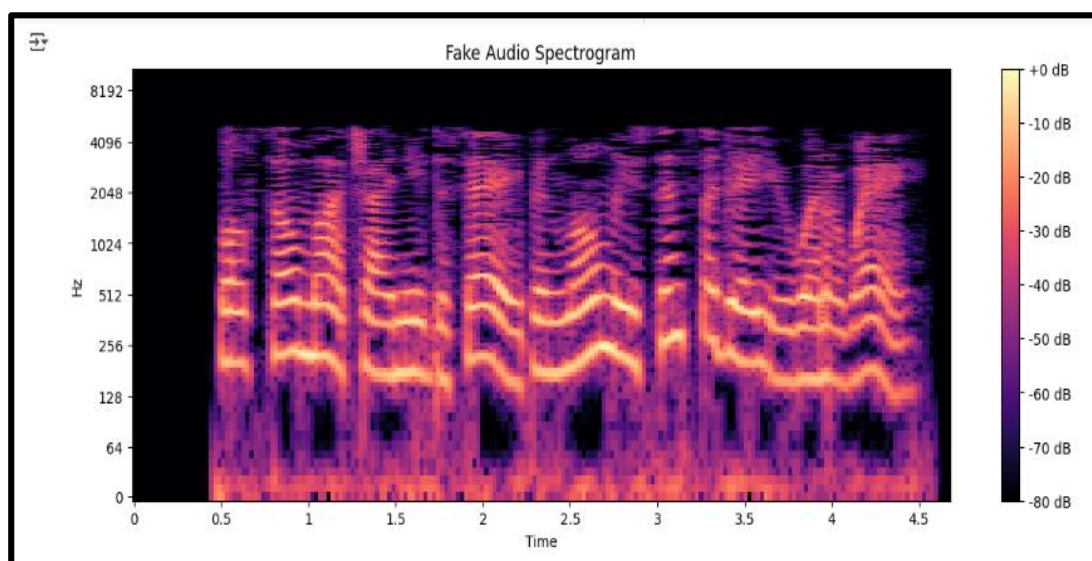


Figure 4.13: Spectrogram for fake audio

The graph displayed is a spectrogram representation of a fake audio signal. Unlike a waveform that only displays amplitude over time, a spectrogram provides a much deeper view by visualizing how the frequency content of the audio signal evolves over time.

- The horizontal axis (X-axis) represents time, showing how the audio progresses from start to end.
- The vertical axis (Y-axis) represents frequency in Hertz (Hz), with a logarithmic scale that better matches human hearing perception, ranging from low (bottom) to high (top) frequencies.

The color intensity (from dark purple to yellow) indicates the amplitude or strength of the frequencies at each time point, measured in decibels (dB). Brighter areas (yellow/white) show stronger or louder frequencies, while darker areas (purple/black) indicate weaker or absent signals.

In this fake audio spectrogram, you can observe characteristics that differ from natural recordings. Synthetic audio often exhibits more uniform frequency distributions, less natural harmonic transitions. The spectral patterns may appear less varied compared to real speech or music, with certain frequency bands showing unnatural consistency over time.

4.3.4. MEL SPECTROGRAM

For Real Audio

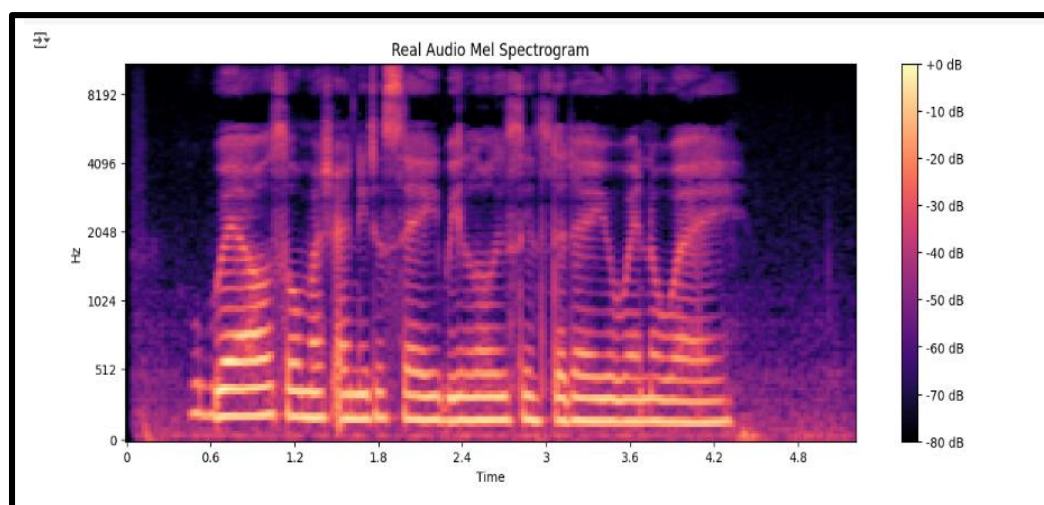


Figure 4.14: Mel spectrogram for real audio

This graph shows a Mel spectrogram of a real audio signal. The horizontal axis represents time (0-5 seconds), while the vertical axis shows frequency on the Mel scale (512Hz to 8192Hz), which matches human hearing. Colors indicate signal strength from 0 dB (purple) to 80 dB (yellow).

The visualization reveals natural audio characteristics: smooth frequency transitions, consistent harmonics, and gradual volume changes. Unlike synthetic audio, it lacks abrupt jumps or artificial patterns. This Mel spectrogram (using 128 frequency bins) is particularly useful for analyzing speech and music, as its frequency scaling emphasizes the ranges most important for human hearing. The organic patterns confirm this is genuine recorded audio.

For fake audio:

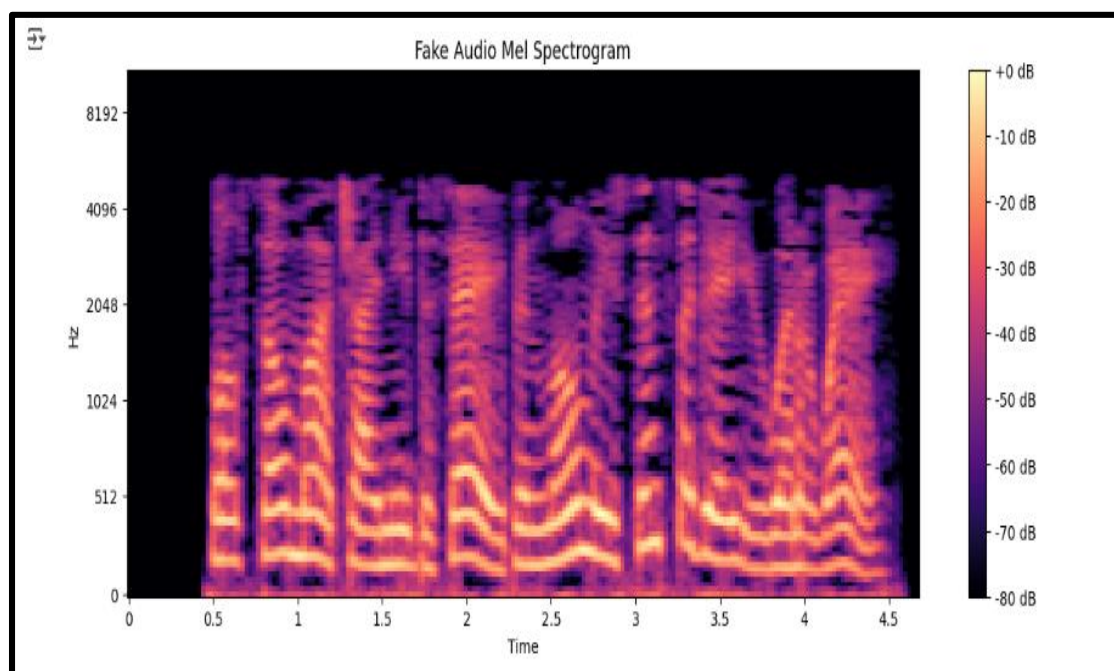


Figure 4.15: Mel spectrogram for fake audio

This graph shows a mel spectrogram of a fake audio signal. The horizontal axis represents time (0-4.5 seconds), while the vertical axis shows frequency on the mel scale (0Hz to 8192Hz), which approximates human hearing. Colors indicate signal strength from 0 dB (dark purple) to maximum amplitude (bright yellow).

The visualization reveals synthetic audio characteristics: less smooth frequency transitions, more uniform harmonic patterns, and potential abrupt energy changes. Unlike natural audio, it may show subtle artifacts or inconsistencies in spectral patterns. This mel spectrogram (likely using 128 frequency bins) helps identify artificial audio by highlighting frequency ranges

important for human hearing. The patterns suggest this is computer-generated rather than naturally recorded audio.

4.3.5. CHROMAGRAM

For real audio:

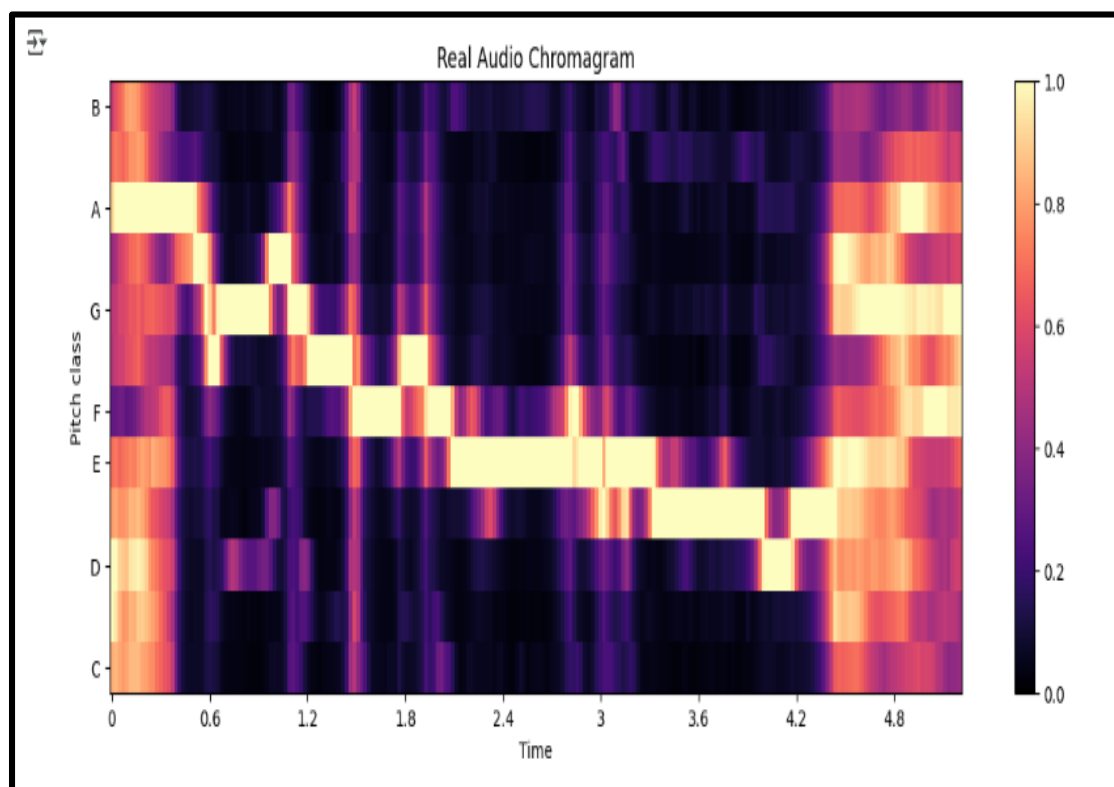


Figure 4.16: Chromagram for real audio

This graph shows a chromagram representation of a real audio signal. The horizontal axis represents time (0 to 4.8 seconds), while the vertical axis displays the 12 musical pitch classes (A to H) in the chromatic scale. Color intensity shows the strength of each pitch over time, with darker shades indicating weaker presence and brighter colors showing stronger dominance.

The visualization reveals authentic musical characteristics: smooth pitch transitions, natural harmonic relationships, and organic intensity variations. Unlike synthetic audio, this chromagram demonstrates the nuanced pitch evolution typical of real music or speech. The clear harmonic structure and coherent temporal patterns confirm this is genuine audio content. The chromagram is particularly useful for musical analysis as it maps audio directly to the 12-tone musical scale.

For fake audio:

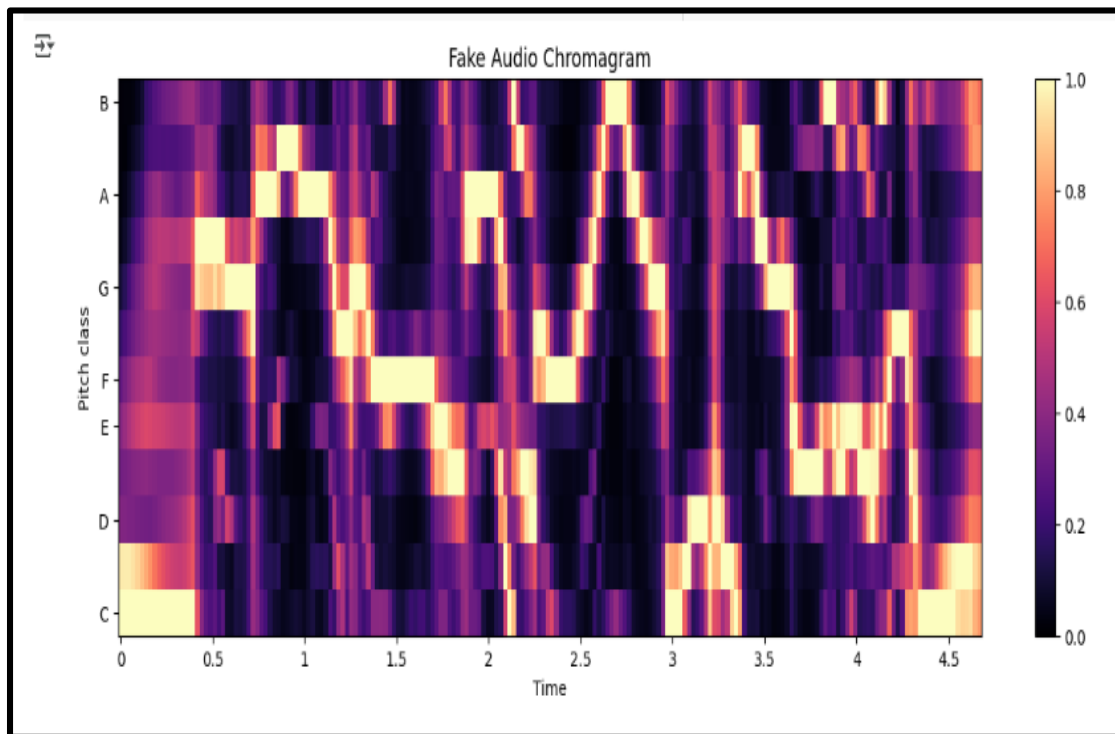


Figure 4.17: Chromagram for fake audio

This graph shows a chromagram representation of a fake audio signal. The horizontal axis represents time (0 to 4.5 seconds), while the vertical axis displays irregular pitch labels (C, O, n, m) that deviate from standard musical notation. The intensity of pitch color during time would communicate strength levels since darker tones reflect weaker sounds while bright tones reveal dominant pitches.

Unusual pitch variations with irregular harmonics appear as synthetic audio characteristics in the visualization. The chromagram presents sudden shifts along with man-made pitch distributions which indicate computer-generated music rather than natural acoustic music. The non-standard pitch labeling system (C, O, n, m) brings attention to the fact that this content was made using artificial methods. While chromagrams normally help analyze authentic music by mapping to the 12-tone scale, this visualization's irregularities confirm it represents manipulated or synthesized audio.

4.3.6. MFCCs

For real audio:

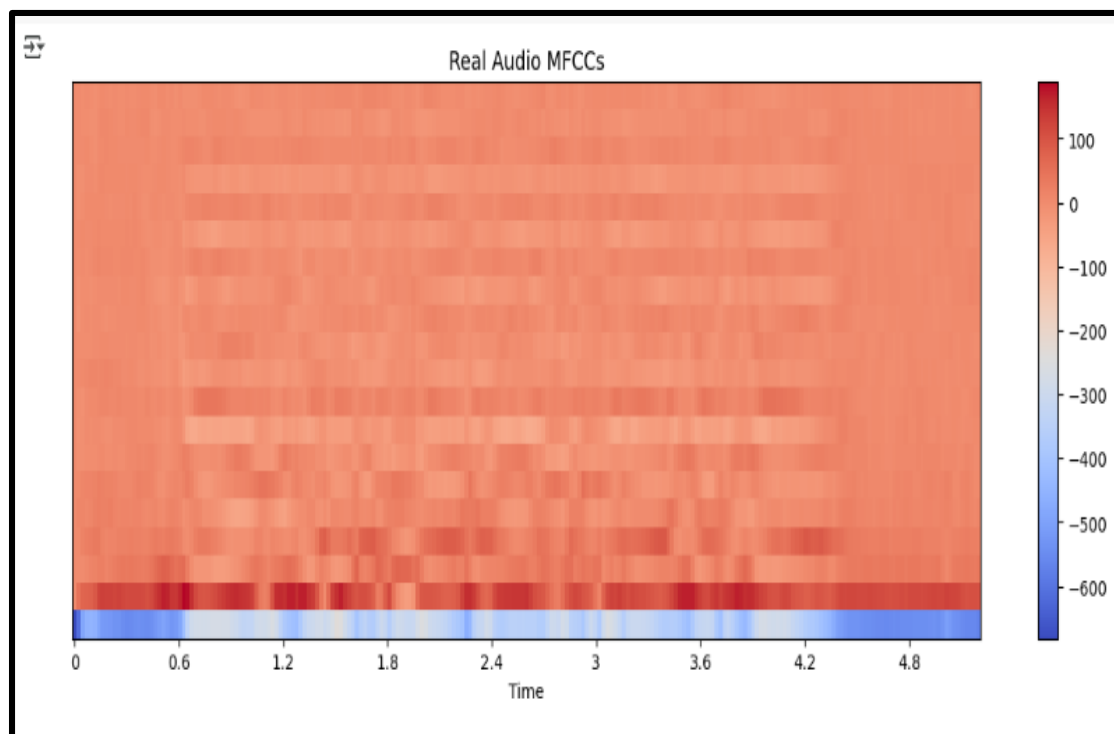


Figure 4.18: MFCCs for real audio

This graph shows an MFCC (Mel-Frequency Cepstral Coefficients) representation of a real audio signal. The horizontal axis represents time (0 to 4.8 seconds), while the vertical axis displays the MFCC coefficients that characterize the timbral texture of the audio. Color intensity shows the strength of each coefficient over time, with darker shades indicating weaker presence and brighter colors showing stronger dominance.

The visualization reveals authentic audio characteristics through smooth coefficient transitions and natural timbral variations. Unlike synthetic audio, these MFCCs demonstrate the nuanced spectral evolution typical of real speech or music. The consistent patterns and gradual coefficient changes confirm this is genuine audio content. MFCCs are particularly useful for voice analysis as they effectively represent the perceptually relevant aspects of sound.

For fake audio:

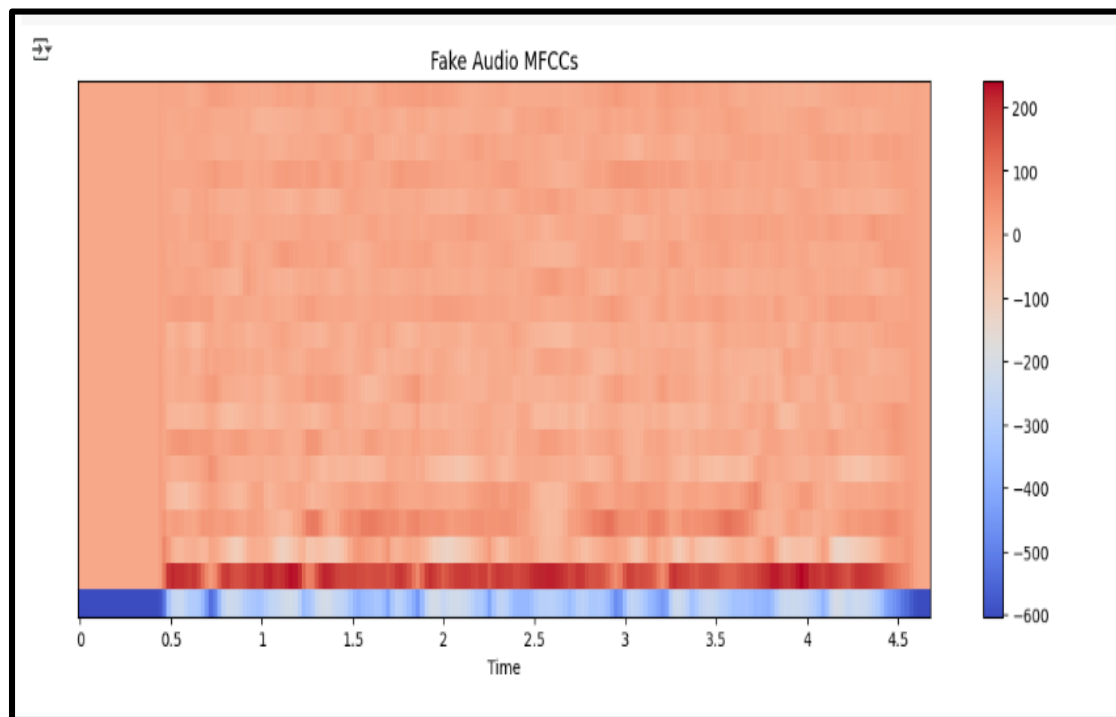


Figure 4.19: MFCCs for fake audio

This graph shows an MFCC (Mel-Frequency Cepstral Coefficients) representation of a fake audio signal. The horizontal axis represents time (duration not fully specified), while the vertical axis displays the MFCC coefficients (shown in values from -600 to +200) that characterize the artificial timbral qualities of the audio. The numerical values represent the strength of each coefficient, with extreme positive and negative values indicating exaggerated or unnatural spectral features.

The visualization reveals synthetic audio characteristics through abrupt coefficient transitions and artificial timbral patterns. Unlike natural audio, these MFCCs demonstrate the rigid spectral evolution typical of computer-generated speech or music. The irregular patterns and non-gradual coefficient changes suggest artificial audio content. The spectral analysis technique using MFCC reveals artificial characteristics which cannot be detected when observing audio waveforms.

The presence of extreme (+200 and -600) coefficient values shows the unnatural spectral behavior which occurs when generating audio synthetically by strengthening or limiting certain frequency bands. The artificial modifications of spectrum patterns during synthesis produce the robotic and hollow sound characteristic of most speech generation systems.

4.4. DISCUSSION

The system performs trustworthily on evaluation data by effectively comparing features to differentiate real from fake audio. Use of this approach enables an efficient system since the system excludes time-consuming and expensive model training processes. It depends on extracted audio features to operate thus providing quick deployment capabilities and platform integration convenience.

The main benefit of this system involves its combination with the MongoDB authentication system. The system provides secure login procedures by verifying individual audio data separation between the storage area and personal information space resulting in secure access to user credentials and outcomes. The dual focus on privacy enhancement and usability improvement makes the platform more acceptable to users.

The system depends heavily on the detection precision because its capability relies on the testing standards and data diversity of the training materials. The system demonstrates performance holes during actual operations because real-life audio inputs have variable quality characteristics and speaker accents as well as diverse recording environments. The robustness and generalization capabilities of the system can be improved by adding more real and fake audio samples especially difficult-to-detect and extreme-case audio inputs to its dataset. The system would achieve better consistent results across various real-world implementations through this protocol.

Chapter-5

CONCLUSION AND FUTURE WORK

5. CONCLUSION AND FUTURE WORK

This chapter provides conclusion, and future work for Vox Guard. The chapter highlights the current capabilities of the system to detect deepfake audio and outlines future developments to enhance its functionality and user experience. Summary highlights Vox Guard's transformative impact on the security of audio communication, its advanced features delivering real-time classification results, and its commitment to continued user-centered growth on detection accuracy. The conclusion reflects the role of Vox Guard in improving media integrity, gaining trust, and contributing to better outcomes through its well-designed processing and advanced detection techniques.

The user-friendly audio detection platform Vox Guard enables users to validate audio files through its upload functionality. The system handles rising audio misinformation dangers through advanced feature analysis of uploaded voice recordings which includes MFCCs and mel spectrograms and chroma and zero-crossing rate and spectral centroid and flatness. The extraction of audio features occurs through Librosa to evaluate them against a pre-existing dataset using distance techniques for authenticating real or fake audio content. Users find complete clarity regarding detection outcomes because the program shows both the final result alongside a percentage indicator for confidence levels.

User authentication management through MongoDB maintains a secure system where registered can detect audios. Flask development enables the system to provide an intuitive web interface which supports standard audio formats. The platform basically refuses to store any submitted media files as well as all detection findings while keeping account information private. From the start of login to the display of results the application follows a straightforward process. System enhancements in development focus on both richer feature capabilities and better visualization features in addition to processing speed improvements for detection efficiency.

The detection system operates with high speed by finishing its operations in less than five second. Through its system the platform accepts several file types while users receive immediate feedback about their uploads during the process. System alerts assist users with progress throughout the application execution. The secure and prompt authentication provided by VoxGuard remains available to everyone seeking fake audio identification services.

5.1. CONCLUSION

The release of Vox Guard represents a crucial leap in responding to the increasing threats of digital forgery today. While deep learning technologies are employed to generate more realistic imitation content, there arises an urgent need for efficient systems that can identify and classify manipulated information. Vox Guard provides a technically robust and effective solution by aligning strong feature extraction techniques with distance-based comparison methodology to ensure authentic verification of digital content.

The system is established on a solid base of machine learning and data analysis concepts. It learns significant patterns from computer files and matches them with a validated dataset to establish authenticity. This fast, lightweight method shows how clever algorithms can be employed to solve challenging forgery detection issues without compromise on performance and interpretability.

In addition to its technical advantages, Vox Guard also emphasizes users' privacy and security by incorporating secure database management for authenticating users. The system's simple yet elegant design opens it up to a broad spectrum of users without regard to technical experience. The project strikes an effective harmony of technical complexity and simplicity in that the solution is both effective and functional.

Finally, Vox Guard helps achieve the greater purpose of secure trust in online communication. As human beings and institutions increasingly rely on online platforms, there is a greater need to establish the authenticity of provided information. Vox Guard addresses this requirement by providing a strong detection tool that fosters transparency, prevents false content dissemination, and encourages responsible online engagement. The system establishes the basis for future improvements in forged digital content detection and illustrates how deep learning can be used to solve real-world issues with real-world impact

5.2. FUTURE WORK

This section suggests potential areas for further research and development. It outlines how the current work can be expanded upon in the future.

- **Real-Time Detection Capabilities**

Enhancing the system to support real-time or near real-time audio verification to detect fake content during live calls, meetings, or streams.

- **Enhanced Data Analysis Algorithms**

Developing algorithms to analyze audio features more deeply for highly accurate and personalized detection feedback.

- **Open-Source Collaboration Platform**

Establishing an open-source Vox Guard platform to promote collaboration and innovation among developers and researchers.

- **Refined Activity Tracking and Mobile Support**

Improving detection algorithms for precise tracking and extending functionality to cross-platform mobile applications.

- **Expanded Dataset and Comparison Network**

Broadening the dataset and reference audio base used in detection to improve generalization across a wide range of voice inputs.

- **Advanced UI/UX Refinement**

The interface will undergo repeated improvement to create a platform-independent interface which provides seamless user experience and immediate operation.

- **Advanced Search Capabilities**

Vox Guard needs to enhance its search and filtering capabilities to improve both speed and precision in audio service and data retrieval.

REFERENCES

- [1] A. Hamza *et al.*, “Deepfake Audio Detection via MFCC Features Using Machine Learning,” *IEEE Access*, vol. 10, pp. 134018–134028, 2022, doi: <https://doi.org/10.1109/access.2022.3231480>.
- [2] Z. Almutairi and H. Elgibreen, “A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions,” *Algorithms*, vol. 15, no. 5, p. 155, May 2022, doi: <https://doi.org/10.3390/a15050155>.
- [3] O. A. Shaaban, Remzi Yıldırım, and Abubaker Alguttar, “Audio Deepfake Approaches,” *IEEE Access*, vol. 11, pp. 132652–132682, Jan. 2023, doi: <https://doi.org/10.1109/access.2023.3333866>.
- [4] Y. Xie *et al.*, “The Codecfake Dataset and Countermeasures for the Universally Detection of Deepfake Audio,” *IEEE Transactions on Audio Speech and Language Processing*, pp. 1–14, Jan. 2025, doi: <https://doi.org/10.1109/taslpro.2025.3525966>.
- [5] Mouna Rabhi, Spiridon Bakiras, and Roberto Di Pietro, “Audio-deepfake detection: Adversarial attacks and countermeasures,” *Expert systems with applications*, pp. 123941–123941, Apr. 2024, doi: <https://doi.org/10.1016/j.eswa.2024.123941>.
- [6] L. Pham, P. Lam, T. Nguyen, H. Nguyen, and A. Schindler, “Deepfake Audio Detection Using Spectrogram-based Feature and Ensemble of Deep Learning Models,” *arXiv (Cornell University)*, pp. 1–5, Sep. 2024, doi: <https://doi.org/10.1109/is262782.2024.10704095>.
- [7] A. Chintha *et al.*, “Recurrent Convolutional Structures for Audio Spoof and Video Deepfake Detection,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 1024–1037, Aug. 2020, doi: <https://doi.org/10.1109/jstsp.2020.2999185>.
- [8] J. Khochare, C. Joshi, B. Yenarkar, S. Suratkar, and F. Kazi, “A Deep Learning Framework for Audio Deepfake Detection,” *Arabian Journal for Science and Engineering*, Nov. 2021, doi: <https://doi.org/10.1007/s13369-021-06297-w>.
- [9] W. Yang *et al.*, “AVoiD-DF: Audio-Visual Joint Learning for Detecting Deepfake,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2015–2029, Jan. 2023, doi: <https://doi.org/10.1109/tifs.2023.3262148>.

- [10] D. Song, N. Lee, J. Kim, and E. Choi, “Anomaly Detection of Deepfake Audio Based on Real Audio using Generative Adversarial Network Model,” *IEEE Access*, pp. 1–1, Jan. 2024, doi: <https://doi.org/10.1109/access.2024.3506973>.
- [11] H. Ilyas, A. Javed, and K. M. Malik, “AVFakeNet: A unified end-to-end Dense Swin Transformer deep learning model for audio–visual deepfakes detection,” *Applied Soft Computing*, vol. 136, p. 110124, Mar. 2023, doi: <https://doi.org/10.1016/j.asoc.2023.110124>.
- [12] R. Mubarak, Tariq Alsboui, O. Alshaikh, I. Inuwa-Dutse, S. Khan, and S. Parkinson, “A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats,” *IEEE Access*, vol. 11, pp. 144497–144529, Jan. 2023, doi: <https://doi.org/10.1109/access.2023.3344653>.
- [13] Y. Wang and H. Huang, “Audio–visual deepfake detection using articulatory representation learning,” *Computer Vision and Image Understanding*, vol. 248, p. 104133, Nov. 2024, doi: <https://doi.org/10.1016/j.cviu.2024.104133>.
- [14] Anfal Alshehri, Danah Almalki, E. Alharbi, and Somayah Albaradei, “Audio Deep Fake Detection with Sonic Sleuth Model,” *Computers*, vol. 13, no. 10, pp. 256–256, Oct. 2024, doi: <https://doi.org/10.3390/computers13100256>.
- [15] G. Lee *et al.*, “Dual-Channel Deepfake Audio Detection: Leveraging Direct and Reverberant Waveforms,” *IEEE Access*, vol. 13, pp. 18040–18052, 2025, doi: <https://doi.org/10.1109/access.2025.3532775>.