

NOISE REDUCTION FROM AUDIO

A

Report on Project Stage-I

*Submitted in partial fulfillment of
the requirements for the award of the degree of*

Bachelor of Technology in Computer Science and Engineering

Submitted by

Badavath Jaipal	(21SS1A0506)
Chakrala Saikiran	(21SS1A0507)
Danturi Nithish	(21SS1A0511)
Devalla Sai Sathwik	(21SS1A0513)

Under the guidance of

Ms.Shumama Ansa

Assistant. Professor(C)



Department of Computer Science and Engineering
JNTUH University College of Engineering Sultanpur
Sultanpur (V), Pulkal (M), Sangareddy (Dist), Telangana-502273

February 2025

**JNTUH UNIVERSITY COLLEGE OF ENGINEERING
SULTANPUR**

Sultanpur(V), Pulkal(M), Sangareddy-502273, Telangana



Department of Computer Science and Engineering

Certificate

This is to certify that the Project Stage-I Report work entitled ”**NOISE REDUCTION FROM AUDIO**” is a bonafide work carried out by a team consisting of **Badavath Jaipal** bearing Roll no.**21SS1A0506**, **Chakrala Saikiran** bearing Roll no.**21SS1A0507**, **Danturi Nithish** bearing Roll no.**21SS1A0511**, **Devalla Sai Sathwik** bearing Roll no.**21SS5A0513**, in partial fulfillment of the requirements for the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING discipline to Jawaharlal Nehru Technological University Hyderabad University College of Engineering Sultanpur during the academic year 2024-2025.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Guide

Ms.Shumama Ansa

Assistant. Professor(C)

Head of the Department

Dr. G. Narsimha

Professor, Principal & Incharge HOD of CSE

External Examiner

Declaration

We hereby declare that Project Stage-I entitled “**NOISE REDUCTION FROM AUDIO**” is a bonafide work carried out by a team consisting of **Badavath Jaipal** bearing Roll no.**21SS1A0506**, **Chakrala Saikiran** bearing Roll no.**21SS1A0506**, **Danturi Nithish** bearing Roll no.**21SS1A0511**,**Devalla Sai Sathwik** bearing Roll no.**20SS5A0513**, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering discipline to Jawaharlal Nehru Technological University Hyderabad College of Engineering Sultanpur during the academic year 2024-2025. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Badavath Jaipal	(21SS1A0506)
Chakrala Saikiran	(21SS1A0507)
Danturi Nithish	(21SS1A0511)
Devalla Sai Sathwik	(21SS1A0513)

Acknowledgment

We wish to take this opportunity to express our deep gratitude to all those who helped us in various ways during our Project Stage-II report work. It is our pleasure to acknowledge the help of all those individuals who were responsible for foreseeing the successful completion of our Project Stage-II report.

We express our sincere gratitude to **Dr. G. Narsimha, Professor and Principal**, JNTUHUCES for his support during the course period.

We express our sincere gratitude to **Dr. Y. Raghavender Rao, Professor, Vice Principal and Head of the Department(ECE)**, JNTUHUCES for his support during the course period.

We are thankful to **Dr. G. Narsimha, Professor, Principal and In-charge Head of the Department**, Computer Science and Engineering, for his support and encouragement throughout the course period.

We are thankful to our Guide **Ms.Shumama Ansa, Assistant. Professor(C)** for his effective suggestions during the course period.

Finally, we express our gratitude with great admiration and respect to our faculty for their moral support and encouragement throughout the course.

Badavath Jaipal	(21SS1A0506)
Chakrala Saikiran	(21SS1A0507)
Danturi Nithish	(21SS1A0511)
Devalla Sai Sathwik	(21SS1A0513)

Contents

Certificate	ii
Declaration	iii
Acknowledgement	iv
Abstract	x
List of Figures	xi
1 INTRODUCTION	1
1.1 Project Overview	1
1.2 Purpose	2
1.3 Existing System	2
1.3.1 Drawbacks of Existing System	2
1.4 Proposed System	3

1.4.1	Advantages of Proposed System	3
1.5	Scope	4
1.6	Conclusion	4
2	LITERATURE SURVEY	5
2.1	Audio de-noising by spectral subtraction technique implemented on re-configurable hardware	5
2.2	The Improvement of Audio Noise Reduction System Based on LMS Algorithm	5
2.3	Improvement of audio noise reduction system based on RLS algorithm .	6
2.4	Stationary Noise Reduction	6
2.5	Conclusion	7
3	SYSTEM ANALYSIS	8
3.1	Algorithms	8
3.1.1	Spectral Gating Algorithm	8
3.2	System Architecture	9
3.3	Feasibility Study	9
3.3.1	Technical Feasibility	10
3.3.2	Operational Feasibility	10

3.3.3	Economic Feasibility	11
3.4	Hardware Requirements	11
3.5	Software Requirements	11
3.6	Conclusion	12
4	SYSTEM DESIGN	13
4.1	UML	13
4.2	Use Case Diagram	15
4.3	Activity Diagram	16
4.4	Sequence Diagram	17
4.5	Conclusion	18
5	IMPLEMENTATION	19
5.1	Frontend Implementation	19
5.1.1	Create HTML Form for File Upload:	19
5.1.2	Display Loading Spinner During File Processing:	19
5.1.3	Show Download Link After Processing:	20
5.1.4	JavaScript to Handle File Upload and Display Results:	20
5.2	Backend Implementation	21

5.2.1	Initialize Flask Application:	21
5.2.2	Route for File Upload and Processing:	21
5.2.3	Run the Flask App:	22
5.3	Model Processing (Noise Reduction Logic):	22
5.3.1	Import Required Libraries	23
5.3.2	Audio Processing Function:	23
5.4	Data Handling	23
5.4.1	Upload and Save Audio File:	23
5.4.2	Save and Serve Processed Audio:	24
5.5	Visualization in Model Development	24
5.5.1	Plot Spectrograms of Original and Processed Audio:	24
5.6	Conclusion	25
6	TESTING	26
6.1	White-Box Testing	26
6.2	Black-Box Testing	26
6.3	Unit Testing	27
6.4	Integration Testing	27
6.5	Validation Testing	27

6.6	System Testing	27
6.7	Conclusion	28
7	RESULTS	29
7.1	Start the flask app	29
7.2	Web Interface	30
7.3	Upload Audio File	30
7.4	Processing the File	31
7.5	Download processed file	31
7.6	Audio noise reduction visualization	32
7.7	Conclusion	32
	CONCLUSION	33
	REFERENCES	34

Abstract

This project focuses on reducing noise from audio signals using digital signal processing techniques. The primary goal is to improve the quality and intelligibility of audio recordings by suppressing unwanted background noise. The project utilizes the `noisereduce` library in Python, which implements a spectral gating algorithm for noise reduction. The performance of the noise reduction system is evaluated using objective metrics including Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and Peak Signal-to-Noise Ratio (PSNR). The results demonstrate a significant improvement in audio quality after applying the noise reduction algorithm. The project's findings have implications for applications such as speech recognition, audio enhancement, and audio restoration, where noise reduction plays a crucial role in achieving optimal performance and user experience.

List of Figures

3.1	<i>System Architecture</i>	9
4.1	<i>Use Case Diagram of System</i>	15
4.2	<i>Activity Diagram of System</i>	16
4.3	<i>Sequence Diagram of System</i>	17
7.1	Start the flask app	29
7.2	Web interface	30
7.3	Upload Audio File	30
7.4	Processing the File	31
7.5	Download processed file	31
7.6	Audio noise reduction visualization	32

Chapter 1

INTRODUCTION

1.1 Project Overview

Noise in audio refers to any unwanted or disruptive sound that interferes with the clarity and quality of an audio signal. This noise can originate from various sources, including environmental factors such as background chatter, traffic, or wind, as well as technical factors like electrical interference, microphone sensitivity, or recording equipment limitations. In audio recordings, noise often reduces the intelligibility of speech or music, making it difficult to extract useful information or enjoy the content.

Noise reduction plays a crucial role in overcoming these challenges by enhancing the quality of audio signals and improving their usability. By suppressing unwanted background noise, noise reduction techniques ensure that the desired audio components, such as speech or music, remain clear and intelligible. This is especially important in applications like speech recognition, telecommunication, video conferencing, and audio restoration, where clean audio signals are vital for achieving optimal performance. Moreover, noise reduction is essential for creating immersive and high-quality experiences in media production, ensuring that listeners can fully engage with the content without distractions.

1.2 Purpose

The purpose of the "Noise Reduction from Audio" project is to enhance the quality of audio recordings by removing unwanted background noise. This can be particularly useful in applications like speech recognition, audio communication, and sound recording where clarity and accuracy are crucial. The project typically aims to apply techniques such as filtering, spectral subtraction, or machine learning models to isolate and preserve the desired sound while eliminating noise, making the audio more intelligible and professional.

1.3 Existing System

Existing systems for audio noise reduction often leverage a combination of hardware and software solutions to enhance audio clarity. One key component is the use of Digital Signal Processors (DSPs), which are specialized microprocessors optimized for handling audio signal processing tasks. DSPs enable real-time application of complex algorithms for noise reduction, such as filtering or spectral subtraction, making them highly effective in processing audio with minimal delay. In addition to DSPs, optimal microphone placement plays a significant role in improving audio quality. By positioning microphones away from noise sources like fans or traffic and incorporating acoustic treatments in the environment, the capture of unwanted background noise can be minimized from the start. Further, accessories such as shock mounts and pop filters are used to reduce mechanical noise and plosive sounds, ensuring the audio remains clear and free of distortions caused by physical vibrations or harsh vocal articulations. These existing systems combine hardware-based noise mitigation with software-driven signal processing to achieve high-quality, noise-free audio recordings.

1.3.1 Drawbacks of Existing System

- Dependence on environmental factors.
- High cost and setup complexity.
- Limited effectiveness in complex noisy environments.

- Real-time processing limitations.

1.4 Proposed System

The proposed system focuses on reducing noise from audio signals using advanced digital signal processing techniques. It leverages the `noisereduce` library in Python, which implements a spectral gating algorithm for effective noise reduction. This algorithm identifies and suppresses unwanted noise in the frequency domain while preserving the desired audio signal.

To evaluate the system's performance, objective metrics such as Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and Peak Signal-to-Noise Ratio (PSNR) are used. These metrics provide measurable insights into the improvement in audio quality.

Additionally, the system includes a user-friendly interface for displaying the results, developed using HTML, JavaScript, and Flask. This web-based interface allows users to upload audio files, process them through the model, and visualize the output, making the system accessible and interactive for various applications such as speech recognition, audio enhancement, and audio restoration.

1.4.1 Advantages of Proposed System

- **Enhanced Audio Clarity:** Our system leverages the spectral gating algorithm implemented through the Python `noisereduce` library, allowing for effective suppression of background noise while preserving the integrity of the desired audio signal. This results in significantly improved audio quality, making it ideal for applications requiring high intelligibility.
- **Quantifiable Performance Metrics:** By evaluating the system using metrics such as Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and Peak Signal-to-Noise Ratio (PSNR), we ensure that the improvement in audio quality is both measurable and consistent, providing a reliable benchmark for its effectiveness in real-world scenarios.

1.5 Scope

The scope of this project involves designing and implementing a system for noise reduction in audio signals using digital signal processing techniques. The system aims to enhance the clarity and intelligibility of audio recordings by suppressing unwanted background noise.

Key aspects of the project include: 1. Developing a noise reduction model using the spectral gating algorithm implemented through Python's `noisereduce` library. 2. Evaluating the model's performance using objective metrics such as Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and Peak Signal-to-Noise Ratio (PSNR) to ensure measurable improvements in audio quality. 3. Creating a user-friendly web interface using HTML, JavaScript, and Flask to allow users to interact with the system, upload audio files, and visualize noise reduction results.

This system is intended to benefit applications such as speech recognition, audio enhancement, and audio restoration, where noise reduction plays a critical role in ensuring optimal performance. The project is designed to provide a scalable and accessible solution for individuals and industries dealing with audio quality challenges.

1.6 Conclusion

The "Noise Reduction from Audio" project successfully improves audio quality by suppressing unwanted noise using a spectral gating algorithm implemented with Python's `noisereduce` library. The system's effectiveness, evaluated through metrics like SNR, MSE, and PSNR, demonstrates significant enhancements in clarity. A user-friendly web interface built with Flask, HTML, and JavaScript allows seamless interaction and output visualization. This solution is practical for applications such as speech recognition, audio restoration, and audio enhancement, providing a reliable and efficient approach to addressing audio noise challenges.

Chapter 2

LITERATURE SURVEY

2.1 Audio de-noising by spectral subtraction technique implemented on reconfigurable hardware

The paper introduces a hardware-based implementation of the spectral subtraction algorithm for speech enhancement. The architecture adaptively estimates and subtracts environmental noise from speech signals, producing noise-free audio. It uses a Spartan6 LX45 FPGA with two principal blocks—noise estimation-subtraction and phase block—executing concurrently to leverage FPGA’s parallel processing capabilities. The design achieves better SNR compared to software implementations and provides resource utilization and delay metrics. This work highlights an efficient, high-speed, hardware-based approach to adaptive noise filtering, which can serve as a reference for integrating advanced noise reduction techniques into hardware systems.

2.2 The Improvement of Audio Noise Reduction System Based on LMS Algorithm

This paper, based on the original LMS algorithm, proposed a novel method with wavelet preprocessing and signal is used as reference input to improve the noise reduction effect of the adaptive noise cancellation system. The new algorithm, in the presence of

impulse noise, low SNR environment, has a higher amount of noise reduction than the original LMS algorithm based on adaptive noise cancellation system with better effect in reducing noise. The method first used wavelet packet threshold noise reduction algorithms, including impulse detection and de-noising, to remove the impulse part of the noise. Then the residual noisy signal is eliminated by adaptive noise cancellation system based on LMS which has signal as the reference input. Finally, the multi-group simulations are compared. Simulation results show that the improved adaptive noise cancellation system do have better noise reduction and practical value.

2.3 Improvement of audio noise reduction system based on RLS algorithm

This paper addresses the challenge of canceling or suppressing mixed noise, a combination of periodic and impulse noise, which significantly impacts audio quality. The proposed method first uses a spectrogram-based approach to detect and eliminate impulse noise. The remaining mixed noise is then processed using an improved Recursive Least Squares (RLS) adaptive filtering system to eliminate the periodic noise. The results show that the method can achieve an SNR of over 12dB and a noise reduction coefficient greater than 0.9, demonstrating its effectiveness in improving audio quality by addressing complex mixed noise environments

2.4 Stationary Noise Reduction

The Stationary Noise Reduction algorithm from PyPI reduces noise in audio signals using a frequency-based approach. The process begins by calculating a spectrogram of the noise clip (optional) and deriving statistical data from it. Based on these statistics, a threshold is determined to create a mask. This mask is applied by comparing the signal spectrogram to the threshold, and it is smoothed over both frequency and time for better accuracy. The mask is then applied to the signal's spectrogram, with an inversion if required. If no separate noise clip is provided, the algorithm treats the signal itself as the noise, which often provides effective results. This method is useful for reducing stationary noise in audio processing applications.

2.5 Conclusion

Based on the literature surveys explored for our project, various effective techniques for noise reduction have been proposed, each focusing on different aspects of audio signal processing. The integration of spectral subtraction algorithms and adaptive filtering methods, such as LMS and RLS algorithms, demonstrates significant improvements in noise reduction, especially in environments with mixed or impulse noise. Moreover, the use of wavelet preprocessing and spectrogram-based approaches further enhances noise removal, offering better SNR and noise reduction coefficients. The Stationary Noise Reduction algorithm from PyPI, utilizing frequency-based processing and adaptive masks, is another promising approach for stationary noise. By combining these techniques, our project aims to leverage their strengths and provide an efficient and scalable solution for noise reduction, with applications in speech enhancement, audio restoration, and other domains requiring high-quality audio processing.

Chapter 3

SYSTEM ANALYSIS

The Noise Reduction System is designed to handle noisy audio files uploaded by users, process them to reduce noise, and return the processed audio files. This system provides a user-friendly web interface and uses advanced noise reduction algorithms to ensure high-quality output. It has the following objectives:

- Facilitate easy audio file uploads.
- Process audio files efficiently and return results promptly.
- Provide reliable noise reduction using established signal processing techniques.
- Ensure scalability and usability for diverse user requirements..

3.1 Algorithms

3.1.1 Spectral Gating Algorithm

This algorithm balances computational efficiency with effective noise reduction, making it suitable for real-time or batch processing scenarios.

- Noise Profile Estimation: - Analyze a portion of the audio signal where only noise

is present. - Compute the spectral power of the noise.

- Short-Time Fourier Transform (STFT): - Apply STFT to the entire audio signal to obtain its time-frequency representation.
- Noise Suppression: - Compare the spectral power of the signal with the noise profile. - Attenuate frequencies where the signal power is close to or below the noise power.
- Inverse STFT: - Reconstruct the audio signal from the modified time-frequency representation.
- Post-Processing: - Apply smoothing to reduce artifacts introduced during the noise suppression step.

3.2 System Architecture

Fig 3.2 shows Architecture of noise reduction from audio.

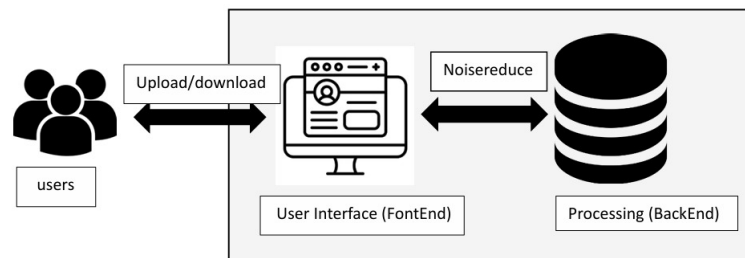


Figure 3.1: System Architecture

3.3 Feasibility Study

The system is designed with simplicity and user experience in mind:

- **Ease of Use:** Users can upload files and retrieve results without needing technical knowledge.
- **Maintenance:** The modular design ensures that individual components (e.g., noise reduction logic) can be updated independently.
- **Scalability:** The system can handle multiple simultaneous uploads with minimal latency.

3.3.1 Technical Feasibility

To determine whether the proposed system is technically feasible, we should take into consideration the technical issues involved behind the system.

- Technical Feasibility is the process of figuring out how you're going to produce your product or service to determine whether it's possible for your company.
- Before launching your offerings, you must plan every part of your operations, from first sourcing your production materials all the way to tracking your sales.
- By looking at all the logistics of this process, you can determine potential challenges and figure out ways to overcome them.
- Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system.

3.3.2 Operational Feasibility

To determine the operational feasibility of the system we should take into consideration the awareness level of the users. This system is operational feasible since the users are familiar with the technologies and hence there is no need to gear up the personnel to use system. Also the system is very friendly and to use.

3.3.3 Economic Feasibility

To decide whether a project is economically feasible, we have to consider various factors as:

- Cost-benefit analysis
- Long-term returns candidates appearing
- Maintenance costs

It requires average computing capabilities and access to internet, which are very basic requirements and can be afforded by any organization hence it doesn't incur additional economic overheads, which renders the system economically feasible. The examination system being an online system should be available anytime.

3.4 Hardware Requirements

Hardware refers to the physical components of a computer. Computer Hardware is any part of the computer that we can touch these parts. These are the primary electronic devices used to build up the computer. Examples of hardware in a computer are the Processor, Memory Devices, Monitor, Printer, Keyboard, Mouse, and the Central Processing Unit.

System: Intel i3 processor and above.

Input devices: Mouse, Keyboard.

RAM: 4GB and above.

Hard disk: 512GB.

3.5 Software Requirements

Software is a collection of instructions, procedures, and documentation that performs different tasks on a computer system. we can say also Computer Software is a programing code executed on a computer processor. The code can be machine-level code or

the code written for an operating system.

Operating System: Windows 10 and above/Linux

Programming Languages: Python

Front-End: HTML, flask.

Back-End: Librosa, Soundfile, Noisereduce, Matplotlib

Editors: VS code

IDE: Googlecolab

3.6 Conclusion

In this chapter, we discussed the Functional and Non-Functional Requirements, Feasibility Study, System Architecture of Admin and User modules, Hardware and Software requirements that are required to implement the system.

Chapter 4

SYSTEM DESIGN

4.1 UML

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system. A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

UML Concepts: The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

Characteristics of UML:

The UML has the following features:

- It is a generalized modeling language.
- It is distinct from other programming languages like C++, Python, etc.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts.

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Building Blocks of the UML: The vocabulary of the UML encompasses three kinds of building blocks.

- **Things:** Things are the abstractions that are first-class citizens in a model
- **Relationships:** relationships tie these things together
- **Diagrams:** diagrams group interesting collections of things

4.2 Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors. In this project, faculty and student are the actors.

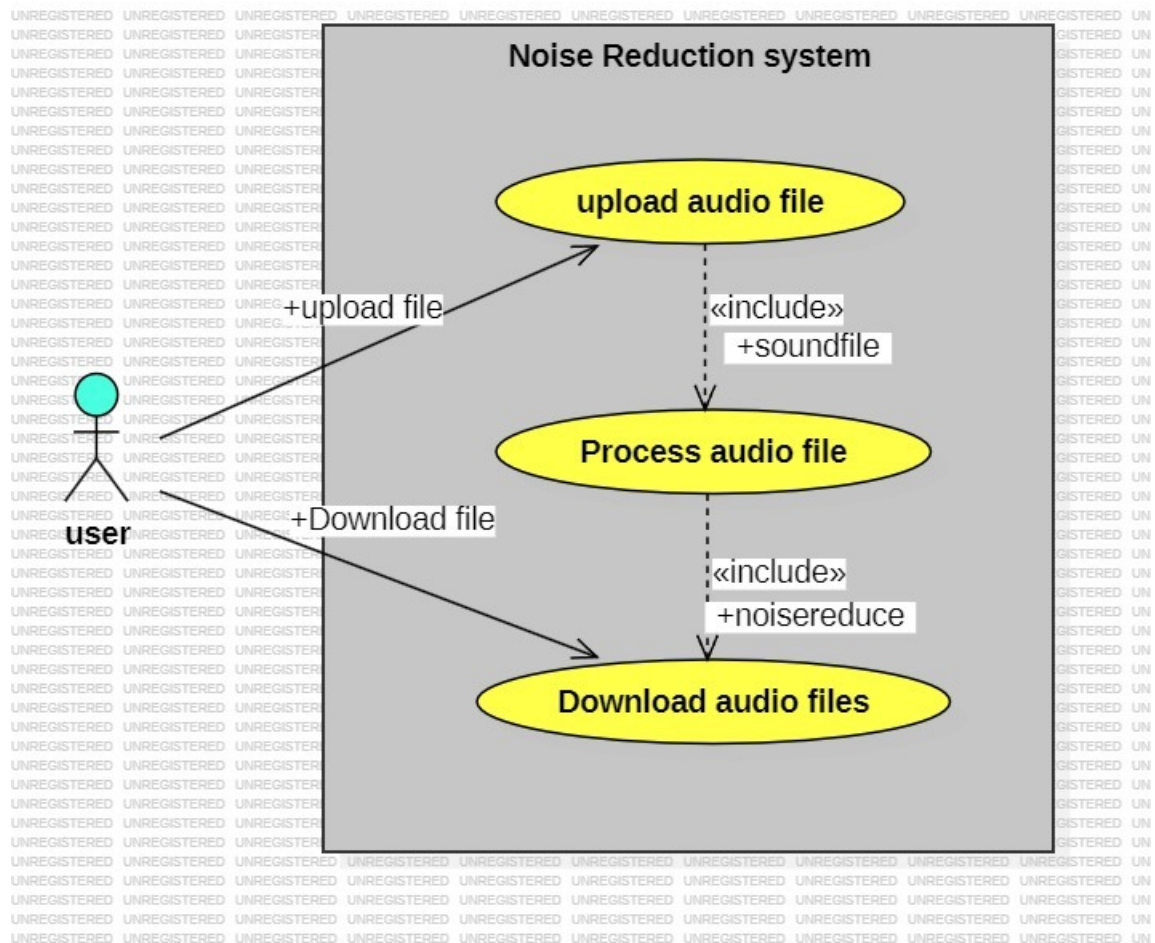


Figure 4.1: Use Case Diagram of System

4.3 Activity Diagram

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

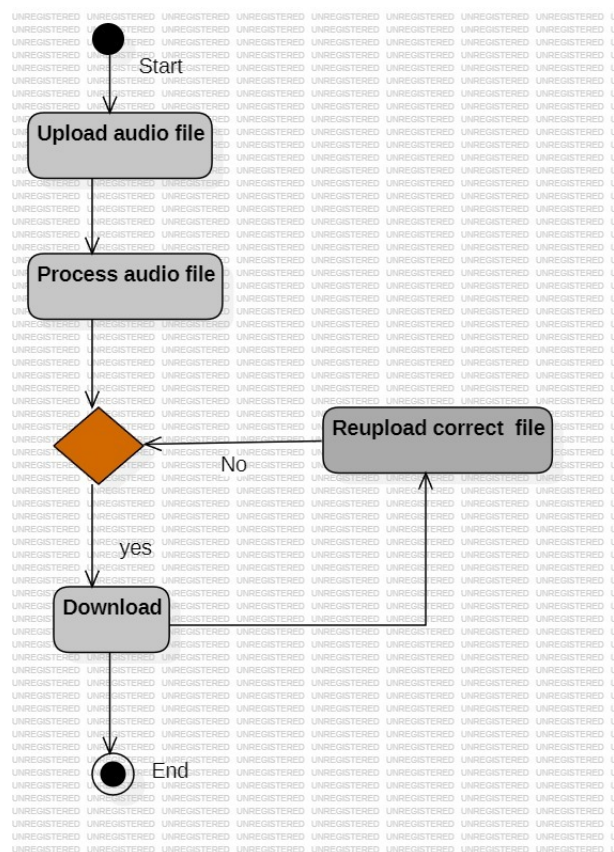


Figure 4.2: Activity Diagram of System

4.4 Sequence Diagram

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

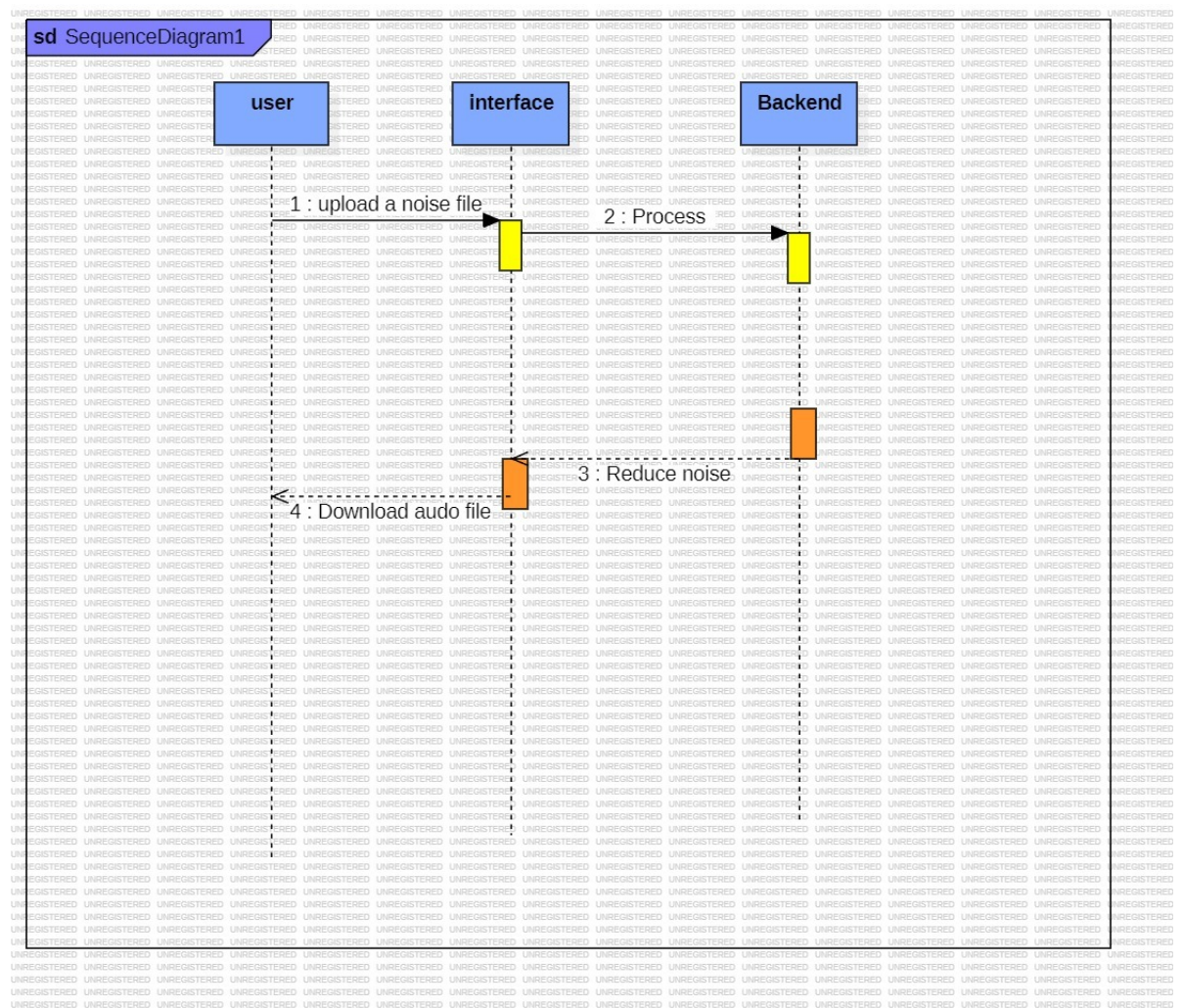


Figure 4.3: Sequence Diagram of System

4.5 Conclusion

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Chapter 5

IMPLEMENTATION

5.1 Frontend Implementation

This section handles the user interface, where users can upload their audio files, view processing status, and download the processed file.

5.1.1 Create HTML Form for File Upload:

```
<form id="upload-form" action="/process" method="POST" enctype="multipart/form-data">
  <label for="audio-upload">Upload your audio file:</label>
  <input type="file" id="audio-upload" name="audio-file" accept="audio/*" required>
  <button type="submit">Upload and Process</button>
</form>
```

5.1.2 Display Loading Spinner During File Processing:

```
<div id="loading">
  <div class="spinner"></div>
```

```
<p>Processing your file, please wait...</p>
</div>
```

5.1.3 Show Download Link After Processing:

```
<div id="download-section" style="display: none;">
  <h2>Processed File Ready</h2>
  <a id="download-link" href="#" download>Download Processed File</a>
</div>
```

5.1.4 JavaScript to Handle File Upload and Display Results:

```
form.addEventListener('submit', async (event) => {
  event.preventDefault();
  loadingIndicator.style.display = 'block';
  downloadSection.style.display = 'none';

  const formData = new FormData(form);

  try {
    const response = await fetch('/process', {
      method: 'POST',
      body: formData
    });

    if (response.ok) {
      const blob = await response.blob();
      const url = URL.createObjectURL(blob);
      downloadLink.href = url;
      downloadSection.style.display = 'block';
    } else {
      alert('Error processing file. Please try again.');
```

```

    } catch (error) {
        alert('An error occurred. Please try again later.');
```

```

    } finally {
        loadingIndicator.style.display = 'none';
    }
}
```

5.2 Backend Implementation

This section describes the server-side code that handles the audio file upload, processing, and serving the processed file back to the user.

5.2.1 Initialize Flask Application:

```

app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
PROCESSED_FOLDER = 'processed'

# Create necessary folders if they don't exist
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(PROCESSED_FOLDER, exist_ok=True)
```

5.2.2 Route for File Upload and Processing:

```

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/process', methods=['POST'])
def process_file():
```



```

if 'audio-file' not in request.files:
    return "No file uploaded!", 400

file = request.files['audio-file']
if file.filename == '':
    return "No selected file!", 400

# Save the uploaded file
file_id = str(uuid.uuid4()) # Unique ID for file
input_path = os.path.join(UPLOAD_FOLDER, f"{file_id}_{file.filename}")
file.save(input_path)

# Process the file
output_path = os.path.join(PROCESSED_FOLDER, f"{file_id}reduced{file.filename}")
process_audio(input_path, output_path)

return send_file(output_path, as_attachment=True)

```

5.2.3 Run the Flask App:

```

if __name__ == '__main__':
    app.run(debug=True)

```

5.3 Model Processing (Noise Reduction Logic):

This section describes the core functionality of the project: audio noise reduction. It uses `noisereduce` and `librosa` libraries for processing the audio file.

5.3.1 Import Required Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

5.3.2 Audio Processing Function:

```
def process_audio(input_path, output_path):
    # Load the noisy audio file
    data, rate = librosa.load(input_path, sr=None, mono=False) # Load stereo i

    # Reduce noise
    reduced_noise = nr.reduce_noise(y=data, sr=rate)

    # Save the noise-reduced audio file
    sf.write(output_path, reduced_noise.T, rate) # Transpose if stereo
```

5.4 Data Handling

This section explains the management of input and output audio data.

5.4.1 Upload and Save Audio File:

```
file_id = str(uuid.uuid4()) # Unique ID for file
input_path = os.path.join(UPLOAD_FOLDER, f"{file_id}_{file.filename}")
file.save(input_path)
```

5.4.2 Save and Serve Processed Audio:

```
output_path = os.path.join(PROCESSED_FOLDER, f"{file_id}reduced{file.filename}")
process_audio(input_path, output_path)
```

5.5 Visualization in Model Development

Although the project primarily focuses on audio processing, you can add visualization to show the effects of noise reduction on the audio signal.

5.5.1 Plot Spectrograms of Original and Processed Audio:

```
import matplotlib.pyplot as plt
import librosa.display

# Plot original and reduced noise spectrograms
def plot_spectrogram(audio_data, rate, title):
    plt.figure(figsize=(10, 4))
    D = librosa.amplitude_to_db(librosa.stft(audio_data), ref=np.max)
    librosa.display.specshow(D, sr=rate, x_axis='time', y_axis='log')
    plt.colorbar(format='%+2.0f dB')
    plt.title(title)
    plt.show()

# Example usage
plot_spectrogram(data, rate, 'Original Spectrogram')
plot_spectrogram(reduced_noise, rate, 'Processed Spectrogram')
```

5.6 Conclusion

In this project, we implemented a simple web application using Flask to upload an audio file, process it for noise reduction, and allow users to download the cleaned audio file. The frontend is responsible for handling user interaction, while the backend processes the uploaded file using noise reduction techniques. Data handling ensures that audio files are correctly managed during the process, and visualization provides insight into the effects of noise reduction.

Chapter 6

TESTING

6.1 White-Box Testing

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independent paths in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

6.2 Black-Box Testing

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

The following are the different tests at various levels:

6.3 Unit Testing

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the[5] module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

6.4 Integration Testing

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

6.5 Validation Testing

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

6.6 System Testing

This testing is a series of different tests whose primary is to fully exercise the computerbased system. This involves:

- Implementing the system in a simulated production environment and testing it.
- Introducing errors and testing for error handling.

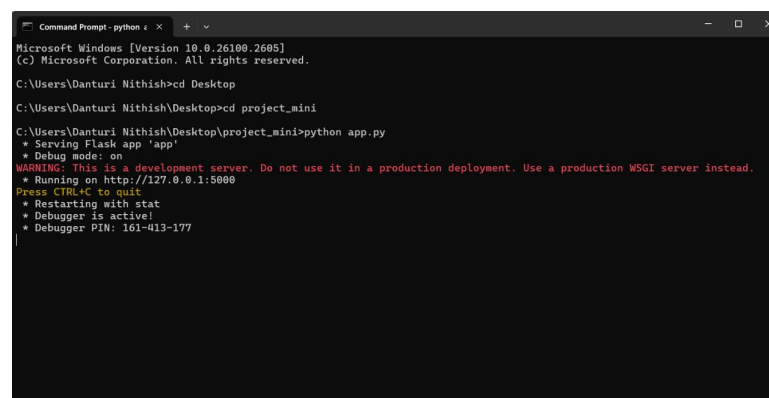
6.7 Conclusion

Software testing is an important part of the software development process. It is not a single activity that takes place after code implementation, but is part of each stage of the lifecycle. A Successful test strategy will begin with consideration during requirements specification.

Chapter 7

RESULTS

7.1 Start the flask app



```
Command Prompt - python 4 x + v
Microsoft Windows [Version 10.0.26100.2685]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Danturi Nithish>cd Desktop

C:\Users\Danturi Nithish\Desktop>cd project_mini

C:\Users\Danturi Nithish\Desktop\project_mini>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 161-413-177
```

Figure 7.1: Start the flask app

7.2 Web Interface

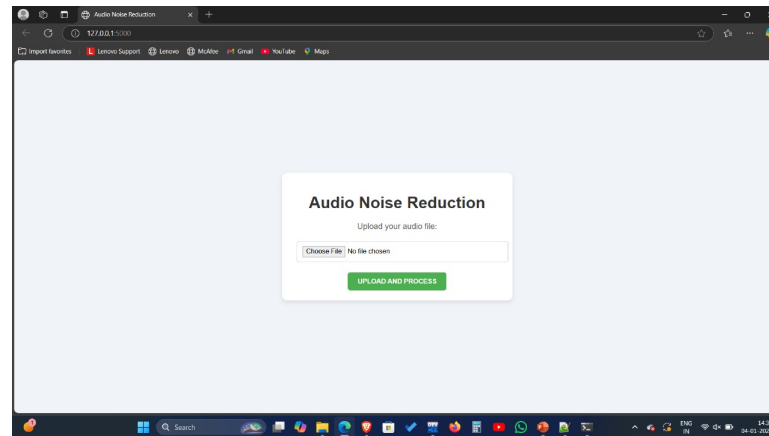


Figure 7.2: Web interface

7.3 Upload Audio File

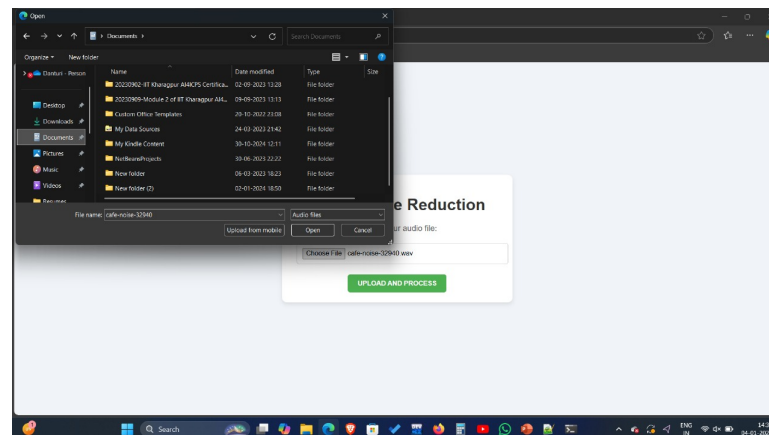


Figure 7.3: Upload Audio File

7.4 Processing the File

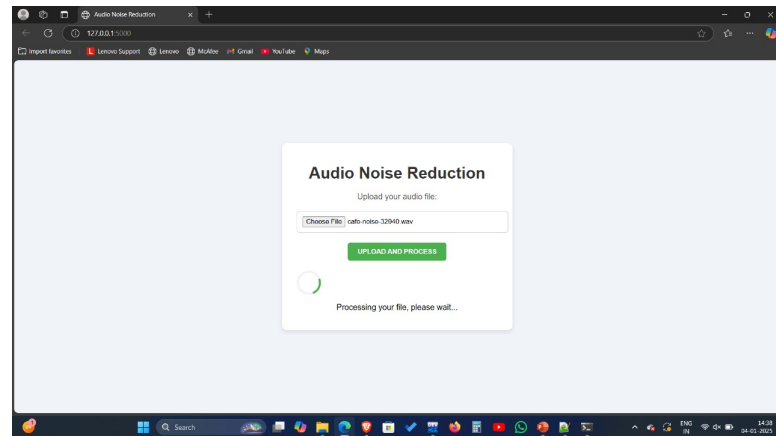


Figure 7.4: Processing the File

7.5 Download processed file

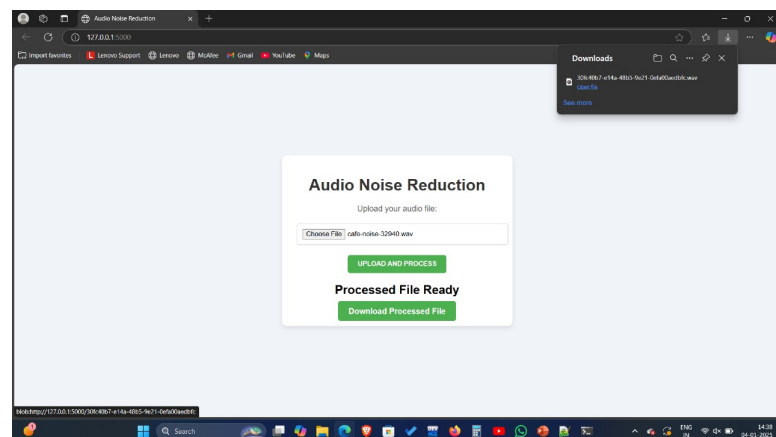


Figure 7.5: Download processed file

7.6 Audio noise reduction visualization

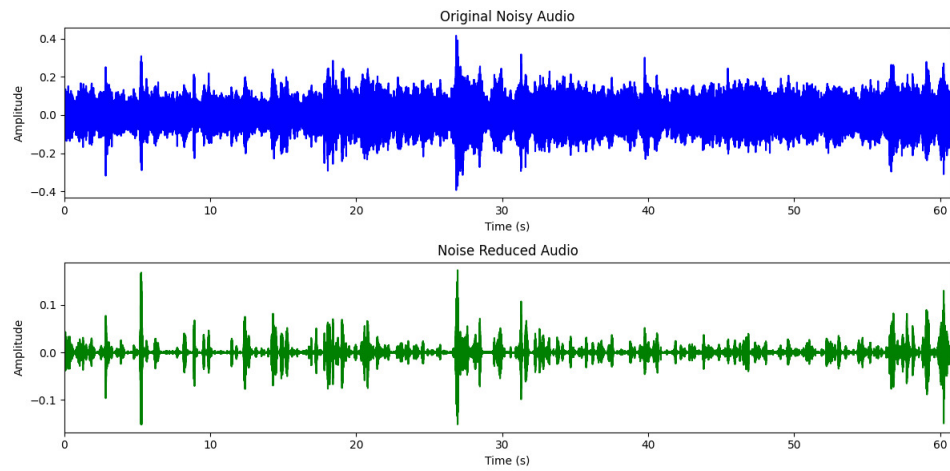


Figure 7.6: Audio noise reduction visualization

7.7 Conclusion

By using existing system difficult to analyze the insights of the data in order to get rid of this problem we proposed a new system which is user friendly interface and very useful in order to noise reduction.

CONCLUSION

The Noise Reduction from Audio project effectively combines digital signal processing techniques with user-friendly web interfaces to enhance the clarity of audio files by suppressing unwanted noise. Using the Spectral Gating Algorithm and Python libraries like librosa and noisereduce, the system ensures significant noise reduction while maintaining the integrity of the original audio. A robust backend powered by Flask handles audio processing, while the intuitive frontend enables seamless file uploads, processing, and downloads. This project demonstrates practical value in applications such as speech recognition, media restoration, and audio enhancement, providing a scalable and efficient solution for real-world audio noise reduction challenges.

REFERENCES

- [1] *Noise Reduction in Audio File Using Spectral Gating and FFT by Python Modules*
- [2] *Noise Reduction using Spectral Gating in Python (Tim Sainburg)*
- [3] *Noise Reduction using Spectral Gating in Python (Parathps7)*
- [4] *Noise Cancellation Using Spectral Gating Filter*
- [5] *Noise Reduction — SpectraLayers Elements Tutorials*