# Identification and Reduction of Noise by Mechanical Systems Onboard Ships


SHAHBAZ FAREED
**01-134221-072**
AHMED SHAHZAD
**01-134221-010**

Supervisor: Dr. Muhammad Asfand e Yar Khan

Co-Supervisor: Dr. Muhammad Khurram Ehsan

**Bachelor of Science in Computer Science**

Department of Computer Science
Bahria University, Islamabad

December 2025

# Certificate

We accept the work contained in the report titled "Identification and Reduction of Noise in Mechanical Systems Onboard Ships", written by Mr. Ahmed Shahzad AND Mr. Shahbaz Fareed as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Dr. Muhammad Asfand e Yar Khan

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr Asim Ali

_____

Head of the Department: Dr. Muhammad Khurram Ehsan

_____

# Abstract

Noise generated by mechanical systems on a warship can pose serious concerns for the ship and crew members. Our project "Identification and Reduction of Noise by Mechanical Systems" focuses on analysis and mitigation of any problem that can appear in mechanical systems. Excessive noise in warships can be a serious issue, it can mean that warship stealth is compromised, which can increase detectability through sonar by enemies ship, which usually results in high costs. In this study, we have employed advanced Machine learning techniques and signal processing techniques to process noise in warships. Our system helps naval engineers improve warship noise control through data driven, AI based solutions.

Our solution is implemented as a web application that allows crew members and engineers to detect unusual noises in real time using the ship's internal noise. Unlike traditional systems that require complex hardware and fixed installations, this approach is lightweight, portable, and accessible on everyday devices. We used CRNN, which is a Convolutional Recurrent Neural Networks to identify and classify noises along with lightweight Convolutional Encoder Decoder network for noise reduction. By identifying these noises, we can improve situational awareness and also maintenance.

In addition to detection, the system keeps the logs of all the identified noise so that the user can study all the noise patterns across time and detect common issues. The features of noise detection, classification and logs storage are important in providing improved decision making to the naval engineers and also minimizing the operational risks and maintenance expenses. With focus on portability as much as it can work on web in addition to Artificial intelligence, the system shows how AI and digital signal processing can increase the reliability and efficiency of naval operations. This work can be extended in future to incorporate shipboard equipment, increase the scope of the sound classification database and be implemented in other applications like industrial equipment and submarines.

# Acknowledgments

Firstly, we express our gratitude to Allah Almighty, the Most Merciful and Most Beneficent, for His blessings. We would like to express our sincere Appreciation to our Supervisor Sir Muhammad Asfandyar Khan for his invaluable guidance, support, and patience throughout the development of this project. We are extremely grateful for his untiring support and encouragement, which plays a vital role in the successful completion of this project. Thank you to both of our parents, who have been the cornerstones of support throughout this journey. They always believed in us and supported us. Without their tremendous understanding and encouragement in the last 4 years, it would be impossible for us to complete this degree and final year project.

SHAHBAZ FAREED & AHMED SHAHZAD
Bahria University
E-8, Islamabad, Pakistan

December 2025

# Contents

# List of Figures

# List of Tables

# Acronyms and Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CNN | Convolutional Neural Network |
| CRNN | Convolutional Recurrent Neural Network |
| DFD | Data Flow Diagram |
| DSP | Digital Signal Processing |
| FR | Functional Requirements |
| FFT | Fast Fourier Transform |
| ML | Machine Learning |
| NFR | Non-Functional Requirements |
| SDK | Software Development Kit |
| SQL | Structured Query Language |
| URN | Underwater radiated noise |

# Chapter 1

# Introduction

In modern systems, noise identification and reduction plays an important role in safety and operational efficiency [12]. Our project focuses on detecting noise in different parts of the ship. Our project uses Machine learning techniques to detect noise. Our system uses CRNN [8] to analyze noise and identify noise sources and produce a graph displaying the peaks along with classification of that noise and reducing that noise. This system aims to improve situational awareness, improve maintenance, and reduce mechanical failures before they escalate into critical issues. Quite a lot of research has already been done on underwater noise produced by vessels/ships that also disturbs marine life [5]. There are many technologies that can be applied to naval ships, meanwhile some technologies are in their early stages of development that could provide promising results. The only problem is that there is a lack of quantitative data on the effectiveness of noise reductions. The lack of economic incentives and data makes it more difficult for naval forces to adopt this technology [10].

## 1.1 Project Overview

This project is a web based application that takes noise as input and classifies that noise in a given category. It is an AI driven solution. CRNN is trained to achieve this goal. The reason why CRNN is used instead of other models is because it has both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) which makes it really useful for this type of analysis. This model is a hybrid [13] approach that helped us analyze the audio and classify it. Unlike diagnostics that rely on manual inspection and subjective human analysis, this AI driven solution detects anomalies and offers real time alerts for naval engineers.

The system uses HTML for the front end of the web application and records audio through the built in microphone and sends it to the Flask which is used as backend. The audio is normalized using the Mel Spectrogram, and, the Mel Spectrogram is passed to the trained models to detect noise. The model predicts the type of noise and returns the classification results; another model is used to reduce the noise using ML. The app then displays the classification

results along with the original audio and the reduced audio. The result is stored locally on the user's device using SQLite.

## 1.2 Problem Description

Naval engineers face a lot of hurdles while designing a ship, but the greatest challenge is the reduction of noise. The goal of naval engineers is to minimize it as much as possible because their goal is to make the ship as stealthy as possible. The most advanced National Navy around the world have shown the importance of noise control in ships, not only in terms of stealth but also in terms of health. Many officials were reported to experience hearing loss due to high vibrations on a ship. More noise also compromises the secrecy of surveillance missions [9]. The noise might be generated from loose objects on the ships, such as the door, bed rack, engine bay, or something else. The ability to distinguish between normal and abnormal noises is crucial because it can mean serious threats to a ship. However, there are several challenges to achieve this goal, including reducing background noise, differentiating between normal and abnormal noise, and ensuring real time accuracy. This solution is robust enough to function in diverse acoustic environments while minimizing false alarms.

## 1.3 Project Objectives

The following are the objectives of this project:

- Develop an AI based system to detect, classify, and reduce unwanted noise.
- Train and optimize machine learning model CRNN for real time noise classification and noise reduction.
- Develop a web application interface for real time results.
- To store classification the logs in user's device for analysis.

## 1.4 Project Scope

We developed an AI based system to detect, classify, and reduce unwanted noise. Trained and optimized ML model (CRNN) for real time noise classification [14] and reduction. The main design is an anomaly detection system to differentiate normal and abnormal noise. Developed a web application interface for real time results.

However, there are some things that cannot be done right now. Like, this project does not detect noise from other ships. Our system does not transcribe human speech. It detects abnormal noise on the ship, but does not provide root cause analysis. The initial version of this software cannot be used with the hardware on the ship, so it uses a standard microphone on the user's device to record the noise for detection.

## 1.5 Tools/Technology

**Tools:**

- **Jupyter:** For data analysis, data pre-processing, and model training.
- **SQLite:** To store labeled noise data and classification results.
- **VS Code:** For coding and integrating the front end with the back end.

**Technologies:**

- **Python:** A high level language used for Machine learning model training and backend logic.
- **Flask:** It acts as back end to handle recording, pre-processing, and fetching results from ML model.
- **SQLite:** For storing noise classification results, such as logs and user information.
- **HTML/JavaScript:** Used to build the front end web interface for recording and displaying results.

## 1.6 Summary

This chapter introduces the problem of mechanical noise in ships and its impact on safety, stealth, and maintenance. It outlines the purpose of the project to design a web based application that detects and classifies abnormal noises in real time. The objectives include building an AI detection model, reducing background noise, and providing alerts through notifications. The scope clarifies that the system focuses on internal ship noise and excludes speech recognition or external noise. Tools and technologies such as Python, Flask, SQLite, and CRNN are also highlighted along with their purpose.

# Chapter 2

# Literature Review

This chapter provides us with an overview of past and current advancements in Noise Identification and Reduction in Mechanical Systems. Many articles and research papers have been written on this technique. The purpose of this chapter is to explore past work and research done on this topic.

## 2.1 Introduction to Noise in Ships

Shipping traffic is one of the main sources of noise in the sea, underwater noise has increased due to increased human activity. Hence, ships should be understood and controlled to reduce the impact of noise on marine fauna. The noise from ships includes flow noise, propeller noise, and machinery noise. Normally, hydrophones are used to collect data from different locations on the ship. The quality of the data is measured by a coherence function [7]. The lowest acoustic and vibration levels were observed in the low and high frequencies.

In order to detect noise in the maritime environment, hydrophones are installed under water to capture noises made by the sea vessels; and they are also used to detect enemy ships and as an early alert system. The noise of ships is high in decibels due to large amounts of noise produced by engines, propellers, and other ship equipment.

## 2.2 Types of Noise in Ships

Noise onboard ships may include propeller noise, engine noise, or flow noise [7]. There are always different types of noise depending on different components of the ship, there may be human voices, ventilation system noise, etc [1]. There are also outside noises such as ocean, birds, other ship etc. But in warships, these acoustic noises can compromise the stealth of the ship, which can ultimately lead to their detection and compromise the mission [16]. Analytical methods such as the coherence function and the estimation of the Transfer Function can be used to reduce noise in ships [7].

## 2.3 Techniques for Noise Detection

Identification Methods of Underwater Noise Sources Generated by Small Ships [4] explains that the Fourier transform, the short Fourier transform, and the noise intensity method can be used for precise analysis. To calculate the signal spectrum, SFTF can be used because it has a time window in which it scans the whole signal. The noise intensity probe is used to capture the noise pressure. Another research shows that the Signal-to-Noise Ratio (SNR) MEMS noise listener can be used to detect noise on the big ships [15]. However, MEMS noise sensor needs to be installed inside the ship, it offers compact size and low power consumption. The traditional methods require hardware installations which is time consuming and resource taking therefore, a modern approach is needed which can detect noises without requiring so many resources.

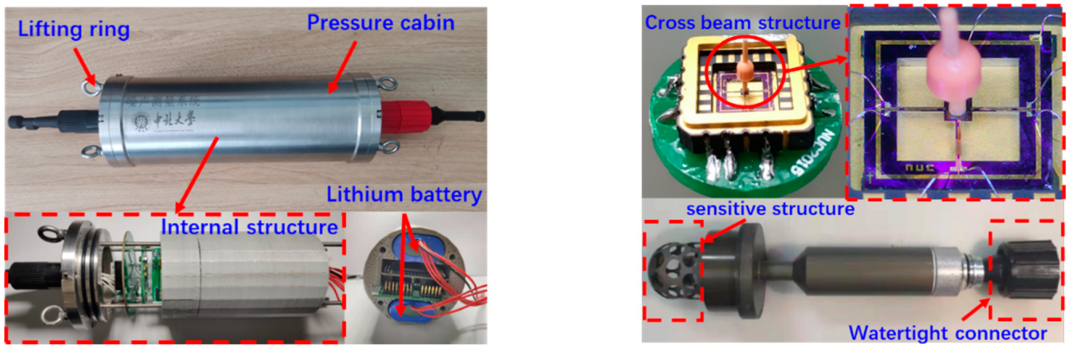


Figure 2.1: MEMS", use: "MEMS sensor used for real-time noise detection in large vessels.

## 2.4 Classical signal processing techniques

During the transitions from hardware to more AI based approaches. Most of the systems used spectral subtraction, Wiener Filtering, and other filtering techniques to reduce background noise. These approaches were cheap and lightweight for operations in low level systems. They were used either independently or combined to reduce stationary noise [3].

## 2.5 Machine Learning in Digital Signal Processing

Automatic modulation recognition (AMR) is a crucial technology that identifies the modulation scheme of received signals without prior information, enabling further signal processing when details are unknown. Recent advances in deep learning (DL) such as CNNs, RNNs, CRNNs and many more have paved the way for high performance DL based AMR methods, outperforming traditional techniques in accuracy and reliability thanks to neural networks' powerful feature extraction and classification capabilities. However, despite their potential, DL AMR methods also introduce challenges such as computational complexity and interoperability, which can impact real world deployment in wireless communication systems.

Through the use of Deep Learning algorithms, the use of hardware installations can be mitigated and results will be accurate and real, but it comes with its own set of challenges such as computation power, dataset's availability, and deployment in embedded systems of the ship itself.

## 2.6 Deep Learning and Noise Reduction

The research in this field has recently transitioned from traditional filtering methods towards deep learning based systems for noise reduction primarily for their ability to make noise spectrograms back into clean representations these systems work by having an encoded compressed input data into a latent representation which decoders then used to reconstruct a device signal.

Hence, many models utilized U-Net style architecture with skip connections to preserve details, resulting in computational and memory costs that are often too high for practical shipboard deployment.

Therefore, current research is shifting towards lightweight, compact networks that prioritize efficiency. By using bottleneck representation and favoring bi-linear up-sampling over transposed convolution, these systems avoid artifacts and reduce overhead.

When combined with mel- or log-scaled frequency processing, these streamlined architectures successfully suppressed noise while retaining the essential spectral features required for ship identification

## 2.7 Mechanical Noise Reduction Systems

Noise pollution from mechanical systems, for example, engines and industrial machinery, poses significant challenges to human health and productivity. Engineers employ two primary mitigation strategies: passive noise control, which uses vibration isolation, noise absorbing materials, and enclosures to block noise at its source; and active noise control (ANC) [6], a sophisticated real time approach that uses microphones, adaptive algorithms, and anti noise generation to cancel unwanted noise. Although passive methods are effective for general noise reduction, ANC proves particularly valuable in confined spaces where traditional methods are ineffective. However, challenges remain in system stability, sensor placement, and algorithm optimization. Recent advances in adaptive control have significantly improved the effectiveness of ANC, making it an increasingly viable solution for modern noise management [**?** ].

## 2.8 Summary of Gaps

The following are gaps in implementing this system:

- Limited Real World Validation: Most noise reduction methods, particularly Active Noise Control (ANC) have been popularly demonstrated in controlled conditions but have not been tested in extensive shipboard environments (such as varying speeds, sea states, or loads on the machinery).

- Integration Challenges: Although machine learning is a better noise detector, it cannot be easily incorporated into existing ship systems due to its computational complexity and lack of interoperability.
- Sensor Limitations: Existing noise detection schemes (such as MEMS listeners, hydrophones) are susceptible to trade off in dynamic marine environments, especially at low frequencies.
- Passive Active Hybrid Gaps:Little research has examined the best approaches to combine passive (materials, isolation) with Active Noise Control techniques on ships, and thus there is a lack of research on the potential synergies.
- Ecological Impact Focus: The study of underwater ship noise is more interested in stealth (such as warships) than in the overall ecological effects, and little is known about the effects on individual species.
- Dataset: The lack of real world and synthetic datasets makes this project and research extremely time consuming and resource-intensive which is why very few work has been conducted in this field.

## 2.9 Summary

This chapter reviews existing research on ship noise sources, detection techniques, and reduction methods. This section explains the types of noise and how they affect stealth and crew health [2]. Various signal processing techniques such as Fourier transforms and MEMS sensors are discussed. The role of machine learning in digital signal processing is highlighted, along with active and passive noise reduction strategies. The chapter identifies remaining research gaps, such as lack of real-world testing, sensor limitations, and integration challenges.

# Chapter 3

# Requirement Specifications

In this chapter, the requirements of the project "Identification and Reduction of Noise by Mechanical Systems Onboard Ships" are presented. It contains user story , what they want , the functional and non functional requirements, how this system works, how it behaves under various scenarios, how users engage with it. What are the challenges and limitations of existing systems and how our system addresses them.

## 3.1 Existing System

There are some existing systems that work on the same problem, which are as follows:

### 3.1.1 HBK Self Noise Monitoring System

HBK Self Noise Monitoring System is a powerful integrated system designed for ships and submarines to monitor their internal noises in real time. It is permanently installed as part of the vessel and uses an array of sensors to continuously record and analyze vibrations generated by mechanical components. This facilitates the location of sources of excessive noise, providing acoustic stealth to naval ships and mechanical performance improvement by allowing proactive maintenance.

### 3.1.2 Microflown Acoustic Imaging for Ships

Microflown Technologies offers a solution for acoustic imaging to detect where noise sources are located in ship engine rooms with its Scan and Paint 2D system. The system creates visual acoustic maps, enabling engineers to precisely see where unwanted mechanical noise is occurring. It is particularly valuable for maintenance, fault finding, and noise control and provides an efficient means of troubleshooting in complex environments of shipboard machinery.

## 3.2 Limitations of Existing Systems

### 3.2.1 High Cost and Hardware Dependency

Systems such as HBK SNMS and Microflown Acoustic Imaging require permanent installations and complex hardware, which makes them not only difficult but also expensive. They are ideal for large vessels and submarines, but not for smaller vessels.

### 3.2.2 Lack of Portability

The existing systems need to be installed permanently or semi-permanently, which limits their usage to only the areas where their sensors and equipment are installed. They cannot be moved easily, as the installation is very complex and costly.

### 3.2.3 Technical Complexity

These systems require thorough understanding and technical expertise of their operations, making them less accessible to non expert users such as typical crew members.

### 3.2.4 Integration and Accessibility

Traditional systems do not offer user friendly interfaces, logs of previous noises that can be accessed and viewed at any time. They also require technical expertise and are mostly limited to trained personnel in controlled environments.

## 3.3 User Scenario

Imagine you are on a warship and noise is coming from somewhere and you had to find it manually, it would be a waste of resources such as time, and energy which could be used on other tasks. Therefore, a solution should be built that can quickly detect and classify noise. Thus, the crew member can work on just fixing that noise instead of wasting time in finding the root cause. The app should ignore human noise, and noise outside the ship as we are focused on the internal noise only. The problem with manual approach is that it can be unreliable, as there can be any human error, unproductive, as resources are wasted in this process. Therefore, the app fixes those problems.

In this scenario, a crew member uses our system to make this task easier and more reliable. The user launches the app, authenticates using their registered credentials, or creates an account if a crew member is using the system for the first time. After authentication, the user is taken to the main dashboard where they can start noise detection, view recorded logs, manage their profile, and access other features.

During vessel operation or engine room inspection, the crew member wants to analyze whether a particular machine is producing abnormal mechanical noise. The crew member taps the “Detect Audio” button, which starts recording audio through the device’s microphone. The

captured noise is processed and sent to the Flask. The trained machine learning model evaluates the noise and classifies it into categories such as "Kaiyuan" , "Speedboat", "UUV".

Then, within a short period of time, the classification result is displayed on the application's interface. The detected noise label is clearly displayed to the user on the screen. If the system detects a potential fault, the application immediately alerts the user, allowing them to take corrective action or notify higher maintenance authorities.

Every classification result is automatically saved to the local SQLite database along with a timestamp. Even if there is no internet connectivity when the ship is out at sea, the logs remain saved locally and accessible to the user. When connectivity is restored, data can be synchronized with the central database.

At any time, the user can open the Logs section to scroll through the detection history. They can also tap on a particular entry to view stored details, such as detected noise type, date and time of detection, confidence score, and other information.

The account information can also be edited at any point, including email, password. The app will update the information and the changes can be seen instantly. The user can log out of the app securely after their shift session.

### 3.3.1 Functional Requirements

The functional requirements are the core functionality of the app that is specified by the User. They are derived from User story or scenario. The following are the functional requirements:

#### FR1: User Registration

**Input:** It takes username, email, and password as input.
**Process:** Allows the user to create a new user profile in the system, which they can use to log in.
**Output:** The new user is registered in the app and the user can log in to the system.
**Description:** The app must allow non-registered users to become members by registering through entering their details. The login credentials are stored securely and easy access to the app is provided after registering.

#### FR2: Secure Login and Logout Functionality

**Input:** The user account credentials for the login process and logout request for terminating the existing session.
**Process:** Validate the user credentials for the log in process and terminate the existing session if log out is requested.
**Output:** The user is redirected to the home screen after successfully logging in and redirected to log out screen after pressing log out button.
**Description:** The user is successfully able to log in and out of the app while having their data protected, ensuring security in the app.

#### FR3: Forgot Password

**Input:** Valid email address.
**Process:** Verify the email address and send a new password to the user.
**Output:** Email with a new password sent to the user.
**Description:** Users must be able to securely reset their password by entering a valid email address. The new password will be sent to them via email.

#### FR4: Profile Viewing and Editing

**Input:** Username, Email, or Password updates.
**Process:** Update profile details in the database.
**Output:** Confirmation of updated profile information.
**Description:** Users can view and edit their profile information such as email, username, and password. The profile information must be updated in the database simultaneously.

#### FR5: noise Detection

**Input:** User presses the noise detection button.
**Process:** Capture audio and classify noises while storing results in logs.
**Output:** Detected noise results stored and notification sent to the user.
**Description:** The app should feature a clearly visible button to initiate noise detection, triggering audio capture and classification, while storing results in logs and notifying the user.

#### FR6: Microphone Audio Capture

**Input:** Audio from the device microphone.
**Process:** Record audio input for analysis and noise classification.
**Output:** Stored audio for further processing.
**Description:** The system must access the device microphone to record audio input for analysis and noise classification.

#### FR7: noise Classification

**Input:** Spectrograms of recorded audio.
**Process:** Process spectrograms using a trained ML model in real time.
**Output:** Classification result of the detected noise.
**Description:** The system must process spectrograms using a trained ML model to classify noises in real time and notify the user about the detected noise.

#### FR8: Detection Logs

**Input:** Classified noise data and timestamp.
**Process:** Store noise type with detection time and details.

**Output:** Logged noise detection history.
**Description:** The system should maintain a log of all classified noises along with the exact time of detection and their details for further analysis.

#### FR9: noise Detection Notification

**Input:** Classified noise patterns or high-frequency detections.
**Process:** Generate a notification upon detecting specific patterns or faults.
**Output:** Notification alert sent to the user.
**Description:** The application must notify users about any noise patterns or high-frequency detections that could indicate faults.

### 3.3.2 Non-Functional Requirements

#### NFR1: Audio Processing

The system should be able to process audio and classify it within seconds to ensure real time classification and quick response to noise detection.

#### NFR2: Compatibility

The application must be compatible to run on almost every device.

#### NFR3: User Data Encryption

The account information or data of user such as login credentials, must be securely stored, as a result no other person can access it.

#### NFR4: English and Urdu Options

The app should support language options in both English and Urdu to ensure usability for a wide range of audiences and improve accessibility.

#### NFR5: Smooth Animations

All transitions within the app should run smoothly to enhance user experience and ensure fast loading times.

#### NFR6: Microphone Permission Handling

The application must clearly request the user's permission to access the microphone, providing users with transparent control over app permissions.

## 3.4 Use Cases

Use cases detailed in this section are used to describe the interactions of users and the system. They cover every possible scenarios. They are used to clarify system's working in detail while considering real life interactions.

### 3.4.1 Use Case Diagram

This diagram informs about the use cases of the system. It involves every possible action that takes place inside the web app from the login screen to the logout screen.

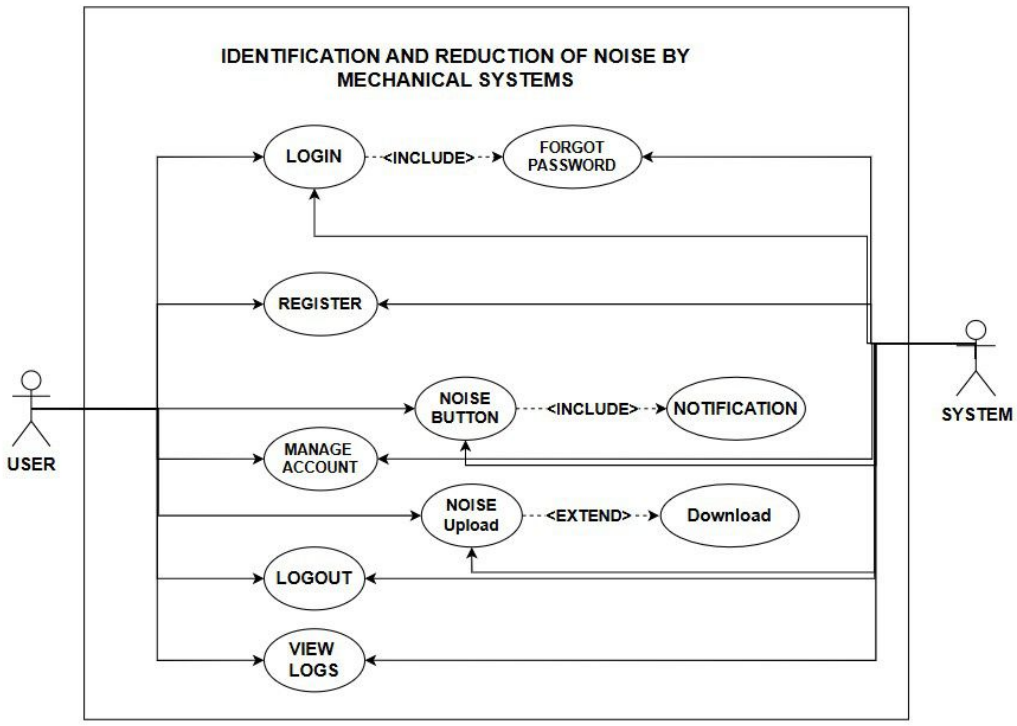


Figure 3.1: Use Case Diagram

### 3.4.2 Register Account

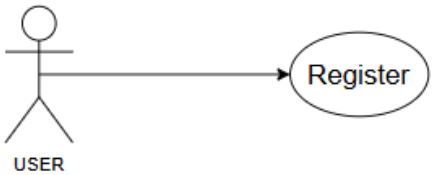


Figure 3.2: Register user account use case diagram

Figure 3.2 displays the process of how the non registered user can register account. The user presses register button and inserts their name, email and password to create the account.

Table 3-1 shows the use case description of the non registered user to register the account by setting up their account providing Username, Email and Password.

Table 3.1: Use Case Description for User Registration

| | |
|---|---|
| **Use Case ID** | UC-01 |
| **Use Case Name** | Register Account |
| **Actor(s)** | New User |
| **Description** | New Users can create their account by providing Username, Email and Password. |
| **Precondition** | User must be Non-Registered |
| **Postcondition** | User will be registered and redirected to the Login screen. |
| **Trigger** | Register Button |
| **Main Flow** | 1. User will click Register button.<br>2. System will display the Registration form to the User.<br>3. The user will provide username, email and password on the registration form.<br>4. System will register user and store data on database.<br>5. System will display login screen after storing data. |
| **Alternative Flow** | 1. User already exists.<br>2. Email already in use.<br>3. System will show error for not registering customer. |
| **Exception Flow** | Internet or system failure prevents user registration. |

### 3.4.3 Login

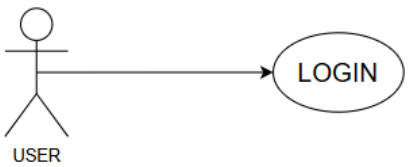


Figure 3.3: Login use case diagram

Figure 3.3 displays the process of how the registered user can enter into the web app. The user clicks on login button and inserts their username/email and password to log into their account.

Table 3-2 displays the use case description of user during the login process. They log into

their account by entering providing correct username and password in the login form. After successful login they will be redirected to home screen.

Table 3.2: Use Case Description for User Login

| | |
|---|---|
| **Use Case ID** | UC-02 |
| **Use Case Name** | Login |
| **Actor(s)** | Registered User |
| **Description** | Users can login into the app using correct credentials. |
| **Pre-condition** | User must be Registered |
| **Post-condition** | User will be granted access to the app and redirected to main screen. |
| **Trigger** | Login Button |
| **Main Flow** | 1. User will click Login button.<br>2. System will require their credentials.<br>3. The user will enter their credentials.<br>4. System will authenticate username and password.<br>5. The System will redirect User to home screen. |
| **Alternative Flow** | Incorrect username/email or password. The system will show an error message for incorrect credentials. |
| **Exception Flow** | Internet connectivity or authentication service failure prevents login. |

### 3.4.4 Forgot Password

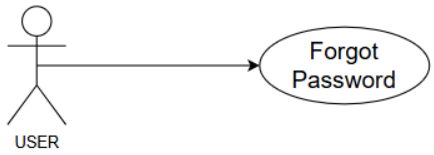


Figure 3.4: Forgot password use case diagram

Figure 3.4 displays the process of how the registered user can recover the password of their account if they forgot about it. The user clicks on forgot password button and inserts their email, the system then sends the reset link to their email where they can create a new password to login to the application.

Table 3-3 shows the use case description of the user who forgot their password. The system asks for their email and sends a new password to that email, which the user can use to log in to the application.

Table 3.3: Use Case Description for Forgot Password

| | |
|---|---|
| **Use Case ID** | UC-03 |
| **Use Case Name** | Forgot Password |
| **Actor(s)** | Registered User |
| **Description** | Registered Users can request for a new password if they forgot their previous password. |
| **Precondition** | User must be registered with a valid email address. |
| **Postcondition** | User will be given a new password via email. |
| **Trigger** | Forgot Password Button |
| **Main Flow** | 1. User will click Forgot Password button.<br>2. System prompts the user for email.<br>3. The user will provide the correct email.<br>4. System verifies the email and send reset link.<br>5. User will update the password.<br>6. System will overwrite the previous password with the new one. |
| **Alternative Flow** | 1. Incorrect email provided. The system shows an error message.<br>2. Email service not working. The system displays a "Try again later" message. |
| **Exception Flow** | Email service or internet connectivity failure prevents password reset. |

### 3.4.5 View noise Logs

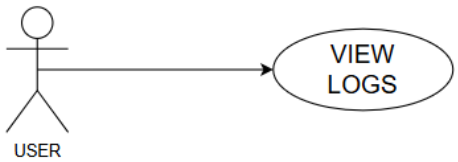


Figure 3.5: View noise logs use case diagram

Figure 3.5 displays the process of how the registered user can view the detected noise logs. The user clicks on logs button and can view all the detected noises along with their details.

Table 3-4 shows the use case description of user viewing noise entries in logs. The user, can view the noise logs along with all previously detected noises.

Table 3.4: Use Case Description for View Noise Logs

| | |
|---|---|
| **Use Case ID** | UC-04 |
| **Use Case Name** | View Noise Logs |
| **Actor(s)** | Registered User |
| **Description** | Users can view their historical noise detection logs with timestamps and noise type information. |
| **Precondition** | User must be logged in and have detected noises. |
| **Postcondition** | Noise log list is displayed. |
| **Trigger** | Selecting “Logs” from navigation drawer |
| **Main Flow** | 1. User opens the navigation drawer.<br>2. System displays menu options.<br>3. User selects “Logs”.<br>4. System retrieves and displays noise logs. |
| **Alternative Flow** | No logs exist: Empty log screen is shown. |
| **Exception Flow** | Network or database failure prevents retrieval of noise logs. |

### 3.4.6 Manage Account


Figure 3.6: Manage account use case diagram

Figure 3.6 displays the process of how the registered user can change their account details such as changing password. The user clicks on manage account button and can change their username, email and password.

Table 3-5 show the use case description of the user making changes to their account. The user, can use the manage account button to edit their account details such as email/password.

Table 3.5: Use Case Description for Manage Account

| | |
|---|---|
| **Use Case ID** | UC-05 |
| **Use Case Name** | Manage Account |
| **Actor(s)** | Registered User |
| **Description** | Users can view and update their account information and change password. |
| **Precondition** | User must be logged in. |
| **Postcondition** | Account information is updated in database. |
| **Trigger** | Selecting “Manage Account” from navigation drawer |
| **Main Flow** | 1. User opens the navigation drawer.<br>2. System displays menu options.<br>3. User selects “Manage Account”.<br>4. System displays current account information.<br>5. User edits information and saves changes.<br>6. System validates and updates information.<br>7. System confirms successful update. |
| **Alternative Flow** | 1. Invalid information – system displays an error message.<br>2. Wrong password – system prevents account update.<br>3. System error – system displays “Update failed. Try again later” message. |
| **Exception Flow** | Database or server failure prevents account update. . |

### 3.4.7 Detect noise

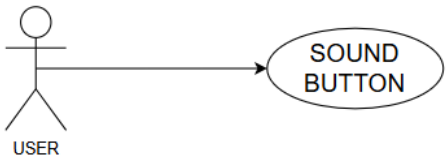


Figure 3.7: Detect noise use case diagram

Figure 3.7 displays the process of how the registered user can detect noise. The user clicks on noise button and waits a few seconds for the system to detect, analyze and classify noise. The results are displayed to the user along with reduced noise and reduced noise audio which they can listen to and compare with original audio.

Table 3-6 show the use case description of the user clicking the noise button to detect noise. After pressing the noise button, the system enters into a listening state and every detected noise is analyzed whilst notifying the user and storing results in the noise logs and showcasing results with graphs.

Table 3.6: Use Case Description for Detect Noise

| | |
|---|---|
| **Use Case ID** | UC-06 |
| **Use Case Name** | Detect Noise |
| **Actor(s)** | Registered User |
| **Description** | Users can initiate noise detection by pressing the noise detection button. |
| **Precondition** | User must be logged in and have microphone permission. |
| **Postcondition** | Noise is detected, analyzed, and the result is stored in logs. |
| **Trigger** | Detect Noise Button |
| **Main Flow** | 1. User presses the “Detect Noise” button.<br>2. System shows the “Listening” state.<br>3. System detects noise and analyzes it.<br>4. System sends a notification about noise to the user.<br>5. System displays and logs the result.<br>6. User views the result. |
| **Alternative Flow** | Microphone error: system shows “Microphone unavailable”. |
| **Exception Flow** | Permission denial or hardware failure prevents noise detection. |

### 3.4.8 Upload Noise

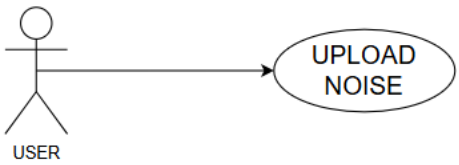


Figure 3.8: Upload noise use case diagram

Figure 3.8 displays the process of how the registered user can upload noise audio file into the app and get results. The user clicks on upload noise button and is shown the option to select the audio they want to upload and classify and clean.

Table 3-7 shows the use case description of the uploading noise when they click on upload noise button. The screen, displays the option to select the file they want to upload. After they have

selected the file and click on upload they will be presented with the classification results and reduced noise audio which they can listen to and compare with original audio.

Table 3.7: Use Case Description for Upload Noise

| | |
|---|---|
| **Use Case ID** | UC-07 |
| **Use Case Name** | Upload Noise |
| **Actor(s)** | Registered User |
| **Description** | Users can upload the noise file directly into the app. |
| **Precondition** | User must be logged in. |
| **Postcondition** | Noise is classified, and reduced noise graphs are generated. |
| **Trigger** | Upload Noise Button |
| **Main Flow** | 1. User clicks on the Upload Noise button.<br>2. System displays options to upload noise.<br>3. User uploads a `.wav` file.<br>4. System classifies the noise, cleans the audio, and generates signal and clean audio graphs. |
| **Alternative Flow** | Wrong or Corrupt data: system shows “Unknown”. |
| **Exception Flow** | Unsupported file format or processing failure prevents noise upload. |

### 3.4.9 Notification

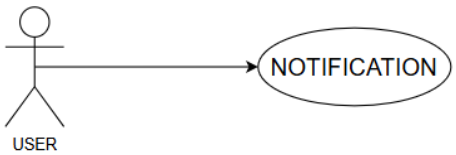


Figure 3.9: Notification Use Case Diagram

Figure 3.10 displays the process of how the registered user is displayed the notification after the noise has been detected.

Table 3-9 shows the use case description for the system notification feature. When the user clicks on the noise detection button, the application captures the audio through the microphone, sends it to the Flask API for processing, and then displays the classification result on the screen via a notification.

Table 3.8: Use Case Description for Notification

| | |
|---|---|
| **Use Case ID** | UC-09 |
| **Use Case Name** | Notification |
| **Actor(s)** | Registered User |
| **Description** | When a user initiates noise detection, the system analyzes the captured audio using the CRNN model. The user is then notified in real time about the detected noise type or abnormal noise through a pop-up notification on the app interface. |
| **Precondition** | User must be logged in and grant microphone permission. |
| **Postcondition** | User receives a real-time notification displaying the noise classification result. |
| **Trigger** | User taps the Noise Detection button |
| **Main Flow** | 1. User taps on the noise detection button.<br>2. System records audio through the microphone.<br>3. Flask API processes the audio and sends it to the CRNN model for classification.<br>4. System displays a notification with the detected noise type and stores it in logs. |
| **Alternative Flow** | Corrupted or incomplete data: system shows “Unknown noise”. |
| **Exception Flow** | Service interruption or model failure prevents notification delivery. |

### 3.4.10 Logout

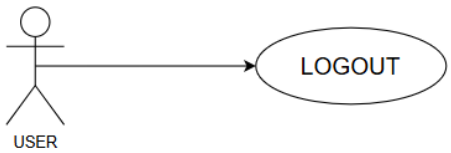


Figure 3.10: Logout use case diagram

Figure 3.11 displays the process of how the registered user can log out of their account. The user clicks on logout button and is redirected back to login screen.

Table 3-10 show the use case description of the user terminating their current app session by logging out. When the user clicks on the logout button. The system redirects the user to the home screen.

Table 3.9: Use Case Description for Logout

| | |
|---|---|
| **Use Case ID** | UC-10 |
| **Use Case Name** | Logout |
| **Actor(s)** | Registered User |
| **Description** | Allows users to terminate their current session and return to the login screen. |
| **Precondition** | User must be logged in. |
| **Postcondition** | The current session is closed and the user is redirected to the login screen. |
| **Trigger** | Selecting “Logout” from the navigation drawer |
| **Main Flow** | 1. User opens the navigation drawer.<br>2. System displays menu options.<br>3. User selects “Logout”.<br>4. System terminates the current session.<br>5. System displays the Login screen. |
| **Alternative Flow** | Session timeout — system automatically logs out the user. |
| **Exception Flow** | Session handling failure prevents proper logout. |

### 3.4.11 Download

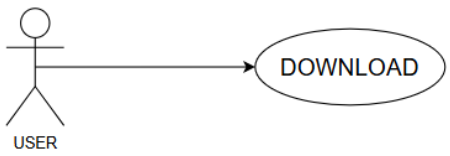


Figure 3.11: Download use case diagram

Figure 3.12 displays the process of downloading the clean audio after the classification. The download button is listed in the bottom of the classification result right besides the clean audio.

Table 3-11 show the use case description of the user downloading the clean audio from the web app. When the user clicks on the download button. The system downloads the clean audio on their local device.

Table 3.10: Use Case Description for Download

| | |
|---|---|
| **Use Case ID** | UC-11 |
| **Use Case Name** | Download |
| **Actor(s)** | Registered User |
| **Description** | Allows users to download the reduced noise audio. |
| **Precondition** | User must be logged in. |
| **Postcondition** | The clean audio is downloaded on the user's device. |
| **Trigger** | Selecting "Download" from the navigation drawer |
| **Main Flow** | 1. User clicks on the Download button.<br>2. System displays the save options.<br>3. User enters the file name and location.<br>4. System downloads the audio to the user's device. |
| **Alternative Flow** | No file uploaded — system prevents download. |
| **Exception Flow** | Storage permission or download failure prevents audio download. |

## 3.5 Summary

This chapter describes the shortcomings of existing solutions such as HBK SNMS and Microflown Acoustic Imaging, which are costly, complex, and non-portable. Our system aims to overcome these systems with a web based, AI powered application. Functional requirements such as noise detection, logs, notifications, user accounts and non functional requirements such as real-time response, security, usability are defined. Use cases illustrate how different user interactions are handled, from login to noise detection and log management.

# Chapter 4

# Design

## 4.1 System Architecture

The mechanical noise detection application is designed to be simple, lightweight, and easy to use by navy personnel without requiring any expertise. The app allows for real time noise classification using the device own microphone and a back end Flask that connects with machine learning model. The system architecture consists of core components that include the front end, back end processing, machine learning model, and data storage.

Following are the core components:

1) **Web Interface:** Easy to use User Interface. Hence everyone can use it without having a lot of knowledge or training.

2) **Back End Processing:** It Handles authentication, preprocessing, and communication between the web app, model, and database. It sends the recorded noise to the Machine learning model for classification and noise reduction.

3) **Machine Learning Model:** The trained model uses the preprocessed noise data to classify noises and return the prediction results via API response.

4) **Data Base:** Maintains logs of classification results, using a local SQLite database for record keeping and performance tracking.

The development process follows an iterative model, where the system is built in small increments and every iteration adds more functionality to the system according to feedback. User feedback is collected at every iteration and changes are incorporated accordingly. This ensures that the application gradually evolves to meet operational needs and improves reliability.

### 4.1.1 Components of the System

#### 4.1.1.1 User Interface

- **Login and Registration Screens:** Provide users with secure access to the app, including login, registration, and password recovery features.
- **Main Screen:** The main screen allows users to start noise detection, view classification results, and access the navigation menu with options for example, profile management, view logs, and logout.

- **Detection Interface:** Enables users to initiate noise recording, displays classification results returned from the model, and stores detected noises in logs along with relevant details.
- **Logs Screen:** Displays a list of all detected noises with details such as timestamps, classification, and confidence levels and reduced noise graph.
- **Account Screen:** Allows users to view and update their profile information in real time.

#### 4.1.1.2 Noise Processing

- **Audio Capture:** Uses the device's microphone to record noises when the "Detect noise" button is activated.
- **ML noise Classification:** The preprocessed audio is sent via Flask to ML model, which classifies the noise and returns the result.

#### 4.1.1.3 Backend

- **Authentication:** Manages user registration, login, session validation, and password reset using Flask.
- **Log Handling:** The backend records each classification event along with metadata such as ship type, timestamp, confidence level, and stores it in SQLite as Logs.

#### 4.1.1.4 Data Storage

- **User Database:** Stores user information and credentials securely in SQLite.
- **Logs Repository:** Maintains a history of classification results, including timestamps and confidence score,.

#### 4.1.1.5 External Interfaces

- **Permissions:** Requests user permission to access their device microphone for noise recording.
- **Compatibility:** Supports different browsers with optimized performance.

## 4.2 Design Constraints

### 4.2.1 Technical Constraints

- **Device Hardware Limitations**
- Limited RAM and CPU i.e resources on low end devices
- Battery drain from long audio recording sessions.

**Microphone Access**

- Requires runtime permissions for microphone from the device.

**Storage Management**

- Large log data may increase storage size

### 4.2.2 Programming Languages and Frameworks

- **Flask**
  - The Backend of this app is built using Flask for high speed asynchronous operations
  - Handles authentication, preprocessing, and model API calls
- **Jupyter Notebook**
  - It is Used for model training, and preprocessing noise for classification.

### 4.2.3 Integration Challenges

- **Model API Reliability**
  - Requires stable internet connection for model to be used.
  - Delays may occur if the server is under high load
- **Cross Platform Issues**
  - Microphone behavior differs between devices.
- **Hardware Variations**
  - Device microphone quality affects noise classification accuracy which may affect classification results.

## 4.3 Design Methodology

The mechanical noise detection system follows an object oriented design approach aided by UML diagrams to represent behavior and structure. Each functional module user authentication, audio detection, log handling, and profile management is implemented as an independent yet connected unit.

The frontend developed in HTML, while the backend uses Python based Flask. The machine learning model was trained and tested in Jupyter Notebook.

- **Separation of Concerns:** Each component such as frontend and backend serves a unique role to enhance maintainability.

- **Reusability:** Modular code design enables reuse of widgets, API calls, and preprocessing routines.
- **UML Modeling:** UML designs such Use case, activity, and sequence diagrams visually represent the workflow.
- **Iterative Refinement:** The design evolves through user testing and iterative feedback.

This methodology ensures that the application remains efficient, scalable, and easy to maintain across future updates.

## 4.4 High Level Design

The high level design displays how the major elements of the system are organized and how they cooperate to provide noise detection. At this level, we do not focus on code or but rather on the structural view of the system. The aim is to show how the web application, backend, and database work together as a complete solution.

### 4.4.1 Conceptual or Logical View

The conceptual or logical view explains how the system's components behave with each other without going into the code level details. This view focuses on the main modules, the responsibilities each component has. The user interacts with the web application, which captures noises and sends them to the backend. The backend analyzes the noises using the machine learning model and returns classification results along with noise reduction, which are then stored locally in SQLite. Each module works independently but communicates in a predictable and structured manner.

#### 4.4.1.1 Component Diagram

This diagram shows the different components of the app and how they work, interact with each other. There are two components, Web app and Flask. The first component is the web application which is basically the front end. It handles user's input, animations and microphone component as well as SQL database.
The second component is the backend which is Flask that handles API calls as well as model itself. It is light weight and accessible from anywhere.

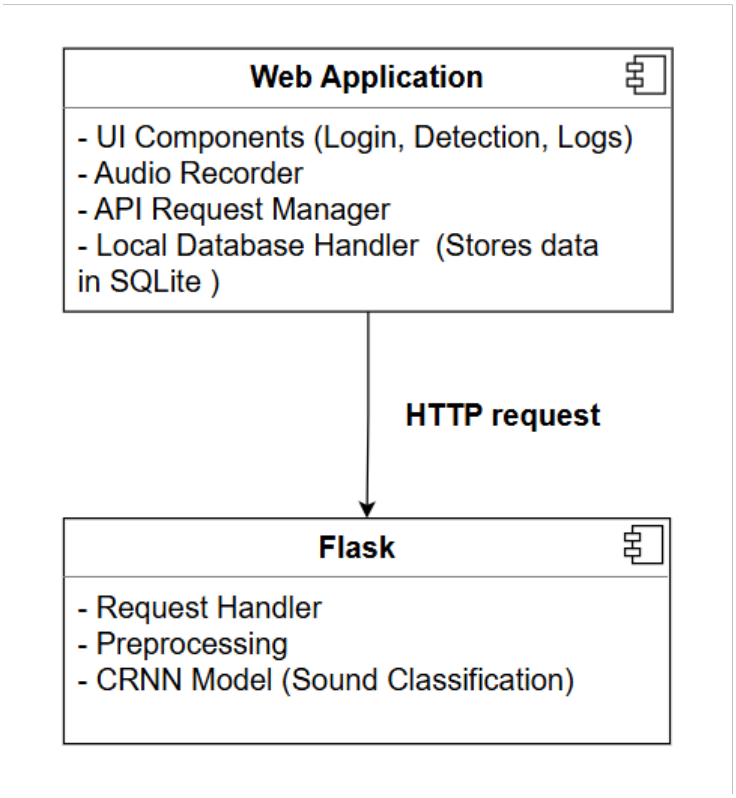


Figure 4.1: Component Diagram

### 4.4.2 Data Flow Diagram

The data flow diagram (DFD) in Fig 4.6 shows how data moves through the mechanical noise detection system. It identifies key data elements such as user input, audio data, spectrograms, classification results, and log entries, and maps their progression through the system. Data originate from the user and are captured by the web application, where the audio is processed into Log Mel spectrograms and passed to a machine learning model for classification where the audio is first checked if it has noise or not. Then if it has noise, it detects the source of noise and reduces it. The resulting classification is logged and stored in the database. The data flow diagram helps visualize the flow of data within the system irrespective of hardware implementation.

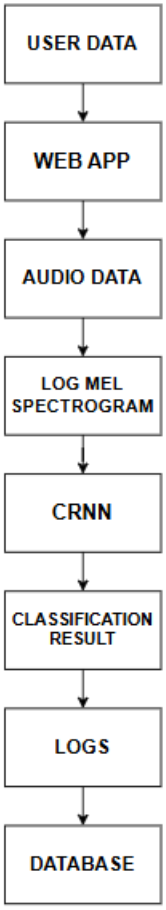


Figure 4.2: Dataflow Diagram (Top-to-Bottom Layout)

## 4.5 Low Level Design

The low level design involves 2 diagrams. One is activity and the other is system sequence diagram. They represent the control flow and behavior of system when interacted by the user.

The activity diagram of the application starts from the splash screen which is loading screen and moves to the log in screen. Users are then presented with options such as login , register and forgot password. After successful authentication, the users arrive at the main screen from where they can start noise detection or access the drawer for further capabilities. If noise detection is initiated, the application records the noise, analyzes it with a neural network, shows the results, and logs them. The drawer provides access to logs, account settings, or log-out. The flow is complete after the user leaves or logs out of the system. This diagram clearly defines sequences of user interaction and system responses across the app life cycle.

### 4.5.1 Activity Diagram

This section shows the overall activity of different parts of the system. This diagram focuses on the flow of decisions, user action, and system's response to that particular action. The system's

activity diagrams are broken into different actions such as login process, register and forgot password. Once the user is logged in they can perform noise detection, change password , view logs and logout.

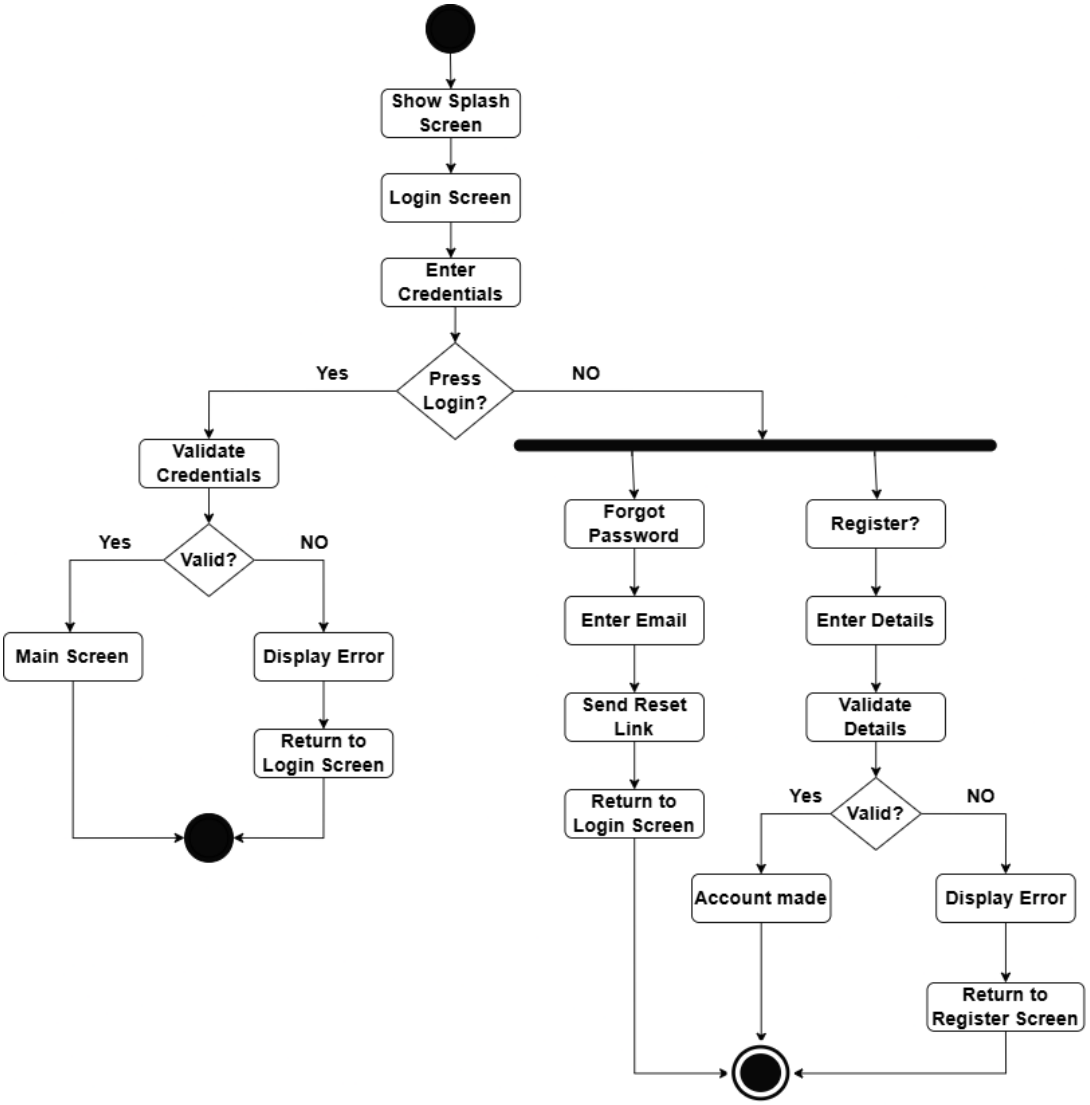


Figure 4.3: Activity Diagram of login,register and forgot password screen

The figure 4.4 displays the activity flow of home screen. The user can click on the noise button to detect noise and classify them. They can click on logs to view the previously detected noises. They can also change their account details and logout of the current session.

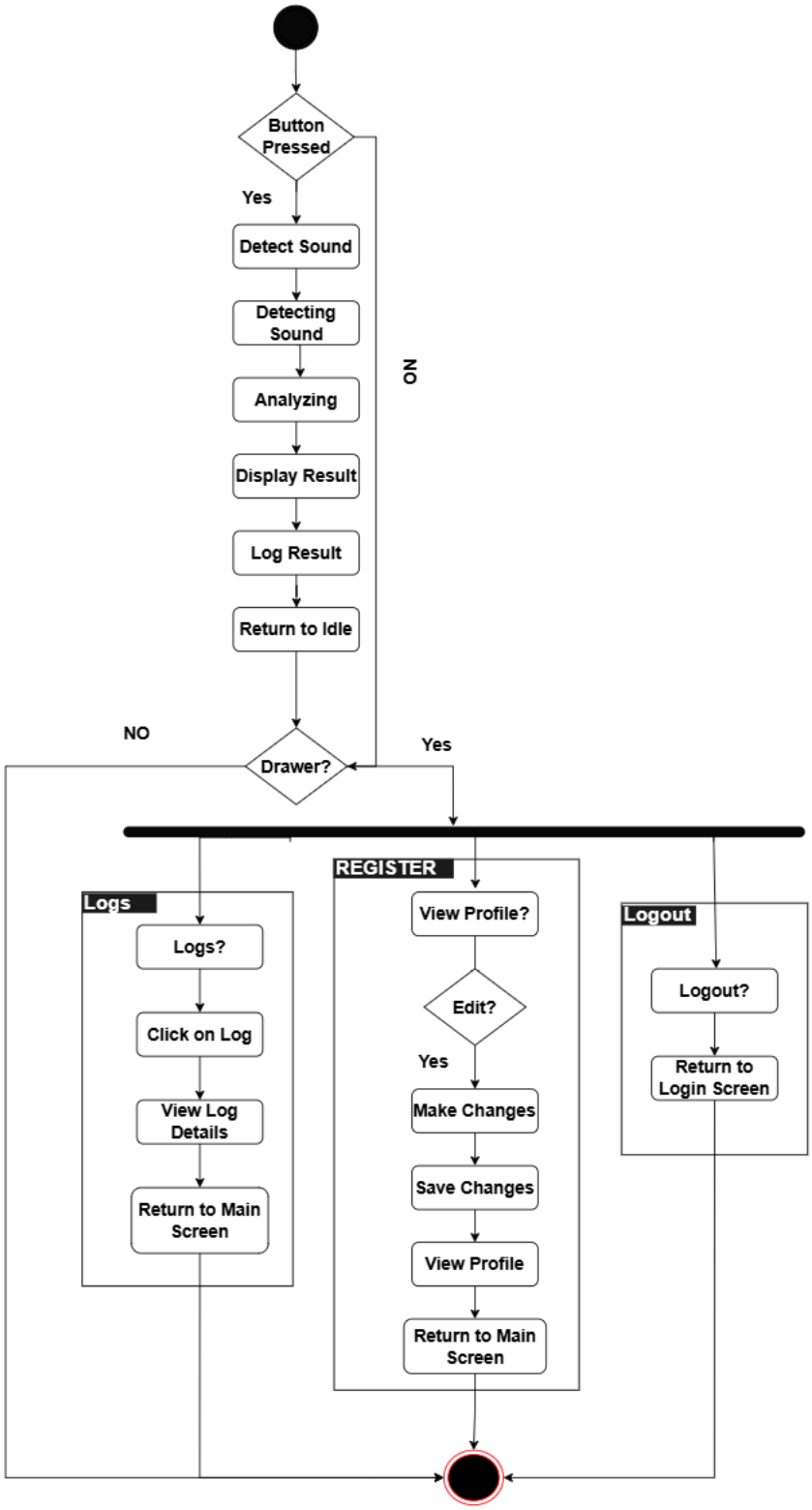


Figure 4.4: Activity Diagram of Main screen

### 4.5.2 System Sequence Diagram

System sequence diagrams are used to represent the dynamic behavior of a system by showing how users interact with it and how the system responds over time. Each diagram focuses on a specific use case, highlighting the exchange of messages between the actor (user) and the system components. For the mechanical noise detection application, SSD provide a clear view of how core functionalities such as registration, login, noise detection, and log management are executed step by step. Following are the sequence diagrams of our systems along with descriptions:

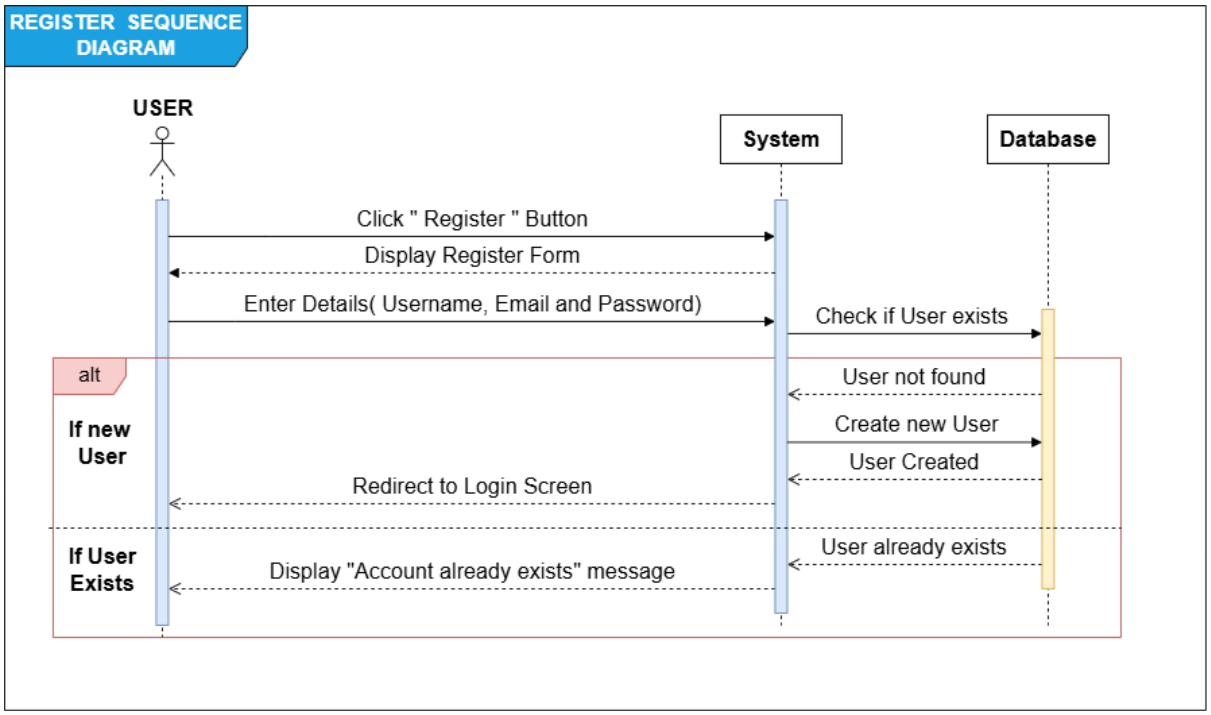


Figure 4.5: Sequence Diagram for User Registration

Register sequence shows how a new user interacts with the system to create an account. The process begins with the user submitting registration details, which are validated by the system. Once verified, the information is stored in the database and a confirmation is sent back, allowing the user to access the application with their new credentials.

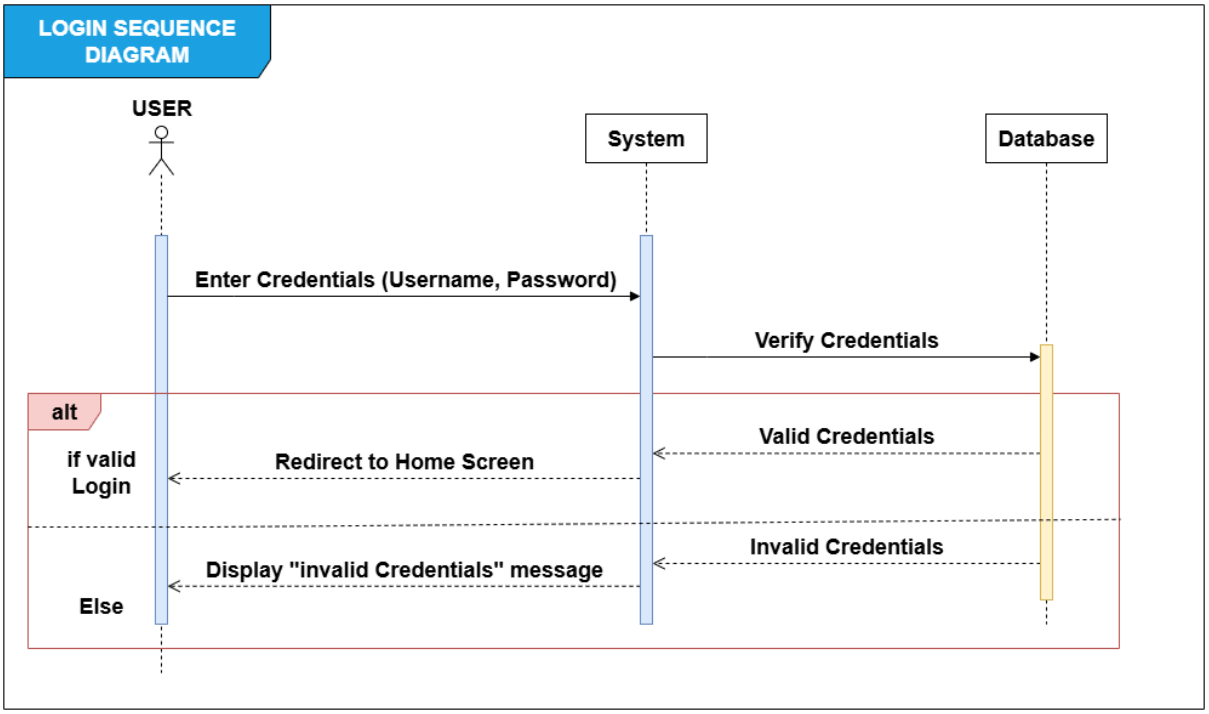


Figure 4.6: Sequence Diagram for User Login

Login sequence illustrates the steps for secure login. The user enters their username and password, which the system forwards to the authentication service. The credentials are checked against stored records in the database. If valid, access is granted; otherwise, an error message is returned.

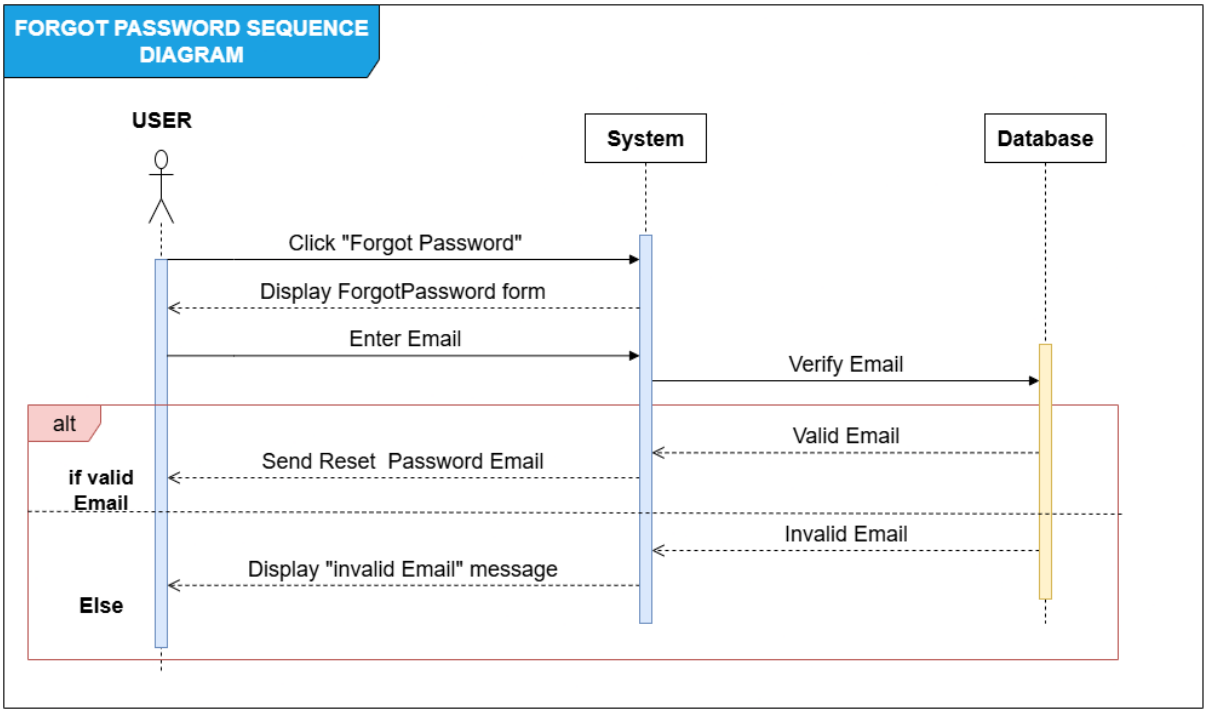


Figure 4.7: Sequence Diagram for Forgot Password

Forgot sequence outlines the password recovery process. When a user requests password

reset, the system verifies the registered email address, generates a reset link or token, and sends it to the user. Upon following the link, the user can set a new password, which is updated in the database.

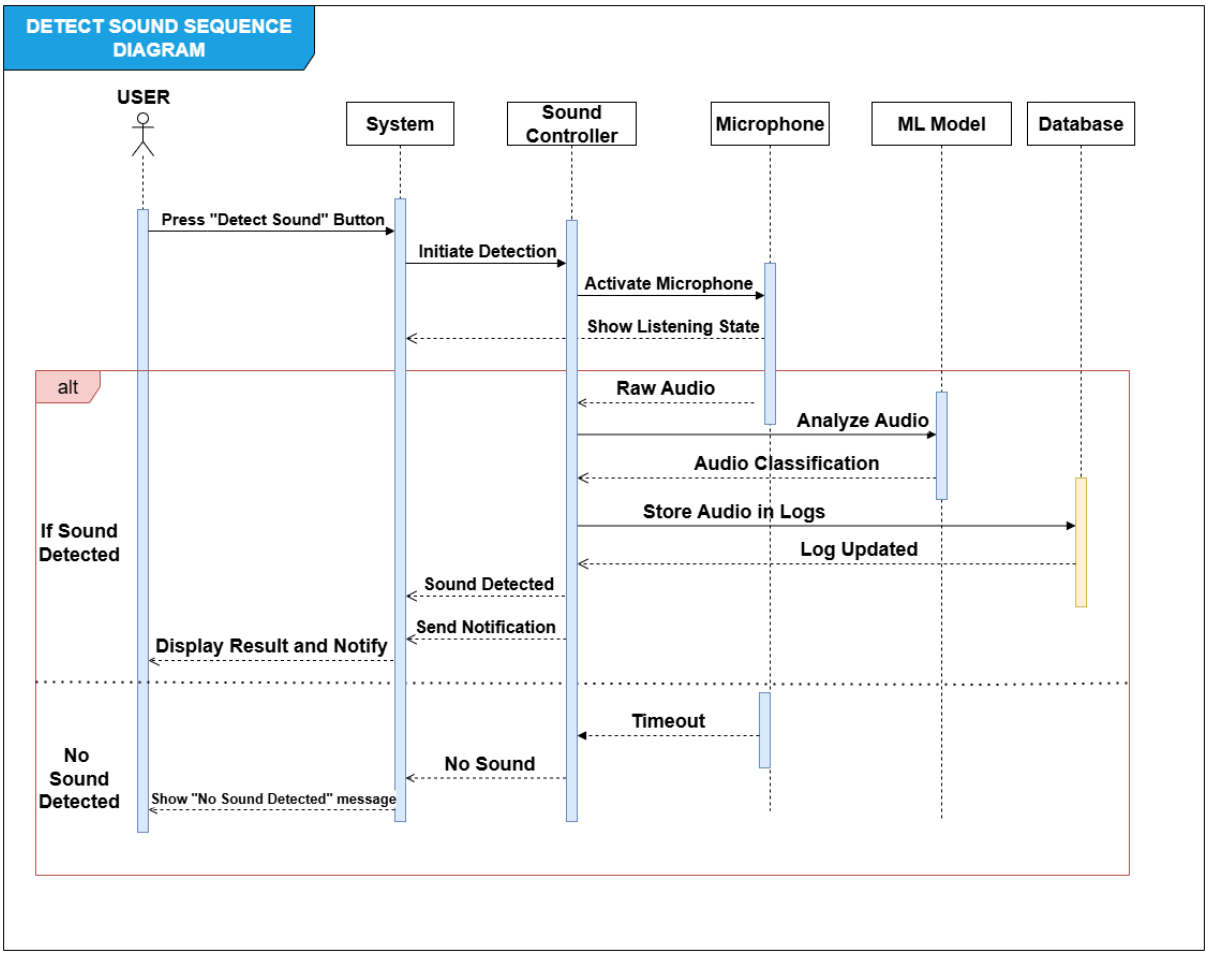


Figure 4.8: Sequence Diagram for Noise Detection

Noise detection sequence represents the core functionality of the system. The user triggers the noise detection feature, and the web app captures audio through the device microphone. The captured noise is preprocessed and classified by the machine learning model. Based on the result, the system notifies the user and logs the detection.

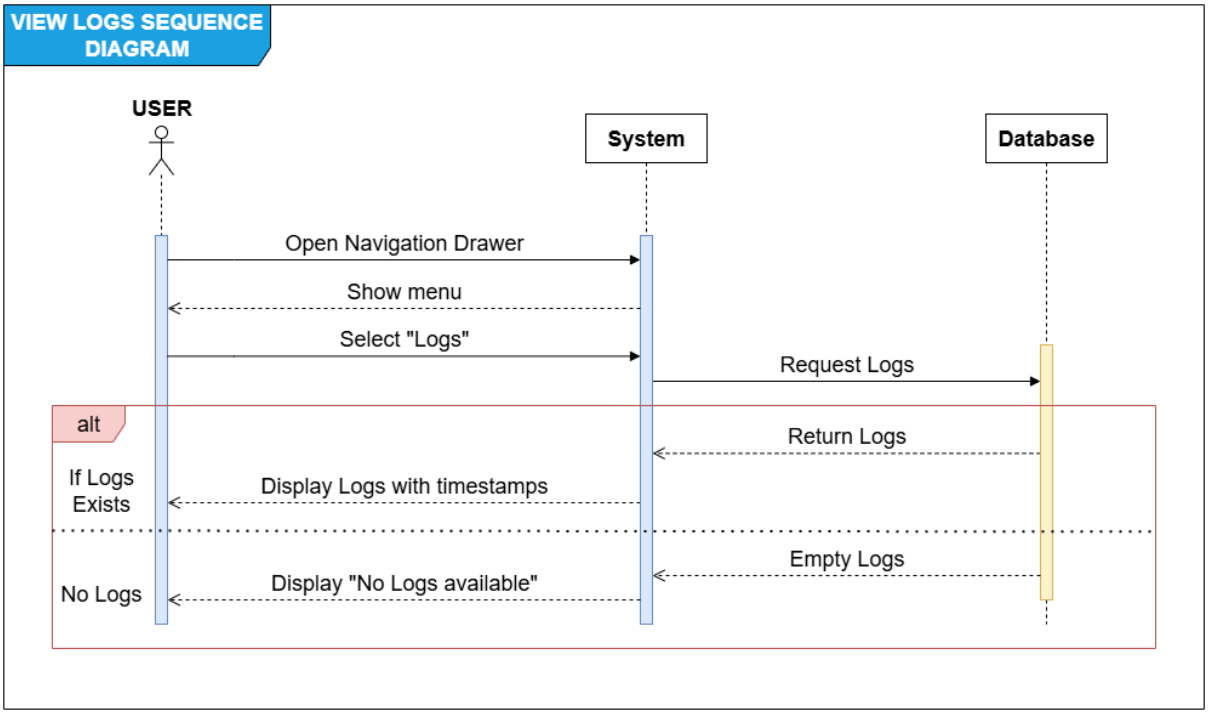


Figure 4.9: Sequence Diagram for Viewing Logs

View logs sequence diagram demonstrates how users can access stored noise detection records. The user requests to view logs, and the system retrieves the relevant entries from the database. The logs are then displayed on the user interface, showing detection details such as timestamps, classifications, and spectrograms.

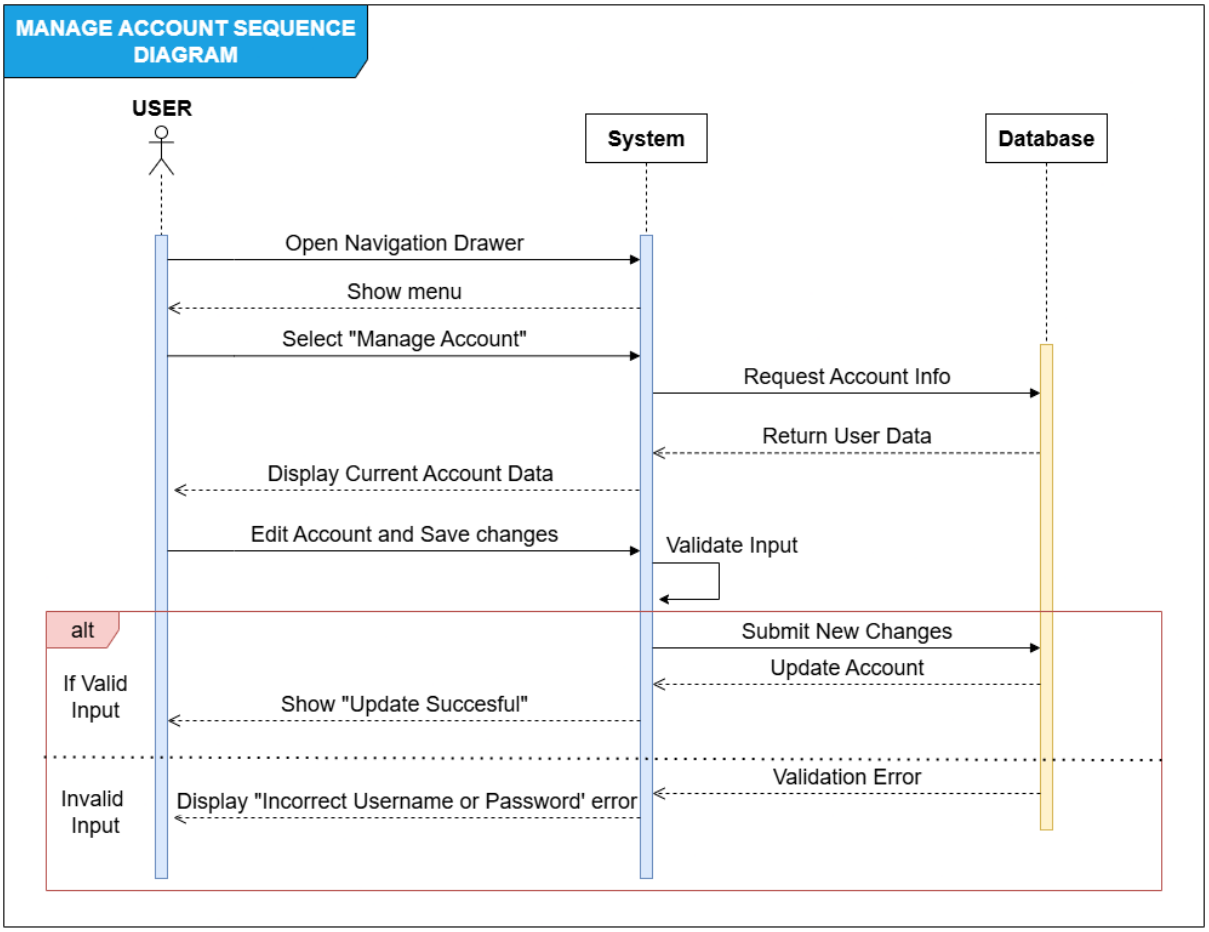


Figure 4.10: Sequence Diagram for User Profile Management

Manage account sequence diagram shows how users can manage their profiles. The process includes retrieving current profile data, making changes (such as updating email or password), and saving updates back to the database. The system confirms the update and reflects the changes in the user account.

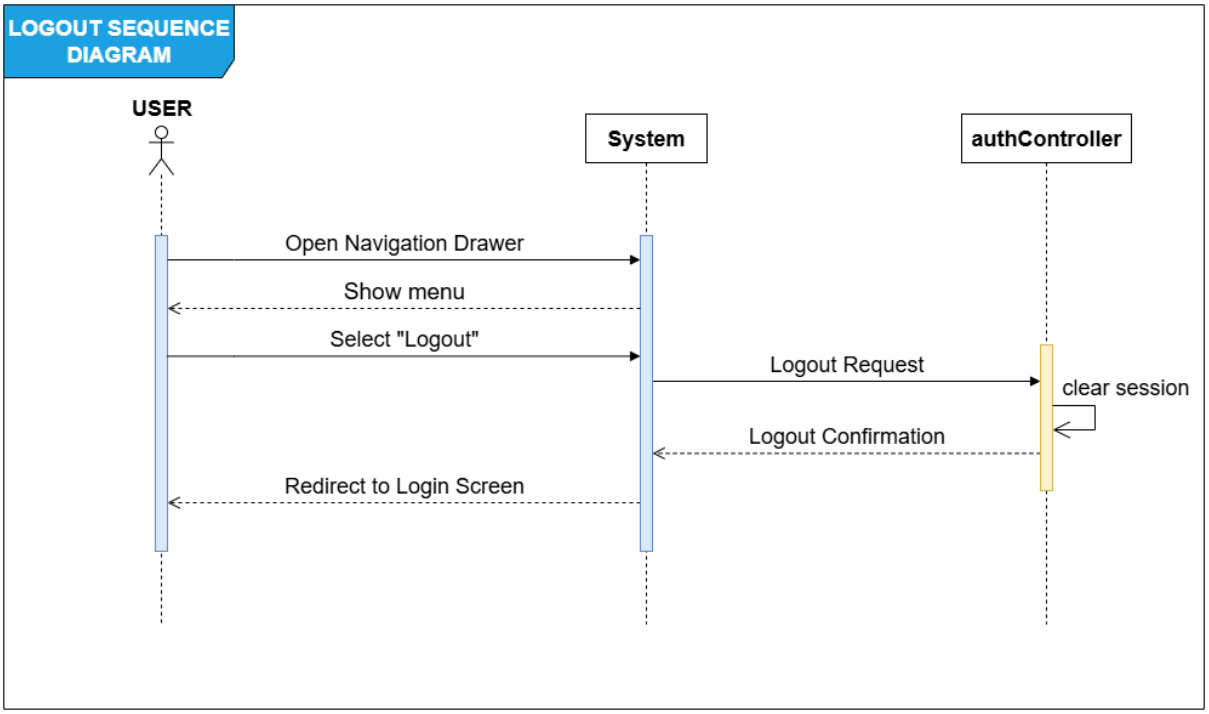


Figure 4.11: Sequence Diagram for Logout

Logout sequence diagram explains the logout process. When the user chooses to log out, the system terminates the session, clears any temporary authentication tokens, and redirects the user back to the login interface to ensure security.

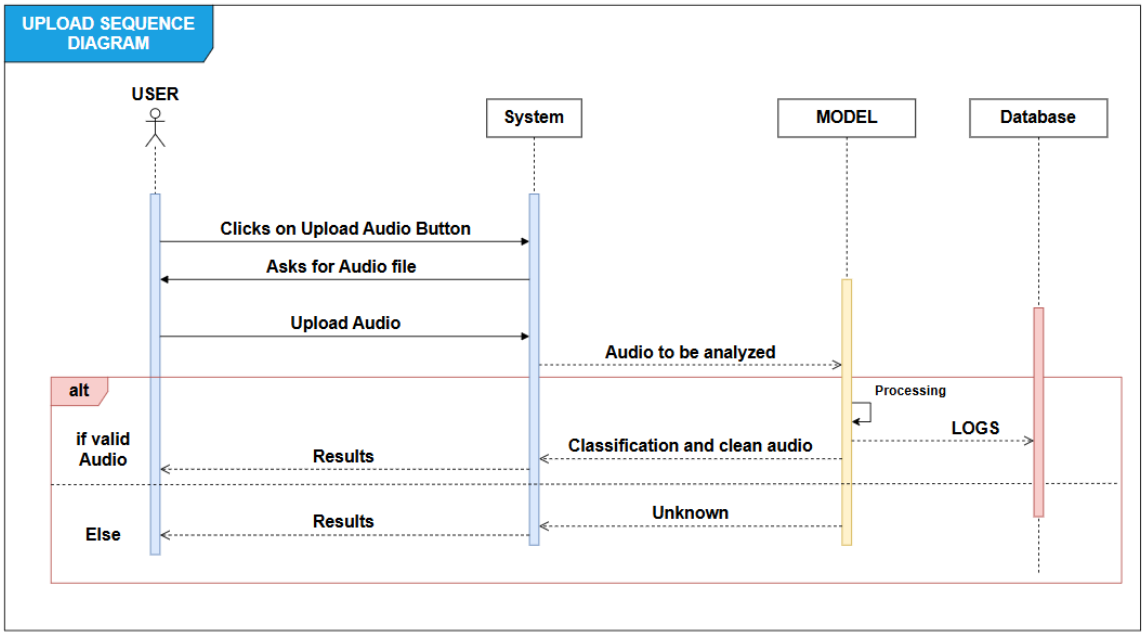


Figure 4.12: Sequence Diagram for Uploading Audio

Logout sequence diagram explains the audio upload process. When the user chooses to classify a sound, they click on upload audio button, which prompts them to select the audio file

they want to analyze. After selecting the audio the system classifies whether it has noise or not. If it has noise it identifies it, reduces the audio and return the classification result. The result is stored in Logs.

## 4.6 Database Design

The database schema ensures structured and efficient storage of user information, detection logs, and authentication credentials. The backend uses a lightweight SQLite database, which is ideal for web and small scale applications since it stores data locally on the device and requires no separate server configuration. SQLite provides sufficient performance for logging, querying, and retrieving user data in offline or low-network environments.

The database schema consists of two primary tables that are User, Detection Log. Each table stores essential information required for authentication, classification tracking.

### 4.6.1 Users Table

Table 4.1: Users Table

| Field Name | Type | Description |
|---|---|---|
| id | INTEGER (PK) | Auto-incremented unique user ID |
| username | TEXT (UNIQUE) | Unique username for account login |
| email | TEXT (UNIQUE) | User's email address |
| password | TEXT | Hashed password for authentication |
| created_at | TEXT | Date and time when account was created |

The User table contains the core details required for authentication and account management. Each user has a unique username and email, and their password is saved in a hashed form for security. This table acts as the foundation for linking user activity and detection logs throughout the system.

### 4.6.2 Noise Detections Table

Table 4.2: Noise Detections Table

| Field Name | Type | Description |
|---|---|---|
| id | INTEGER (PK) | Auto-incremented detection ID |
| user_id | INTEGER (FK) | References `users.id` |
| noise_class | TEXT | Classified noise label (Normal/Abnormal) |
| confidence | REAL | Confidence score returned by ML model |
| timestamp | TEXT | Time when the detection occurred |

Noise Detection table stores every noise that the system processes and classifies. Each entry includes the predicted class, the confidence level returned by the Flask, and the time of detection. By linking each log to a user, the app is able to show detection history.

### 4.6.3 Data Synchronization and Persistence

The local SQLite database that is used in our application is used to store user information and noise detection logs directly on the device. This guarantees that the application can be used even in a state of no connectivity to the internet that is critical in places for example, the ship or in distant locations.

noise results are detected when the device is online, and Flask receives the results and stores the processed output in the SQLite for fast retrieval. It is an easy effective hybrid model; online and offline respectively.

To maintain storage utilization effectiveness, the previous log entries can be cleaned or deleted in case they surpass the permitted threshold. The lightweight design of SQLite makes it suitable in web applications with limited resources of the device.

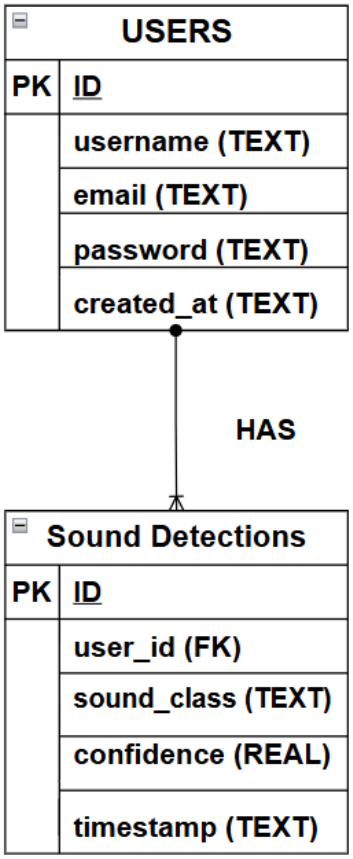


Figure 4.13: ERD Diagram

## 4.7 Summary

The following chapter describes the architecture and design of the application. It describes the primary elements of the system including user interface, noise processing, Machine learning classification, backend services and database. Device hardware limitations, real time processing and cross platform compatibility are taken care of by design constraints. System behavior is represented with the help of UML diagrams, e.g. use case, activity, sequence, data flow, ERD.

The design approach is based on object-oriented design philosophy and incremental model. High-level and low-level designs are also given to show clarity.

# Chapter 5

# System Implementation

### 5.0.1 System Architecture

Our system is developed using iterative development model, where the app was designed, tested and refined after every feedback. Each cycle , new functionality was added and the app slowly improved and fulfilled the functional requirements. The final product was reliable, user friendly and met all functional requirements.

The architecture of our app is lightweight , easy to access as front end is based on WEB which enables compatibility with all types of devices and backend is based on Flask which is python based and works well with models. The web app is light weight and accessible from basically anywhere as long as you have internet connection.

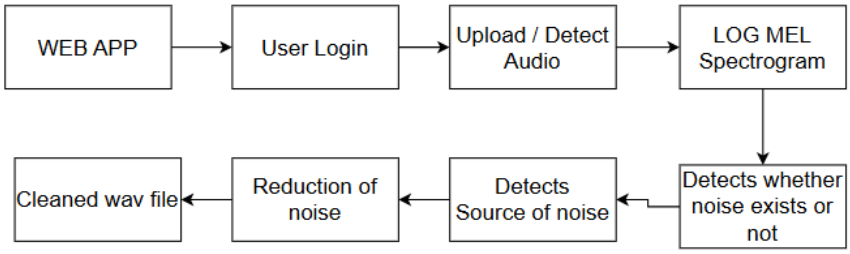


Figure 5.1: System Architecture

#### 5.0.1.1 Internal Components

- **Web Interface:** Provides users with all the functionality needed such as login , register, forgot password etc. Basically user friendly interface with all the necessary functionality.
- **noise Processing Module:** The noise processing module records audio through device's phone.
- **Machine Learning Model:** It classifies the noise and returns the result to front end.

- **Back End Processing:** The backend is based on Flask along with the model. It receives requests from front end along with noise and classifies it through model and returns the result for front end to display to user.
- **Database:** Has user's information and logs of all the noise detected stored locally on user's device.

#### 5.0.1.2 Component Functionality

- The Web App is basically used for the Frontend of our app. It displays the user friendly interfaces ranging from log in to detect noise to log out.
- The noise Processing Module records noises using the device's own microphone and sends the instance which is 3 seconds long to backend. The noise is preprocessed before being sent to backend.
- The CRNN Model, is a convolutional recurrent neural network. That was trained on QiandaoEar22 dataset. It classifies the audio based on noises and returns the result.
- The Flask which acts as a Backend, handles audio classification requests from Frontend , sends them into the model for classification and returns the result to Frontend for displaying to the user and storing purposes.
- The SQLite is a local database, that uses device's own storage to store logs (classification result) and user's information.

#### 5.0.1.3 Communication Between Components

- The Frontend which is based on web communicates with the Flask via requests. It sends instances of noise (3 seconds) to backend.
- The app records these instances of noise using the device's own microphone and sends it to backend for preprocessing and classification and noise reduction.
- The noise is converted into preprocessed and fed into CRNN model and classified accordingly.
- Flask then sends the classification results back to Frontend. The user is notified about the classification along with specifications of the noise and reduced noise graph.
- All the detected audios are stored as logs in SQLite and can be accessed by clicking on Logs button inside the app.

#### 5.0.1.4 Tools and Technologies

- **Python and Jupyter:** Jupyter notebook is used to train model, which in our case is CRNN. Jupyter Notebook uses python for it's operation.

- **Flask :** Flask is used for backend in our system. It acts as a bridge between web app and model to send audio requests and receive classification results.
- **SQLite:** SQLite is a local database which uses device's own storage to store user's information along with all the detected noises (logs).

#### 5.0.1.5 Development Environment and Languages

- **Python:** It is a high level language which is used to train model and preprocess audio and act as a backend using Flask.
- **SQLite:** Used for storing structured noise logs and user data locally.
- **VS Code and Jupyter Notebook:** Used for backend API development and ML experimentation, including training, evaluation, and model export for deployment via Flask.

#### 5.0.1.6 Processing Logic and Algorithms

1. Capture audio input using the device microphone via the frontend.
2. Preprocess the audio.
3. Send the audio to the Flask, which routes it to the trained CRNN model.
4. The model classifies the noise.
5. Return classification results to the web app, display them along with a timestamps and confidence, and notify the user.
6. Store detection results in the SQLite database for future retrieval.

## 5.1 GUI Design

Following are the prototypes of our system that were designed according to the requirements. They are not permanent and may change in the future.

### 5.1.1 Login

The figure 5.2 shows the graphical user interface of the login screen. User can log into the app by entering their Email/Username and Password.

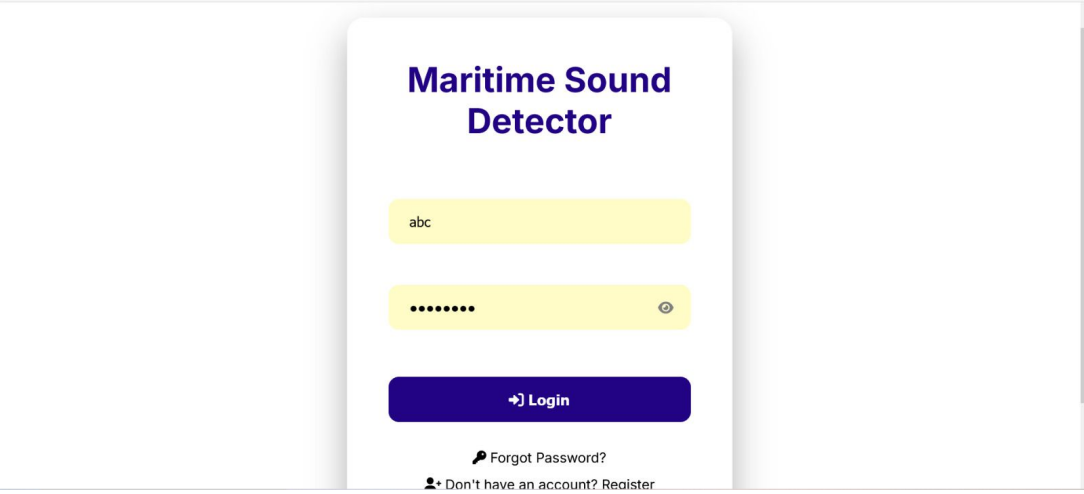


Figure 5.2: Login Screen

### 5.1.2 Register

The figure 5.3 shows the graphical user interface of the register screen. When the user clicks on the register button in the login screen, they are taken to registration screen where they can create a new account by entering their Username, Email and Password.

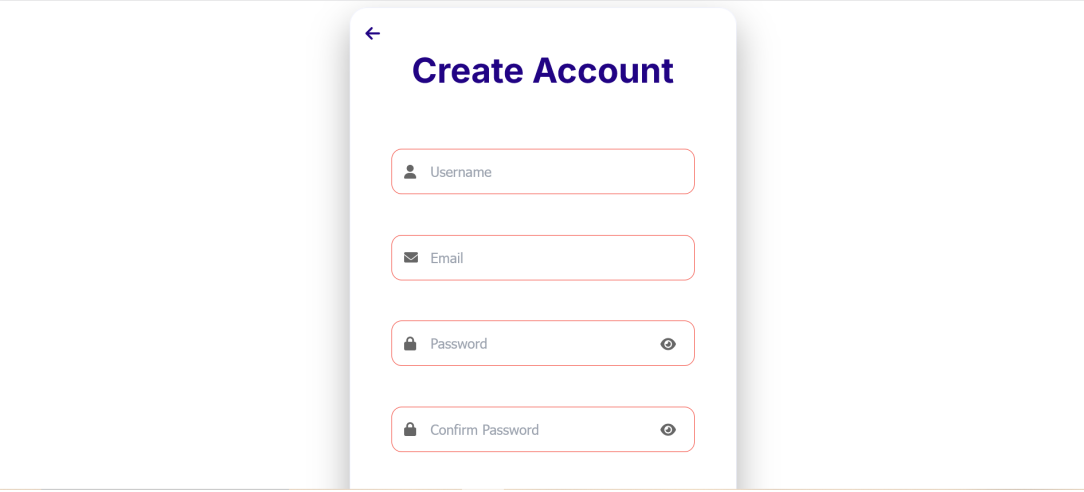


Figure 5.3: Register Screen

### 5.1.3 Forgot Password

The figure 5.4 shows the graphical user interface of the forgot password screen. When the user clicks on the forgot password button in the login screen, they are navigated to a new screen where they can enter their email and receive a reset link where you can update your password to login with.

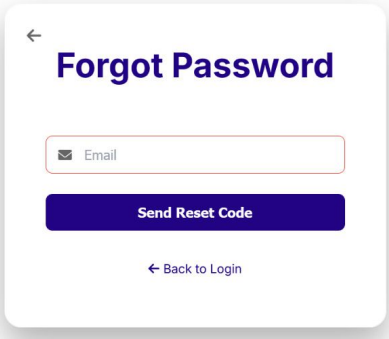


Figure 5.4: Forgot Password Screen

### 5.1.4 Home

The figures 5.5, 5.6, show the graphical user interface of the home screen with its different options. It contains a noise button which detects noise and displays animations while capturing noises. A drawer with options such as logs, Manage account, and logout. It also has the option to upload audio.

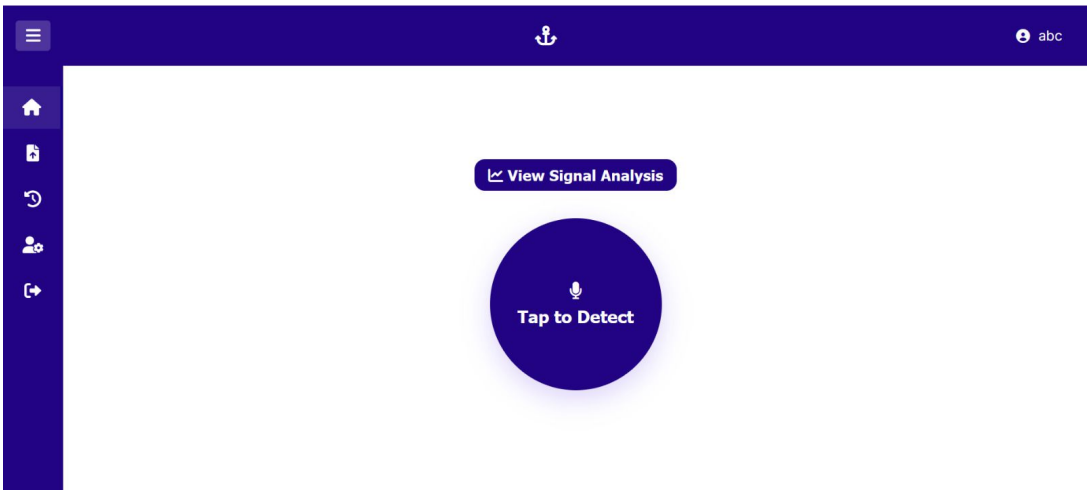


Figure 5.5: Home Screen

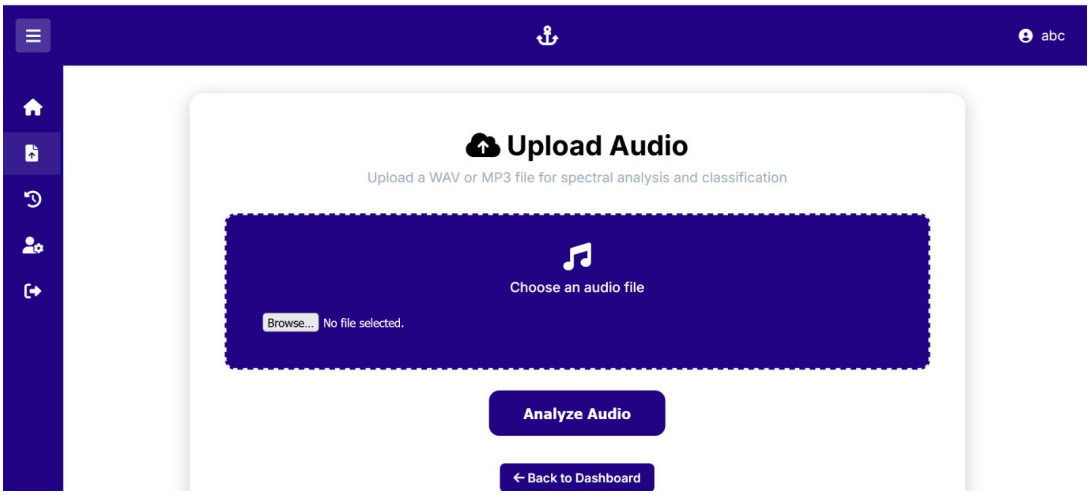


Figure 5.6: Audio Upload Screen

### 5.1.5 View Logs

The figure 5.7 shows the graphical user interface of the view logs screen. When the user clicks on the view logs button in the drawer, they can view the entire history of detected noises.

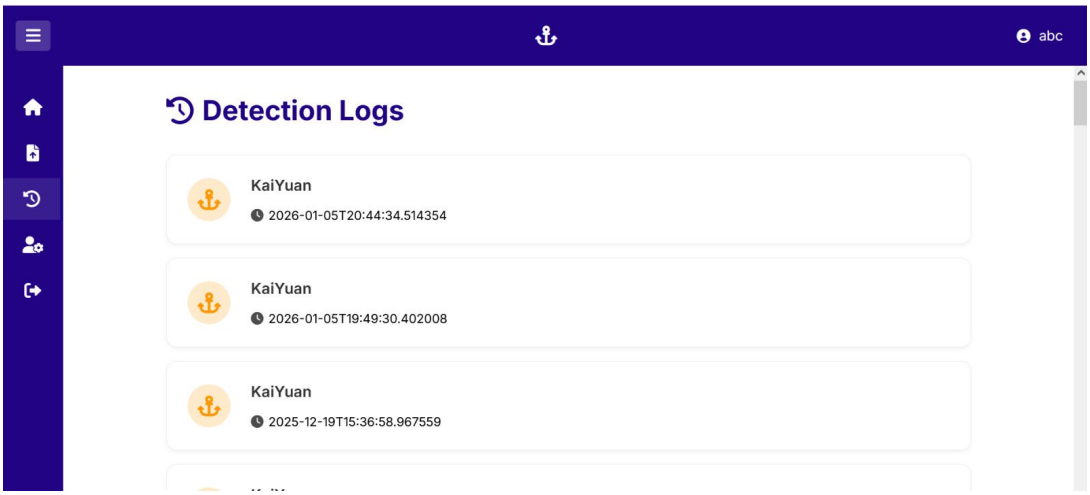


Figure 5.7: View Logs Screen

### 5.1.6 Manage Account

The figure 5.8 shows the graphical user interface of the Manage account screen. When the user clicks on the manage account button in the drawer, they can edit their profile information such as Username, Email and Password.

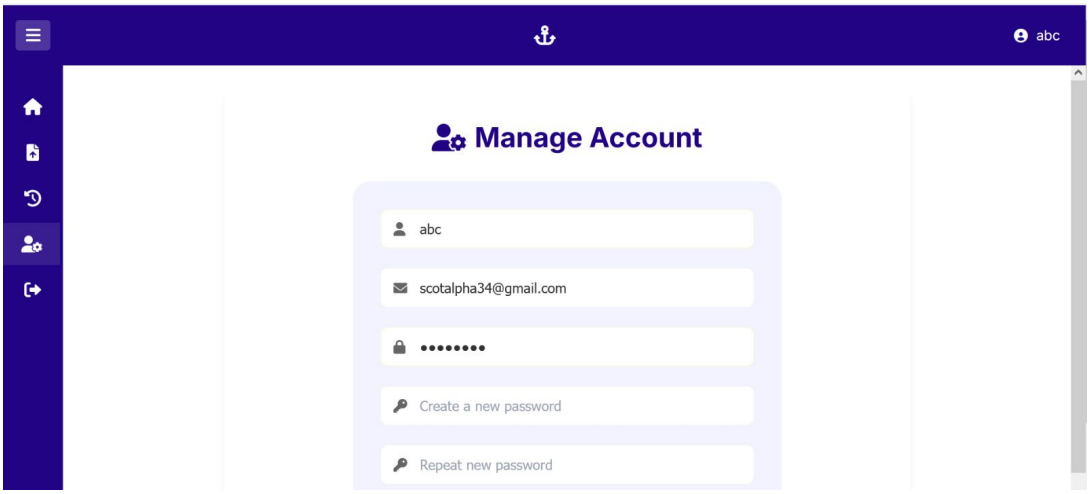


Figure 5.8: Manage Account Screen

### 5.1.7 Logout

The figure 5.9 shows the graphical user interface of the logout screen. When the user clicks on the logout button, they are led back to the login screen.

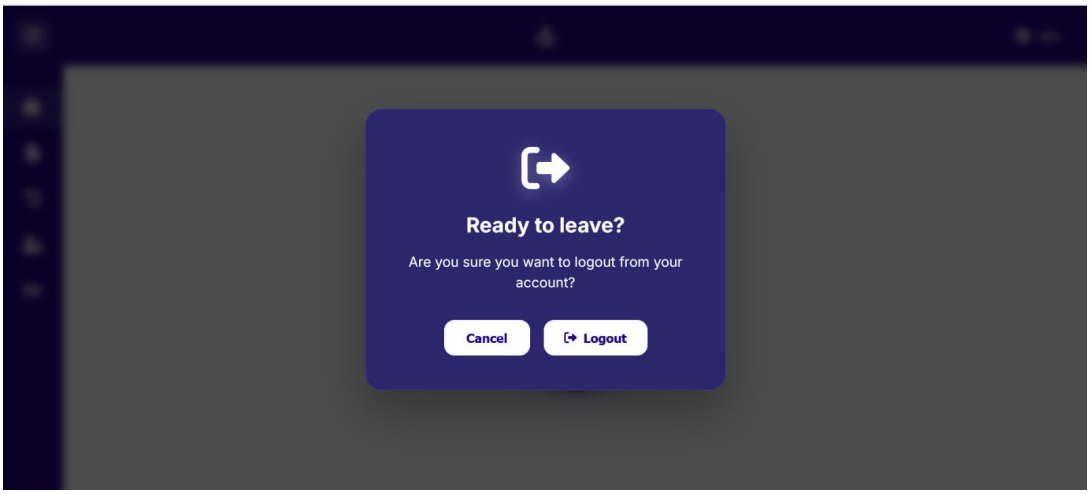


Figure 5.9: Logout Screen

## 5.2 Summary

This chapter explains how the system was implemented based on a modular architecture integrating the front-end, Flask back-end, CRNN machine learning model, and a local SQLite database. Tools such as VS Code, Jupyter Notebook, were used for development. The audio processing pipeline includes recording noise through the web app, and classifying noises using the CRNN model. The chapter ends with prototypes of the app itself.

# Chapter 6

# System Testing and Evaluation

System testing and Evaluation chapter presents the overall testing for this project. The reason why we conducted this testing phase was to ensure the system does not crash by wrong input and validated each module from frontend to backend . Testing ensures that the system is accurate, reliable, and stable for users to use.

## 6.1 Testing Strategy and Methodology

Following are the different types of testing performed to ensure the system was reliable:

- **Functional Testing:** This type of testing ensures that each feature acts according to the requirements specified.
- **Integration Testing:** This type of testing ensures that frontend , backend and database communicate with each other without any errors.
- **System Testing:** System testing evaluates the complete process from start to end.
- **Non Functional Testing:** Non functional testing evaluates performance, system's response, compatibility of the system.
- **Usability Testing:** It ensures that the system is easy to use for users.
- **Exception Handling Testing:** This type of testing ensures that the system, handles itself when it is presented with wrong or unexpected input.

## 6.2 Functional Testing

Functional testing was performed to make sure every feature was stable and working accordingly to functional requirements. Following are the tests performed for each feature in tables below:

### Test Case F01: User Registration

Table 6.1: Test Case F01: User Registration

| | |
|---|---|
| **Objective** | Verify that new users can register successfully with valid details. |
| **Input / Action** | User enters valid username, email, and password in the web app. |
| **Expected Output** | Backend stores user details in SQLite and displays confirmation. |
| **Actual Output** | Registration completed and confirmation shown. |
| **Status** | Pass |

User registration test case evaluates that new users can successfully register their account on our system by entering their details. The system stored user's details securely in the database.

### Test Case F02: User Login

Table 6.2: Test Case F02: User Login

| | |
|---|---|
| **Objective** | Ensure that valid users can log in successfully. |
| **Input / Action** | Enter registered credentials; User data is verified via SQLite. |
| **Expected Output** | User authenticated and redirected to dashboard. |
| **Actual Output** | Login successful, dashboard displayed. |
| **Status** | Pass |

User login test case validates that registered users can log into the app using the correct credentials. The credentials were cross verified with stored credentials and if they match, the user is redirected to home screen.

### Test Case F03: Forgot Password

Table 6.3: Test Case F03: Forgot Password

| | |
|---|---|
| **Objective** | Check that users can reset passwords using registered email. |
| **Input / Action** | Enter registered email in app and request reset link. |
| **Expected Output** | Backend sends verification code or link to user's email. |
| **Actual Output** | Password reset email received and processed successfully. |
| **Status** | Pass |

Forgot password test case was made to make sure, if the users forgot their password they can recover it quickly and log into the app.

### Test Case F04: noise Detection

Table 6.4: Test Case F04: noise Detection

| | |
|---|---|
| **Objective** | Verify that the app captures and processes noise correctly. |
| **Input / Action** | User initiates recording through the "Detect Noise" button. |
| **Expected Output** | Audio captured and sent to Flask for further processing. |
| **Actual Output** | noise recorded and successfully sent to backend. |
| **Status** | Pass |

noise detection test case checks, if the system can capture the noise properly via microphone and send it to backend for classification.

### Test Case F05: Classification

Table 6.5: Test Case F05: Classification

| | |
|---|---|
| **Objective** | Ensure backend communicates with ML model and retrieves accurate classification. |
| **Input / Action** | Processed audio is sent by Flask to the model endpoint. |
| **Expected Output** | Classification result returned with confidence score. |
| **Actual Output** | Result returned successfully from model API. |
| **Status** | Pass |

Classification test was conducted to make sure the communication between backend and frontend was accurate and the noise classification result was accurate.

### Test Case F06: Log Management

Table 6.6: Test Case F06: Log Management

| | |
|---|---|
| **Objective** | Verify that detection results are stored and retrieved correctly. |
| **Input / Action** | After classification, user views history of detections in app. |
| **Expected Output** | Logs retrieved from SQLite and displayed with timestamps. |
| **Actual Output** | Data displayed correctly with full details. |
| **Status** | Pass |

Log management test evaluated that detection results were being stored in the database locally and shown correctly to the user.

**Test Case F07: Profile Management**

Table 6.7: Test Case F07 — Profile Management

| | |
|---|---|
| **Objective** | Ensure user can update and save profile data. |
| **Input / Action** | Modify user name or password in profile section. |
| **Expected Output** | Data updated and saved in database. |
| **Actual Output** | Profile updated successfully. |
| **Status** | Pass |

Profile management test made sure the registered users were able to successfully edit their account information such as username, email and password. It also ensured that the new data was updated in the database.

**Test Case F08: Logout**

Table 6.8: Test Case F08 — Logout

| | |
|---|---|
| **Objective** | Verify user can safely log out. |
| **Input / Action** | Click “Logout” in app. |
| **Expected Output** | Session cleared and redirected to login screen. |
| **Actual Output** | Logout successful. |
| **Status** | Pass |

Logout test made sure that the registered users were successfully able to exit the existing sessions and redirected to login screen by clicking the logout button,

## 6.3 Integration Testing

Integration testing validated the communication between the frontend , backend and database.

**Test Case I01: Web to Flask Communication**

Table 6.9: Test Case I01 — Web to Flask Communication

| | |
|---|---|
| **Objective** | Confirm that requests from web app are correctly handled by Flask. |
| **Input / Action** | App sends noise recording to backend endpoint. |
| **Expected Output** | Flask receives and processes request successfully. |
| **Actual Output** | Data transmitted correctly with valid response. |
| **Status** | Pass |

Web to Flask integration test validated whether the system can send the requests from Frontend that is HTML to Backend that being Flask. It also ensured that requests were being processed by Flask. Basically, communication between frontend and backend was accurate.

### Test Case I02: Flask to SQLite Synchronization

Table 6.10: Test Case I03 — Flask to SQLite Synchronization

| | |
|---|---|
| **Objective** | Verify that backend stores and retrieves logs from SQLite accurately. |
| **Input / Action** | User information and Classification result inserted. |
| **Expected Output** | Data integrity maintained across operations. |
| **Actual Output** | User data , Logs stored and fetched successfully. |
| **Status** | Pass |

Flask to SQLite test checked if the database could receive the classification information from Flask and store it accurately.

## 6.4 System Testing

System testing was performed to make sure the entire system worked perfectly from login to classification to storing results and logging out.

### Test Case S01: End to End Process Validation

Table 6.11: Test Case S01 — End-to-End Process Validation

| | |
|---|---|
| **Objective** | Validate full workflow (login to detection to classification to log view to logout). |
| **Expected Output** | All modules interact seamlessly and output correct classification. |
| **Actual Output** | All stages executed successfully. |
| **Status** | Pass |

End to End test validated whether the system behaved accurately from starting process that is login till the ending process that is logging out,

## 6.5 Usability Testing

Usability testing validated if the app's interface was simple enough for users to understand and use.

### Test Case U01: Interface Simplicity

Table 6.12: Test Case U01 — Interface Simplicity

| | |
|---|---|
| **Objective** | Ensure navigation and icons are easy to understand. |
| **Input / Action** | New users operate app without guidance. |
| **Expected Output** | Users navigate smoothly through core functions. |
| **Actual Output** | All participants reported intuitive layout. |
| **Status** | Pass |

Interface Simplicity, test was conducted to ensure that the user can use the app with minimal training required. The app was made to be very straight forward that even a person with minimal technical knowledge can use it.

## 6.6 Exception Handling Testing

Exception handling tests simulated how the app behaved if it is presented with something unknown or wrong input.

**Test Case E01: Permission Denied**

Table 6.13: Test Case E01: Permission Denied

| | |
|---|---|
| **Objective** | Verify app response when microphone permission denied. |
| **Expected Output** | App requests permission and pauses detection. |
| **Actual Output** | Permission dialog displayed correctly. |
| **Status** | Pass |

The permission denied test confirmed the system's response when microphone access permission wasn't granted by the user. When the noise detect button is pressed, a dialog box appears to ask permission to use microphone to detect noise. When permission is denied the app doesn't detect noise.

## 6.7 Summary

Chapter 6 involves the different testing phases the system went through to improve reliability, usability and provide accurate results. The functional testing tested every feature of app to ensure it worked according to requirements. The Integration testing ensured the Frontend, backend and database worked together. The system testing tested the entire work flow of the app. Exception handling testing tested the edge cases and how the system reacted to them. All of this testing led to major improvements in the app.

# Chapter 7

# Conclusion

## Overview

Our project "Identification and Reduction of Noise by Mechanical Systems Onboard Ships" focuses on detecting noises and classifying them accordingly by capturing the audio through device's microphone, preprocessing it and running them through CRNN model to classify them and return results to be displayed to user and stored locally on their own device.

The need for such automated system exists, to minimize human resources wastage on detecting and diagnosing such noises where they can spend these resources on something more productive. Early detection of noises can fix issues such as structural issues which can help in long term stability of the ship.

## Project Summary

Our system is made up of multiple technologies, such as HTML for the front end of the web app, Flask for back end, and SQLite for the database. The noise is classified using the CRNN model which is a Convolutional Recurrent Neural Network trained on QiandaoEar22 dataset.

The HTML is used for the web app which serves as a front end. It offers a user friendly interface that is easy to navigate, memorable. Users can log into the app using their credentials, if they are registered. Otherwise, they can register themselves. In case, they forgot their password, they can click on forgot password button and prompted with enter their existing email. Once they enter their email, the system sends a token to their email which they can use to create a new password.

Once, they log in to the app. They can detect noise by clicking the noise detection button. Once clicked, the app begins detecting noise using device's own microphone. Once the noise is detected, it sends the instance of that noise which is 3 seconds long to Flask.

Flask acts as a backend for our system along with the model. Flask, then preprocesses the audio and feeds it to the CRNN model for classification and Spectral gating for noise deduction. The noise is classified and returned to the Flask to send it to web to be displayed to the user. Once received, the web app displays the result and notifies the user. The result contains the classification of noise, detection time, and confidence score.

The result is then stored as logs locally via SQLite. They can be accessed through logs option. The app also has user profile management section which can be used to change the user information at any point and update it within the database itself.

Finally, the user can logout of the app using logout button which redirects them to the login screen.

## Achievements and Results

Our system achieved it's main objective that was detecting noises and classifying them. It was done, by using CRNN model which classifies them using different techniques to clean it which is provided by Flask. The reason why we used this model was that, it is hybrid and offers greater accuracy with good response time which is perfect for most devices.

Compared to QaindaoEar22 study which uses, deep learning model for ship identification. They achieved accuracy of "97.78%", for UUV using "DenseNet" with "MFCC" using binary classification and not reducing noise, meanwhile our system achieved "98%" accuracy for UUV using "CRNN" and "Log Mel spectrogram" with Multiclass classification and noise reduction aswell.

Comparison:

Table 7.1: Comparison between QiandaoEar22 and our System

| **Aspect** | **QiandaoEar22** | **Our System** |
|---|---|---|
| Dataset | QiandaoEar22 | QiandaoEar22 |
| Feature Representation | MFCC | Log Mel Spectrogram |
| Core Classifier | DenseNet | CRNN |
| Noise Detection Stage | Not included | Binary CRNN (Noise / Not Noise) |
| Noise Reduction | Not included | Lightweight CRNN Encoder Decoder [11] |
| Classification Strategy | Single-stage | Multi stage (Binary → Multiclass → Reduction) |
| Task Type | Ship identification | Noise identification and Reduction |

Confusion Matrix: Following is the confusion matrix for our app. Confusion matrix tells us about how well the model is performing by comparing predictions results with the actual results. It has 4 labels , True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). By using it we can calculate accuracy, precision, recall.

The accuracy of our model is "85.58%" for multiclass. "97.4%" for speed boat. "98%" for UUV and KaiYuan.

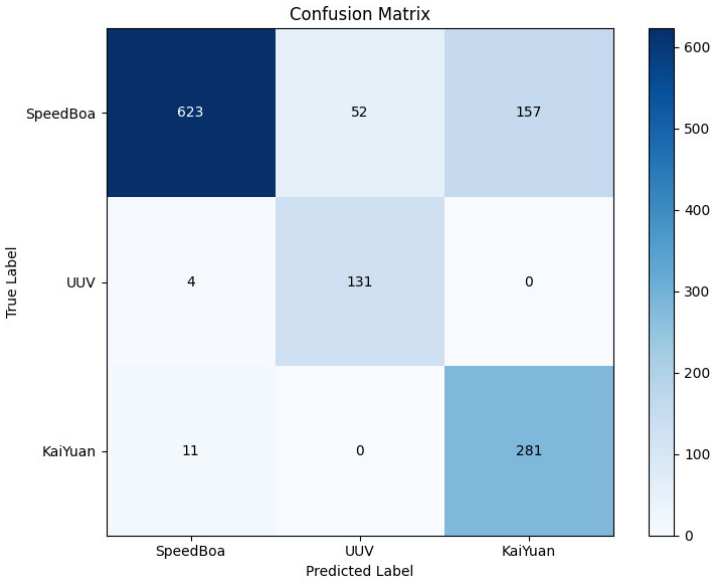


Figure 7.1: Confusion Matrix

Following are the key points from the results:

- **High accuracy:** The model demonstrated high accuracy in noise classification.
- **Compatibility:** The app was made in such a way that it could run in most devices easily.
- **User experience:** The app was made to be user friendly, easy to use for non technical and technical persons alike.

## Challenges and Limitations

While our system met the functional requirements, there were still some challenges faced during it's development. One of the biggest challenges we faced was model selection that would run smoothly on devices and provide accurate results. Looking at the dataset and the limitations of device's hardware, the best model to be used in our case was CRNN. It is fast , accurate and can be easily used on most devices.

The limitations of this project involves hardware of the device itself. The quality of it's microphone, has effect on audio capturing and classification. The better the quality, the quicker it can detect and classify. Also another limitation is the internet connection, as the app is web based which makes it accessible only on internet, because of this the quicker it can send and receive request, the faster it can classify and return result.

The current system can only detect and classify noises, but does not include root cause analysis of that particular noise.

## Impact and Applications

The automated system offers multiple advantages for both industrial and research purposes. In navy, it can be used for noise detection and classification which saves human resources and lets them diagnose the issue early and fix it reducing resources such as time spent on detecting noise, maintenance cost.

Since, our system is based on tools and technologies that are open source. It is easy to be customized, extended for new requirements. Thus, making it scalable.

## Future Enhancements

While our system is limited to detecting and classification and reduction only, it can be extended and include the following enhancements to make it a proper solution:

- **IOT:** Since, mobile devices may or may not be used in every ship. Thus, embedding this system on devices such as Raspberry Pi along with microphone at different areas of ship can make the noise detection very scalable and make it on site monitoring system.
- **Dataset:** Increasing the dataset to include more noises will make this system more accurate and classify more noises.

## Summary

This chapter involves the conclusion of previous 6 chapters. It starts with overview of the system. The system focuses on detecting noise, classifying them , displaying results and storing them locally on the user's device. The need for such system exists is to automate the process of manually finding noisy area and diagnosing the issue and fixing it. The system automates the noise identification and classification part. Thus saving time of the user.

# Appendix A

# User Manual

## Overview

### A.0.1 Purpose

The Noise Detector app is a web based application that is designed to assist naval crew members to detect, identify, and reduce noise in real time. This manual provides a brief guide on how to use the system.

## A.1 System Requirements

Following, are the specifications needed for this app.

- **Device:** A mobile device , laptop or computer with functional microphone.
- **Software:** Modern web browsers that can support HTML, JavaScript pages.
- **Internet Connectivity:** A stable internet connection.
- **Storage:** Local device storage to store logs and user profile information.

## A.2 Getting Started:

### A.2.1 Creating your Account:

Launch the web application, and on the login screen click on Register button that is at the bottom of login form. It will take you to Registration form where you can enter valid credentials and submit the form to create new account.

The figure A.1, displays the registration form which a new user has to fill in order to create a new account.

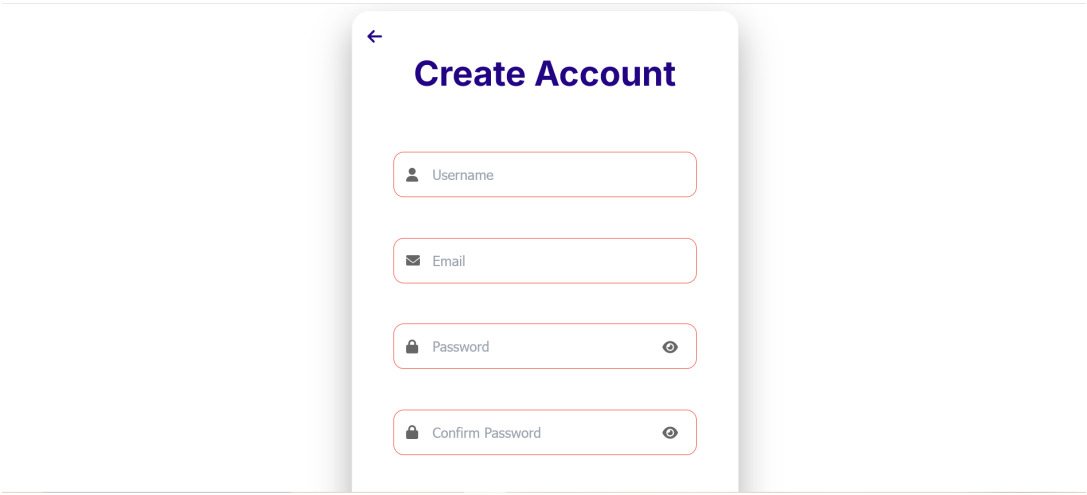


Figure A.1: Register Screen

### A.2.2 Login:

Open the login form and enter the registered credentials. Click on login button and once the credentials are verified the app will redirect to home screen. If you forgot your password you can use Forgot password button to reset it.

The figure A.2, displays the login form along with options to create an account and reset password if needed.

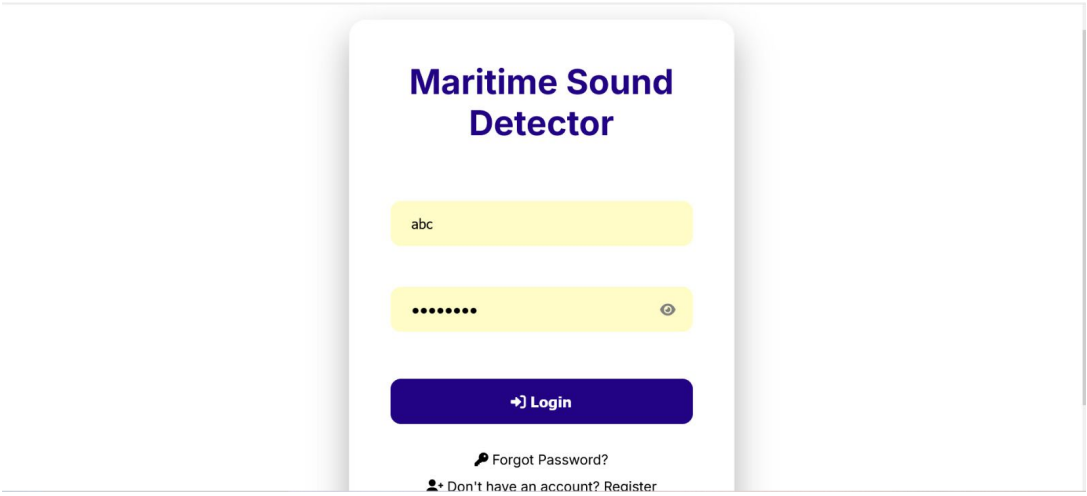


Figure A.2: Login Screen

### A.2.3 Forgot Password:

In case you forgot your password. Click on Forgot Password button in login page. Enter your registered email address. System will send a reset link to your email thus you can update it.

The figure A.3, displays the forgot password form which takes registered email as an input and sends a reset link to the user to update the password.

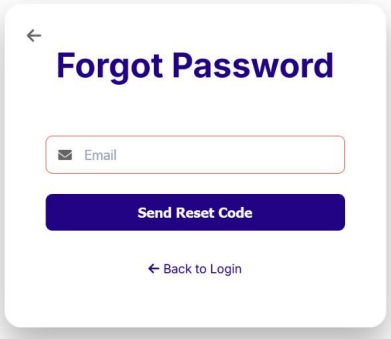


Figure A.3: Forgot Password Screen

## A.3 Workflow

### A.3.1 Home:

It contains a noise button which detects noise and displays animations while capturing noises. A drawer with options such as logs, Manage account, and logout. It also has the option to upload noise.

The figure A.4, represents the home screen that has a detection button which identifies and analyzes the sound once clicked. On the left there is a drawer that contains multiple options such as profile management , viewing logs, uploading noise and logging out.

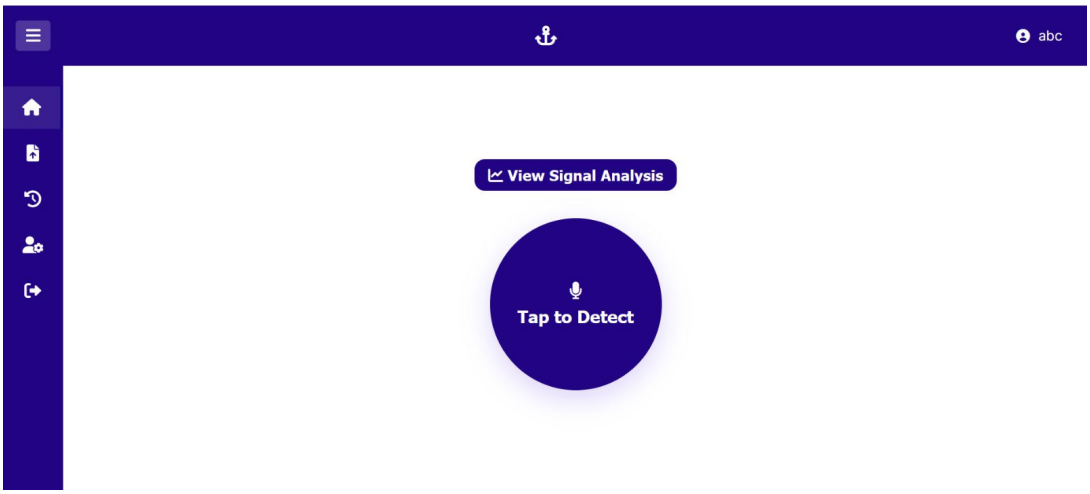


Figure A.4: Home Screen

### A.3.2 Upload Audio:

Open the drawer on left and select upload noise option. Browse the file in your device you want to analyze then click upload. The file must be in ".wav" or ".mp3" format. Click Analyze audio and the system will classify and display result.

The figure A.5, displays the upload audio screen where user can upload audio that contain noise and get classification results.

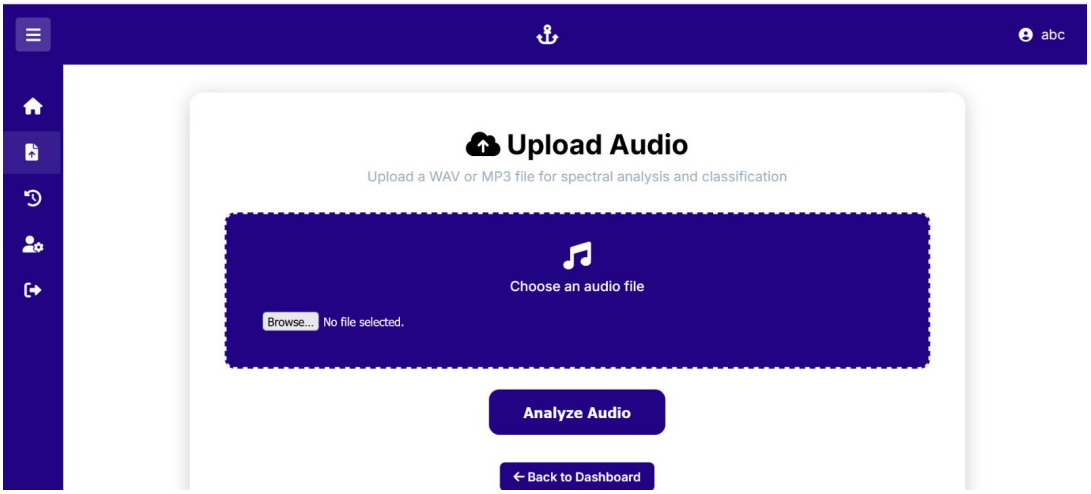


Figure A.5: Audio Upload Screen

### A.3.3 Viewing Logs:

Open the drawer on the left and select View Logs option. All the noises that have been classified along with timestamps will be mentioned.

The figure A.6, displays the view logs screen with multiple entries of detected noises.

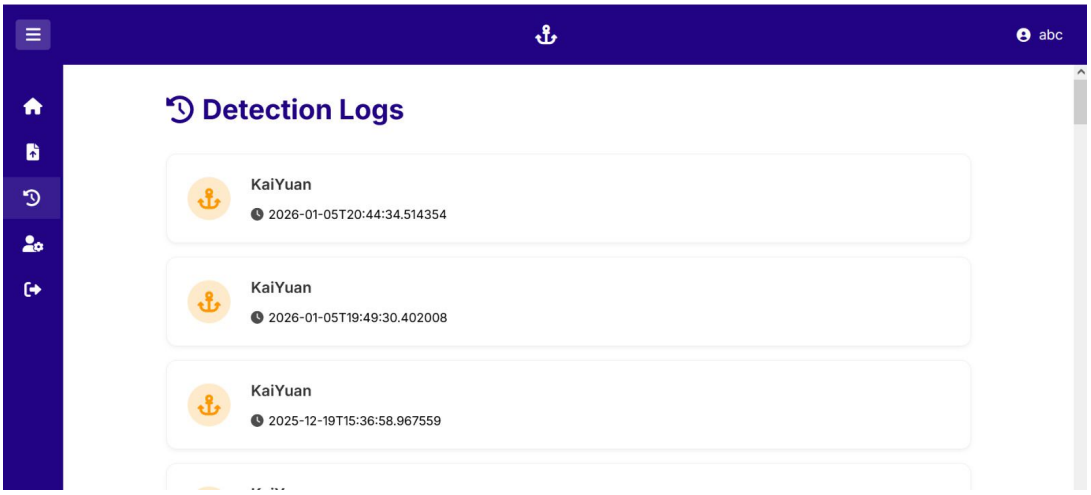


Figure A.6: View Logs Screen

### A.3.4 Profile Management:

Open the drawer on the left and select Manage Account option. The profile information is displayed, therefore you can edit it accordingly. Click on save and information is updated.

The figure A.7, displays the Manage Account screen with user information displayed on it.

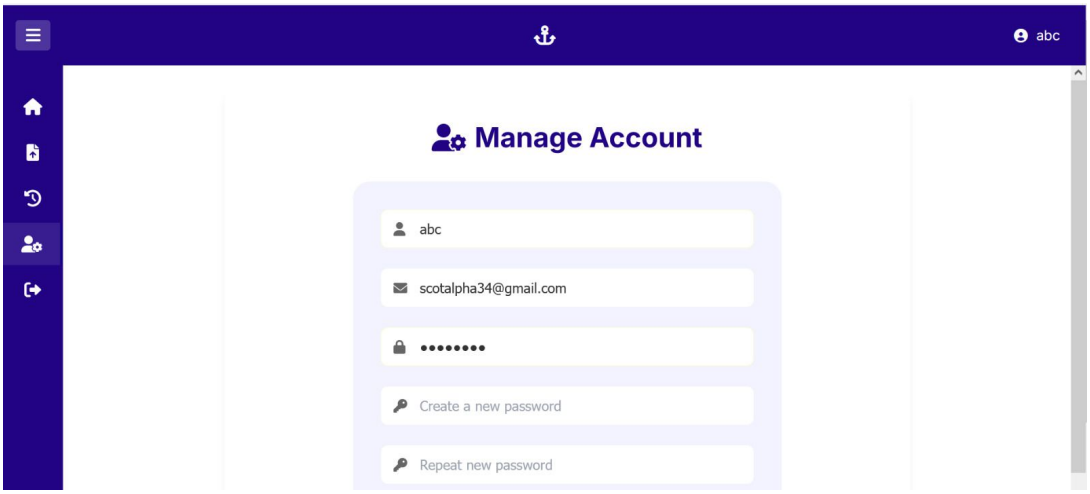


Figure A.7: Manage Account Screen

## A.4 Troubleshooting:

### A.4.1 Microphone:

- **Error**: Microphone Permission denied
- **Solution:** Allow microphone to be accessed by the browser and make sure it's functional.

### A.4.2 Unknown Classification:

- **Error:** System returns "Unknown" in classification.
- **Cause:** The audio can be corrupted or the background noise is of different ship.

## A.5 Support and Contact:

For any technical issues or queries, please contact support at "maritimesoundapp@gmail.com".

# References


[1] Johan A. Bocanegra, Davide Borelli, Tomaso Gaggero, Enrico Rizzuto, and Corrado Schenone. Characterizing onboard noise in ships: Insights from statistical, machine learning and advanced noise index analyses. *Ocean Engineering*, 285:115273, 2023. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2023.115273. URL https://www.sciencedirect.com/science/article/pii/S0029801823016578. Cited on p. 4.

[2] Luis Alfonso Díaz-Secades, Rebeca Bouzón Otero, Yolanda Amado-Sánchez, and Fernando Crestelo Moreno. Noise exposure and mitigation on high-speed craft: Assessing acoustic environment and regulatory compliance. *Journal of Marine Science and Engineering*, 12(12), 2024. ISSN 2077-1312. doi: 10.3390/jmse12122329. URL https://www.mdpi.com/2077-1312/12/12/2329. Cited on p. 7.

[3] Ruobin Gao, Maohan Liang, Heng Dong, Xuewen Luo, and P. N. Suganthan. Underwater acoustic signal denoising algorithms: A survey of the state-of-the-art. *arXiv preprint*, arXiv:2407.13264v1, 2024. URL https://arxiv.org/abs/2407.13264v1. Cited on p. 5.

[4] I Gloza. Identification methods of underwater noise sources generated by small ships. *Acta Physica Polonica A*, 119(6A):961–965, 2011. Cited on p. 5.

[5] Marc-André Guy, Kamal Kesour, Mathis Vulliez, Stéphane Gagnon, and Olivier Robin. Assessment of the effectiveness of ship machinery noise reduction measures using a test platform in a water basin. *Ocean Engineering*, 313, 10 2024. doi: 10.1016/j.oceaneng.2024.119380. Cited on p. 1.

[6] Miguel Hidalgo-Rodriguez, Edmanuel Cruz, Cesar Pinzon-Acosta, Franchesca Gonzalez-Olivardia, and José Carlos Rangel. Magi: A low-cost iot architecture for distributed ais-based vessel monitoring and maritime emissions assessment in panama. *Applied System Innovation*, 8(6), 2025. ISSN 2571-5577. doi: 10.3390/asi8060177. URL https://www.mdpi.com/2571-5577/8/6/177. Cited on p. 6.

[7] Rodrigo F. Javier, Ramis Jaime, Poveda Pedro, Carbajo Jesus, and Segovia Enrique. Analysis of the underwater radiated noise generated by hull vibrations of the ships. *Sensors*, 23(2), 2023. ISSN 1424-8220. doi: 10.3390/s23021035. URL https://www.mdpi.com/1424-8220/23/2/1035. Cited on p. 4.

[8] Feng Liu, Tongsheng Shen, Luo Zailei, Dexin Zhao, and Shaojun Guo. Underwater target recognition using convolutional recurrent neural networks with 3-d mel-spectrogram and data augmentation. *Applied Acoustics*, 178:107989, 07 2021. doi: 10.1016/j.apacoust.2021.107989. Cited on p. 1.

[9] J. M. Riola, J. C. Díaz-Cuadra, and Publio Beltrán. Noise and vibration control program for warship: The new spanish frigate f110. In Vice Admiral Jorge Enrique Carreño Moreno,

Adan Vega Saenz, Luis Carral Couce, and Jymmy Saravia Arenas, editors, *Proceeding of the VI International Ship Design & Naval Engineering Congress (CIDIN) and XXVI Pan-American Congress of Naval Engineering, Maritime Transportation and Port Engineering (COPINAVAL)*, pages 163–174, Cham, 2020. Springer International Publishing. ISBN 978-3-030-35963-8. Cited on p. 2.

[10] Tom Smith and Jake Rigby. Underwater radiated noise from marine vessels: A review of noise reduction methods and technology. *Ocean Engineering*, 266:112863, 12 2022. doi: 10.1016/j.oceaneng.2022.112863. Cited on p. 1.

[11] Yongqiang Song, Qian Chu, Feng Liu, Tao Wang, and Tongsheng Shen. Underwater acoustic signal noise reduction based on a fully convolutional encoder-decoder neural network. *Journal of Ocean University of China*, 22:1487–1496, 11 2023. doi: 10.1007/s11802-023-5458-z. Cited on p. 55.

[12] Imam Sutrisno, Fahmi Umasangadji, Ardiansyah, Nafi Almuzani, Achmad Bashori, Urip Mudjiono, Projek Priyonggo, and Endang Purwanti. Development of an iot-based ship engine performance monitoring system to enhance operational efficiency. *Formosa Journal of Computer and Information Science*, 4:83–92, 03 2025. doi: 10.55927/fjcis.v4i1.14134. Cited on p. 1.

[13] Yan Wang, Hao Zhang, Wei Huang, Manli Zhou, Yong Gao, Yuan An, and Huifeng Jiao. Dwstr: a hybrid framework for ship-radiated noise recognition. *Frontiers in Marine Science*, Volume 11 - 2024, 2024. ISSN 2296-7745. doi: 10.3389/fmars.2024.1334057. URL https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2024.1334057. Cited on p. 1.

[14] Chenhong Yan, Shefeng Yan, Tianyi Yao, Yang Yu, Guang Pan, Lu Liu, Mou Wang, and Jisheng Bai. A lightweight network based on multi-scale asymmetric convolutional neural networks with attention mechanism for ship-radiated noise classification. *Journal of Marine Science and Engineering*, 12(1), 2024. ISSN 2077-1312. doi: 10.3390/jmse12010130. URL https://www.mdpi.com/2077-1312/12/1/130. Cited on p. 2.

[15] Shan Zhu, Guojun Zhang, Daiyue Wu, Li Jia, Yifan Zhang, Yanan Geng, Yan Liu, Weirong Ren, and Wendong Zhang. High signal-to-noise ratio mems noise listener for ship noise detection. *Remote Sensing*, 15(3), 2023. ISSN 2072-4292. URL https://www.mdpi.com/2072-4292/15/3/777. Cited on p. 5.

[16] Ali Zolfagharian, Amin Noshadi, Mohammad Khosravani, and M. Z. Md zain. Unwanted noise and vibration control using finite element analysis and artificial intelligence. *Applied Mathematical Modelling*, 38, 11 2013. doi: 10.1016/j.apm.2013.10.039. Cited on p. 4.