

Sound Sense  
(Smart Visual Alert System for Deaf)

# Sound Sense Smart Visual Alert System for Deaf



## Authors

Muhammad Usman Ghani  
Faheem Haider  
Ahmed Bin Yasin  
Muhammad Arsam Khalid

## Supervisor

Dr. Naima Iltaf

Submitted to Faculty of Department of Computer Software Engineering National University of Sciences and technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering.

Sound Sense  
(Smart Visual Alert System for Deaf)

In the name of Allah, the Most Beneficent, the Most Merciful

## DECLARATION

We affirm that we have carried out the original work presented in this project report titled "Sound Sense – Smart Visual Alert System," submitted to the Computer Software Engineering Department, under the supervision of Dr. Naima Iltaf. We confirm that all sources used in this work have been duly cited, and no part of this project has been plagiarized. Furthermore, we submit this project work as part of the fulfillment of the requirements for the degree of BE Software Engineering.

### Team Members

Muhammad Usman Ghani

Faheem Haider

Ahmed Bin Yasin

Muhammad Arsam Khalid

### Signature

---

---

---

---

### Supervisor

Dr. Naima Iltaf

### Signature

---

## ACKNOWLEDGEMENTS

There is no success without the will of ALLAH Almighty. We are grateful to ALLAH, who has given us guidance, and strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality.

We are also grateful to our parents, family, and well-wishers for their admirable support and critical reviews.

We are extremely grateful to our project supervisor Asst. Professor Dr. Naima Iltaf, for her continuous guidance and motivation throughout the course of our project. Without her help, we would not have been able to accomplish this feat.

We are highly thankful to all the faculty and staff of the Computer Software Department of the Military College of Signals, NUST, for their support and training throughout our coursework. Their guidance helped us to carry out this project.

In the end, we would like to acknowledge the efforts of all our friends, colleagues, and well-wishers whose prayers and support helped us in achieving this milestone.

## DEDICATION

We would like to dedicate this thesis to our parents, whose unwavering support, guidance, and encouragement have been the foundation of our academic and personal success. Their love and sacrifice have inspired us to pursue our dreams and never give up on our aspirations. We are grateful for their constant presence in our life, and We hope that this thesis will make them proud. We also dedicate this work to our friends and colleagues, who have provided invaluable insights, feedback, and motivation throughout the research process. Finally, we want to express our deep appreciation and gratitude to our supervisor, Dr. Naima Iltaf, for their expert guidance, mentorship, and unwavering support throughout this journey. Their insights, feedback, and encouragement have been critical to the success of this thesis, and I am deeply grateful for their contribution.

## ABSTRACT

Deaf people face numerous challenges in their daily lives due to their inability to hear sounds in the environment. To address this issue, we have developed a smart visual alert system for deaf people that can detect and classify sounds in real-time and provide visual alerts using LED lights. The system is implemented on an edge computing device (a Raspberry Pi) to take the processing closer to data gathering in order to ensure fast and efficient processing of sound data and classification. The input sound is pre-processed to generate spectrograms which are then classified using a Convolutional Neural Network into several categories, including "Baby Cry", "Doorbell", "Talking", etc. The LED lights are controlled using GPIO pins on the Raspberry Pi, to provide different patterns or colors to indicate different types of sounds. A mobile app is also developed to allow users to view the history of events, adjust configurations, and access other assistive features that include Reminder, Speech-to-text, etc. The system has the potential to improve the quality of life for deaf people by providing fast and reliable visual alerts for important sounds at their homes.

# Table of Contents

## Contents

<b>Table of Contents .....</b>	<b>vii</b>
<b>1. Introduction .....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Project Scope.....	1
1.3 Overview .....	1
1.4 Objectives: .....	2
1.5 Deliverables: .....	2
1.6 References .....	3
<b>2. Literature Review .....</b>	<b>3</b>
<b>3. Problem Definition .....</b>	<b>4</b>
<b>4. Solution Statement .....</b>	<b>5</b>
<b>5. Detailed Design and Architecture .....</b>	<b>7</b>
5.1 Architectural Design .....	7
5.1.1 System Block Diagram .....	8
5.1.2 System Block Diagram (Machine Learning Model).....	8
5.2 Requirements Specification.....	9
5.2.1 External Interface Requirements .....	9
5.2.2 Functional Requirements.....	11
5.2.2 Non-Functional Requirements: .....	16
5.3 Use Cases .....	16
5.1.3.1 Account Login (App’s perspective) .....	16
5.1.3.2 Account Verification (Servers’ perspective).....	18
5.1.3.3 Account Creation (Server’s perspective).....	19
5.1.3.4 Use Case: App Reminders (App’s perspective) .....	20
5.1.3.5 Use Case: Sound Classification.....	24
5.1.3.6 Use Case: Server-End.....	27
5.1.3.7 Use Case: Speech-to-text .....	30
5.4 Class Diagram .....	32
5.5 Detail System Design.....	32
5.5.1 Sequence Diagram.....	33
5.5.2 Dynamic View (Activity Diagram).....	38
5.5.3 Logical View (State Diagram).....	42
5.5.4 Design Rationale.....	45
5.5.5 Data Description .....	46
5.5.6 Component Diagram .....	48
<b>6. Implementation and Testing.....</b>	<b>48</b>
6.1 Environment Needs .....	57
6.1.1 Hardware.....	57
6.1.2 Software.....	58
6.2 User Interface Design.....	59
6.2.1 Desktop Application .....	59
6.2.2 Mobile Application .....	61
<b>7. Results and Discussion .....</b>	<b>65</b>

Sound Sense  
(Smart Visual Alert System for Deaf)

7.1 Overview .....	65
7.2 Objectives achieved. ....	65
<b>8. Future work .....</b>	<b>66</b>



# 1. Introduction

## 1.1 Purpose

This document describes the architecture and system design of project Smart Visual Alert System. This document serves as a comprehensive guide for developers and a software validation document for prospective clients. It outlines the design of features and project requirements and includes detailed descriptions of use cases, class inter-relationships, sequence diagrams, and flow charts.

## 1.2 Project Scope

The scope of this project is limited to sound content 'only'. Meaning that this project would not utilize any associated video from the environment for any purpose at all. In this project, we will be considering sound events for instance the baby cry, doorbell, the sound of machines, talking, etc., and would detect and classify those events using an edge computing device and furthermore display a visual alert for that event via different colors of light.

The primary goal of the project is to assist a deaf person in his/her home by giving visual alerts of occurring sound events that they cannot hear.

The Project also offers some other assistive features for deaf people that includes setting up Reminders of a particular time to get notified by visual alerts on that time furthermore it also provides a feature for recording the speech of a person and getting a textual script of it on the application screen.

## 1.3 Overview

This document is about the detailed architectural design and implementation of Smart Visual Alert System. The document is segmented into different sections for convenience.

- The document is introduced and given an executive summary in Section 1.
- Section 2 provides a thorough discussion of the system along with numerous charts and pictures. This section contains all of the system's architectural information.
- Section 3 describes all the modules and components of the system in detail one by one.
- Section 4 compares this product to various other similar products available in market.
- Section 5 throws light on the design decisions and tradeoffs.

This document is intended for developers, testers, users, documentation writers, project clients, project supervisors, and project evaluators.

## 1.4 Objectives:

The objectives of developing such a system are to provide visual services and smart notification platform with reasonable accuracy and user-friendly interface. Following are the main objectives:

1. To develop an IoT device that captures environmental sounds and provides visual feedback through LED lights to assist deaf individuals.
2. To implement edge computing on the Raspberry Pi for real-time sound classification and visual response generation.
3. To design a mobile application that receives the classified sound from the device and generates notifications to alert the user.
4. To use socket programming to establish communication between the device and the mobile application.
5. To create a cost-effective solution for deaf individuals that can be easily deployed and used in various environments.
6. To improve the quality of life for deaf individuals by providing them with a helpful and accessible environment.

## 1.5 Deliverables:

Project plan	A document outlining the project scope, timeline, and resources required
Raspberry Pi setup	A configured Raspberry Pi with a microphone and LED lights connected
Edge computing software	A program that can classify environmental sounds and trigger LED light responses on the Raspberry Pi
Mobile application design	A design document for the user interface and functionality of the mobile application
Socket programming implementation	Code that establishes communication between the device and the mobile application
Sound classification model	A trained machine learning model that can accurately classify environmental sounds on the Raspberry Pi
Mobile application development	A functional mobile application that receives sound classifications and generates notifications
Testing plan	A document outlining the testing strategy and criteria for the project

User manual	A document providing instructions for setting up and using the device and mobile application
-------------	--

## 1.6 References

1. Approved SRS Document Version 1.2 For Smart Visual Alert System
2. Approved SDS Document Version 1.1 For Smart Visual Alert System
3. [http://en.wikipedia.org/wiki/Sequence\\_diagram](http://en.wikipedia.org/wiki/Sequence_diagram)

## 2. Literature Review

### **TensorFlow Audio:**

TensorFlow Audio is a library for audio processing in TensorFlow that provides a variety of tools for working with audio data, including tools for loading and preprocessing audio files, building audio models, and performing audio analysis. The library has been used in a variety of applications, including speech recognition, music classification, and environmental sound analysis.

### **Mobile Net:**

Mobile Net on the other hand, is a deep learning model architecture that was designed to be fast and lightweight, making it well-suited for use on edge devices with limited processing power and memory. The architecture uses depth wise separable convolutions to reduce the number of parameters and computation required, while maintaining a high level of accuracy on a variety of tasks.

Several studies have explored the use of MobileNet for audio classification tasks using TensorFlow Audio. For example, a study by Choi et al. (2019) used MobileNet for environmental sound classification, achieving high accuracy on a dataset of urban sounds. Similarly, a study by Kao et al. (2020) used MobileNet for speaker identification, achieving high accuracy while reducing the size and computational requirements of the model.

**Simple Audio Recognition (SAR) and MobileNet** are two widely used models for audio classification tasks. While SAR is designed specifically for audio classification, MobileNet is a more general purpose model that can be used for image classification, object detection, and other tasks in addition to audio classification.

One key difference between SAR and MobileNet is the architecture of the models. SAR uses a Convolutional Neural Network (CNN) with a sequence of convolutional, pooling, and fully connected layers. On the other hand, MobileNet uses a streamlined architecture that reduces the number of parameters and computational complexity by using depthwise separable convolutions. This makes MobileNet faster and more efficient to run, but may result in lower accuracy compared to SAR for some audio classification tasks.

Another difference between SAR and MobileNet is the type of input data they are designed to handle. SAR is

specifically designed for keyword spotting, where the goal is to identify a small set of target words or phrases from a larger audio stream. In contrast, MobileNet is more general purpose and can be trained on a wide range of audio data for classification tasks such as music genre classification, speaker identification, or sound event detection.

**Our System encapsulated in the literature review:**

Our project involves using the Simple Audio Recognition (SAR) model for audio classification tasks, specifically for identifying environmental sounds in real-time and providing visual feedback for deaf individuals. Since the audio files have been transformed into spectrogram images, a simple convolutional neural network (CNN) will be utilized for the model. The SAR model is integrated into our IoT device which collects environmental sounds using a microphone, classifies them in real-time using SAR, and generates visual feedback through LED lights attached to the device. The classified sound is also sent to a server for further processing and sent to a mobile application as a notification. The use of SAR ensures high accuracy in identifying target sounds while keeping the computational complexity and memory requirements low, making it suitable for deployment in resource-constrained IoT devices.

### **3. Problem Definition**

Deaf individuals face numerous challenges in their daily lives due to their limited ability to perceive and interpret auditory information. While previous technologies have attempted to address this issue by providing visual aids, such as flashing lights or text-based notifications, these solutions have been limited in their effectiveness and accessibility.

One of the main challenges faced by deaf individuals is the need to constantly monitor their environment for potential hazards or events. For example, a deaf individual may not be able to hear a fire alarm or an approaching vehicle, which can be extremely dangerous. While some existing technologies attempt to address this issue by providing visual cues, such as flashing lights or vibrating alerts, these solutions can be limited in their effectiveness. For example, flashing lights may not be visible in bright sunlight, and vibrating alerts may not be noticed if the individual is not in physical contact with the device.

Another challenge faced by deaf individuals is the need to communicate effectively with others. While sign language and text-based communication can be effective in many situations, they may not always be practical or feasible. For example, a deaf individual may struggle to communicate with a hearing individual in a noisy environment or may be unable to communicate with someone who does not know sign language. Existing technologies that attempt to address this issue, such as speech-to-text software or live captioning services, can

be expensive and require specialized hardware or software.

In addition to these challenges, many existing assistive technologies for deaf individuals are expensive, bulky, and rely on cloud computing or high-end hardware, making them difficult for deaf individuals to access or use. For example, some existing technologies require specialized hardware or software, such as hearing aids or cochlear implants, which can be expensive and difficult to maintain. Other technologies rely on cloud computing or high-end hardware, such as deep learning models for sound recognition, which can be expensive and difficult to deploy in real-world situations.

As a result of these challenges, there is a need for a cost-effective and accurate solution that can provide deaf individuals with reliable visual feedback based on their environment. One promising approach is the use of edge computing and deep learning models for sound recognition, such as the MobileNet model in TensorFlow Audio. By capturing environmental sounds using a microphone and processing them in real-time on a Raspberry Pi, it is possible to provide deaf individuals with reliable visual feedback based on their environment, without the need for expensive hardware or cloud computing.

However, there are still challenges to be overcome in developing such a solution. For example, selecting and preprocessing training data for sound recognition models can be difficult, particularly when dealing with complex and varied real-world sounds. Additionally, tuning model parameters and hyperparameters to ensure optimal accuracy and performance can be time-consuming and challenging. Nevertheless, ongoing research and development in this area are essential to continue advancing the state of the art and improving the accuracy and effectiveness of assistive technologies for deaf individuals.

## **4. Solution Statement**

The solution statement for this project is to provide a comprehensive and accessible solution for deaf individuals that uses state-of-the-art edge computing and deep learning models for sound recognition to provide reliable visual feedback based on their environment. By leveraging the power of the Raspberry Pi and the TensorFlow Audio library, we aim to provide accurate and timely visual feedback to deaf individuals, which will improve their safety and quality of life.

The first step in our solution is to gather sound from the environment using a microphone attached to the IOT device. The sound is then processed using the TensorFlow Audio library, which is a powerful and flexible library for audio signal processing and deep learning.

## Sound Sense (Smart Visual Alert System for Deaf)

The waveforms received from the environment are represented in the time domain. Afterwards, we converted the waveforms from the time-domain signals to the time-frequency-domain signals by performing the short-time Fourier transform (STFT), which converted the waveforms to spectrograms - 2D images depicting the signal's frequency content over time. The neural network was trained using the spectrogram images that we provided as input.

When using a Fourier transform (`tf.signal.fft`), the time information is lost even though the signal is converted to its component frequencies. On the other hand, STFT (`tf.signal.stft`) divides the signal into time windows, applies a Fourier transform to each window, which retains some time information, and produces a 2D tensor as output. You can run standard convolutions on Create a utility function for converting waveforms to spectrograms:

- The waveforms need to be of the same length, so that when you convert them to spectrograms, the results have similar dimensions. You can accomplish this task by using `tf.zeros` to pad the audio clips that have a duration of less than one second with zeros. (using [`tf.zeros`](#)).
- When utilizing `tf.signal.stft`, select the `frame_length` and `frame_step` parameters in such a way that the spectrogram generated appears nearly square in shape.
- An array of complex numbers representing both magnitude and phase is generated by the STFT. Nevertheless, for this tutorial, only the magnitude component will be utilized. For the model, we used a simple convolutional neural network (CNN), since we had transformed the audio files into spectrogram images.

The Keras preprocessing layers that will be utilized in our `tf.keras.Sequential` model are as follows:

- `tf.keras.layers.Resizing`: to reduce the input's resolution and expedite the training process of the model.
- [`tf.keras.layers.Normalization`](#): to normalize each pixel in the image based on its mean and standard deviation.

To obtain aggregate statistics such as the mean and standard deviation, the `adapt` method of the `Normalization` layer must be called on the training data. This model has been trained on dataset gathered partially from ECS-50 dataset and partially collected by the group manually.

Once the sound has been classified, the Raspberry Pi is able to generate visual feedback using LED lights. For example, a loud noise such as a car horn might trigger a flashing red light, while a doorbell might trigger a

flashing green light.

Simultaneously, the classified label is sent to the server using socket programming. The server is responsible for storing the data and sending notifications to the mobile application. The mobile application can be installed on any smartphone or tablet and provides a simple and intuitive interface for deaf individuals to monitor their environment.

The mobile application receives the classified label from the server and generates a notification based on the type of sound detected. For example, if the sound is classified as a smoke detector, the mobile application might generate a notification that says "Smoke detector detected in the kitchen". The mobile application can also provide additional information about the sound, such as its intensity or duration.

One of the key benefits of our solution is that it is cost-effective and accessible. By using edge computing and a lightweight deep learning model, we are able to process audio signals in real-time without the need for expensive hardware or cloud computing. Additionally, our solution is customizable and scalable, allowing for future expansion and integration with other assistive technologies.

In summary, our solution provides a comprehensive and accessible solution for deaf individuals that uses state-of-the-art edge computing and deep learning models for sound recognition to provide reliable visual feedback based on their environment. By leveraging the power of the Raspberry Pi and the TensorFlow Audio library, we are able to provide accurate and timely visual feedback to deaf individuals, which will improve their safety and quality of life.

## **5. Detailed Design and Architecture**

This section covers the overall architectural description of Smart Visual Alert System. It encompasses the high-level and low-level descriptions of the project including block diagrams of the application and the deep learning model. Moreover, a complete object-oriented description includes class diagrams, sequence diagrams, and others. Finally, the rationale for the design pattern is provided.

### **5.1 Architectural Design**

The architectural design has been divided into two portions;

- One covers the application architecture

Sound Sense  
(Smart Visual Alert System for Deaf)

- Other depicts the architecture of the deep learning model used.

### 5.1.1 System Block Diagram

The diagram provides an overview of the application by displaying its various modules and the connections and data flow among them at a higher level:

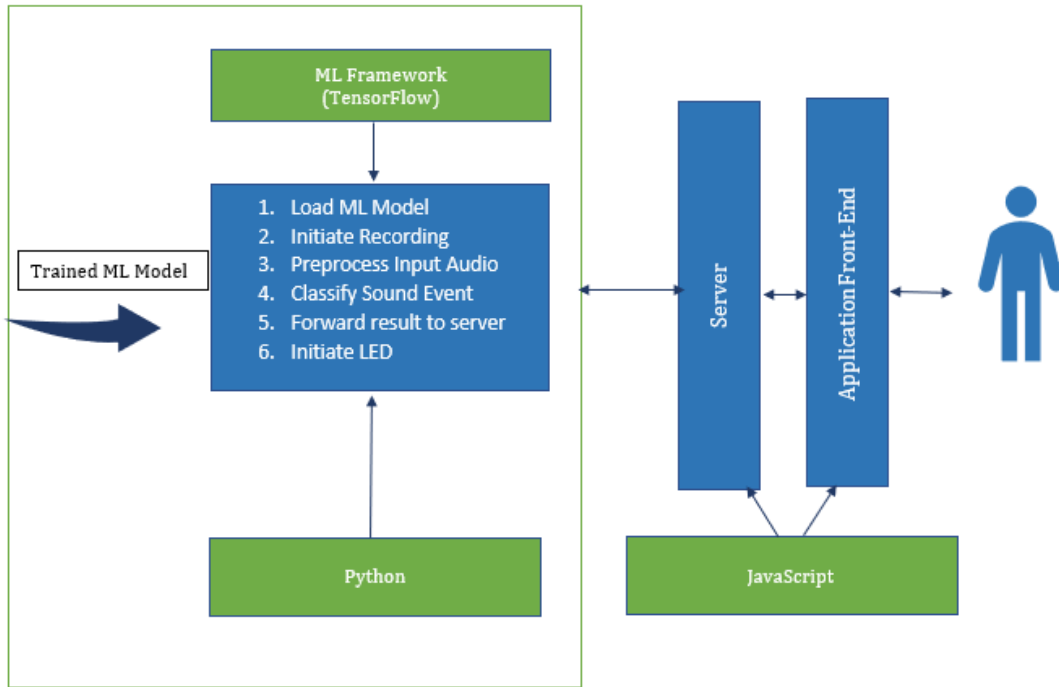
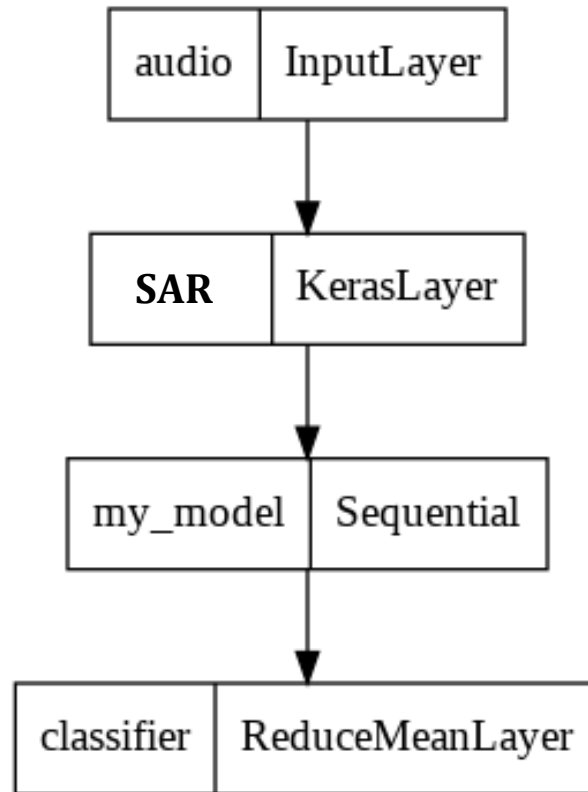


Fig 3.1.1.1 (System Block Diagram)

All the modules mentioned above are incorporated in this system block diagram. Firstly, we have the deep learning model which we have trained on dataset. This model has been trained on dataset gathered partially from ECS-50 dataset and partially collected by the group manually. We classify the audio in the edge computing device and pass the results onto the server. Secondly, we would have a server in Nodejs that would receive results from edge computing device and further log it into database and display notification on screen and handle all business logic implementation of other features. Lastly, we have the front-end of the application through which the user will interact with the system.

### 5.1.2 System Block Diagram (Machine Learning Model)





A spectrogram of input sound would be generated which will then be passed into a CNN for classification Layer which is a pre trained deep learning model trained on YouTube Audio Dataset afterward we would pass it into a sequential layer, the number of neurons in the sequential layer would be set as per the classes available in the secondary data set, Lastly, we have a reduce mean Layer It takes as input a list of tensors, all of the same shape, and returns a single tensor. The model will classify the sound input into 4 classes that includes “crying babies”, “doorbells”, “machines”,” Talking”.

## 5.2 Requirements Specification

### 5.2.1 External Interface Requirements

#### 5.2.1.1 User Interfaces

1. Login/Register: The screen that would allow a user to log in to the app or create a new account.
2. Main Screen: the screen with the user info dialogue box at the top and icons of the main features of the app below it.
3. Connected devices: the page that would lists all connected devices.
4. Alarm page: the page that would allow users to add alarms and their configuration.
5. Configuration page: the page to configure device settings.
6. Theme settings: the page that would allow the user to change the theme of the app.

Sound Sense  
(Smart Visual Alert System for Deaf)

7. Sidebar: the navigation sidebar that would contain options to view logs, settings, and history.
8. Settings page: the page that contains settings of notification and alarms.
9. History page: the page that would show past activates.
10. Logs page: the page that would show all messages related to debugging and errors.

### 5.2.1.2 *Hardware Interfaces*

Edge computing refers to a decentralized computing approach that brings data storage and computation closer to the data sources. This approach is anticipated to enhance response times and reduce bandwidth usage. Rather than a particular technology, it is a framework or architecture. It is a topology- and location-sensitive form of distributed computing.

Our Project Is an application of edge computing that is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers. Being in close proximity to data at its source can provide significant advantages for businesses, such as faster insights, enhanced response times, and greater bandwidth availability.

In the past, the promise of cloud and AI was to automate and speed innovation by driving actionable insight from data. However, the network and infrastructure capabilities have been surpassed by the unprecedented scale and complexity of data generated by interconnected devices.

Transmitting all of the data generated by these devices to a centralized data center or the cloud can result in issues such as bandwidth limitations and increased latency. Edge computing offers a more efficient alternative; data is processed and analyzed closer to the point where it's created

The Central Hub device of our Project is the edge device with the mic installed inside it. This device would listen to sounds entering the mic and would run a model that would recognize the sounds and send the result to the backend server. Since the data is not transmitted over a network to a data center or cloud for processing, there is a substantial reduction in latency.

### 5.2.1.3 *Software Interfaces*

The central hub would run the Raspbian Pi OS, A free operating system based on Debian optimized for the Raspberry Pi hardware. the mobile application would be required to be install on a smart phone running android 4.2+. The Server Program and the Models for the backend would run on the NodeJS runtime Environment. The backend server can be set up on any computer or even on a server online but since the performance and the hardware required for the backend program are present inside the central hub it would also host the backend.

#### 5.2.1.4 *Communications Interfaces*

Universal Serial Bus (USB) is an industry standard that establishes specifications for cables, connectors, and protocols for connection, communication, and power supply (interfacing) between computers, peripherals, and other computers. Our device would use Universal Serial Bus (USB) to communicate between the mic and the central hub.

A local area network (LAN) refers to a computer network that links computers within a confined geographical area, such as a school, office building, laboratory, university campus, or residence. Wi-Fi is a group of wireless network protocols that utilize the IEEE 802.11 family of standards, and are typically employed for local area networking and internet access, This technology facilitates the exchange of data between digital devices in close proximity by utilizing radio waves. The central hub (the Backend) would communicate with the Mobile Application (the frontend) through a local area network hosted on either a router or the central hub supporting both Wi-Fi and LAN.

### 5.2.2 Functional Requirements

#### 5.2.1.1 *Account Login*

<b>Use Case Name</b>	Account Login
<b>XRef</b>	Section 4.1 Account Login
<b>Trigger</b>	The User launches the App
<b>Precondition</b>	The User has already installed the required App
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The User opens the App.</li> <li>2. The system displays the login screen to the user.</li> <li>3. The User Enter his Name and Password if user already has an account.</li> <li>4. The system displays the Register screen to the user..</li> <li>5. The User Enter his Name and new Password for the account.</li> <li>6. The User then clicks on Login button.</li> <li>7. The System proceeds with the login procedure</li> </ol>
<b>Alternative Paths</b>	<p>In step 3, if the User selects to create new account, then proceed to the Register screen</p> <p>In step 3, if user enters wrong username or password then.</p> <ol style="list-style-type: none"> <li>1. The App displays the Wrong credentials dialog.</li> <li>2. The App Returns to step 2.</li> </ol>
<b>Postcondition</b>	The app receives the authentication token and displays the main screen
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The auth token is stored for future authorization process

#### 5.2.1.2 *Account Verification*

<b>Use Case Name</b>	Account Verification
<b>XRef</b>	Section 4.2 Account Verification
<b>Trigger</b>	The User has clicked the login button
<b>Precondition</b>	The User has provided the login credentials
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system scans the username and password.</li> <li>2. The system attempts to connect with the server.</li> <li>3. The system encrypts the password and send username and encrypted password to server.</li> </ol>

Sound Sense  
(Smart Visual Alert System for Deaf)

	<ol style="list-style-type: none"> <li>4. The Server receives the Username and Password and attempt to verify it using verification token from database.</li> <li>5. The Server return an Authorization token if verification is successful.</li> <li>6. The System receives the auth token and store it.</li> <li>7. The System proceeds with the login procedure.</li> </ol>
<b>Alternative Paths</b>	<p>In step 3, if the communication with the server failed then display the no response dialog</p> <p>In step 5, if server did not find the user in database, then return no user found message.</p> <ol style="list-style-type: none"> <li>1. Server send the no user found message</li> <li>2. The app displays no user found dialog</li> </ol>
<b>Postcondition</b>	The app receives the authentication token and displays the main screen
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The auth token is stored for future authorization process

5.2.1.3 Account Creation

<b>Use Case Name</b>	Account Creation
<b>XRef</b>	Section 4.3Account Creation
<b>Trigger</b>	The User has clicked the Register option
<b>Precondition</b>	The User has provided the Registration credentials
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system matches the entered Password and reentered Password</li> <li>2. The system scans the username and password.</li> <li>3. The system attempts to connect with the server.</li> <li>4. The system encrypts the password and send username and encrypted password to server.</li> <li>5. The Server receives the Username and Password and verify if the user already does not exist.</li> <li>6. The Server generate new Verification token and stores it in the database.</li> <li>7. The Server return an Authorization token if user creation is successful.</li> <li>8. The System receives the auth token and store it .</li> <li>9. The System proceeds with the login procedure.</li> </ol>
<b>Alternative Paths</b>	<p>In step 3, if the communication with the server failed then display the no response dialog</p> <p>In step 5, if server find a user already in database, then return user already exist message.</p> <ol style="list-style-type: none"> <li>1. Server send the user already exist message</li> <li>2. The app displays user already exist dialog</li> </ol>
<b>Postcondition</b>	The App receives the authentication token and displays the main screen
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The auth token is stored for future authorization process

5.2.1.4 Add Reminder

<b>Use Case Name</b>	Add Reminder
<b>XRef</b>	Section 4.4.2.1 Add Reminder
<b>Trigger</b>	The User selects the reminders section in the app.

Sound Sense  
(Smart Visual Alert System for Deaf)

<b>Precondition</b>	The User has already logged in the app
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The User selects the Add reminder option</li> <li>2. The system asks for the time of reminder trigger, reminder Description, and Alarm settings</li> <li>3. The user selects the time, enters the Reminder description and configure the alarm settings.</li> <li>4. The App creates a new Event and push it to Alarm Engine.</li> <li>5. The Alarm Engine checks through all events.</li> <li>6. The Alarm Engine waits for the event trigger.</li> <li>7. When the event is triggered the alarm engine Interrupts the app</li> <li>8. The App respond to the interrupt and display notifications and trigger the alarm</li> </ol>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog
<b>Postcondition</b>	The App sets a new reminder
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The reminders are stored in app storage and alarm engine wait for its trigger

5.2.1.5 Update Reminder

<b>Use Case Name</b>	Update Reminder
<b>XRef</b>	Section 4.4.2.2 Update Reminder
<b>Trigger</b>	The User selects the reminders section in the app.
<b>Precondition</b>	The User has already logged in the app
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The User selects the View reminder option.</li> <li>2. The App displays all the added reminders.</li> <li>3. The User selects a reminder.</li> <li>4. The App asks for the time of reminder trigger, reminder Description, and Alarm settings</li> <li>5. The user selects the time, enters the Reminder description and configure the alarm settings.</li> <li>6. The App creates a Update Event and push it to Alarm Engine.</li> </ol>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog
<b>Postcondition</b>	The App updates the selected reminder
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The reminders are stored in app storage and alarm engine wait for its trigger

5.2.1.6 Check status Reminder

<b>Use Case Name</b>	Check status Reminder
<b>XRef</b>	Section 4.4.2.3 Check status Reminder
<b>Trigger</b>	The User selects the reminders section in the app.
<b>Precondition</b>	The User has already logged in the app
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The User selects the view reminder option.</li> <li>2. The App displays all the added reminders.</li> </ol>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog

Sound Sense  
(Smart Visual Alert System for Deaf)

<b>Postcondition</b>	The App displays all set reminders
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The reminders are stored in app storage and alarm engine wait for its trigger

5.2.1.7 Delete Reminder

<b>Use Case Name</b>	Delete Reminder
<b>XRef</b>	Section 4.4.2.4 Delete Reminder
<b>Trigger</b>	The User selects the reminders section in the app.
<b>Precondition</b>	The User has already logged in the app
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>3. The User selects the View reminder option.</li> <li>4. The App displays all the added reminders.</li> <li>5. The User selects a reminder.</li> <li>6. The user clicks on delete button</li> <li>7. The App send the alarm engine a delete request with the event.</li> <li>8. The alarm engine deletes the event and return true</li> </ol>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog
<b>Postcondition</b>	The App removes the selected reminder
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The reminders are stored in app storage and alarm engine wait for its trigger

5.2.1.8 Trigger Reminder on or off

<b>Use Case Name</b>	Trigger Reminder On or Off
<b>XRef</b>	Section 4.4.2.5 Trigger Reminder On or Off
<b>Trigger</b>	The User selects the reminders section in the app.
<b>Precondition</b>	The User has already logged in the app
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>9. The User selects the View reminder option.</li> <li>10. The App displays all the added reminders.</li> <li>11. The User selects a reminder.</li> <li>12. The User toggle the Alarm on/off</li> <li>13. The App send the alarm engine a trigger request with the option and the event.</li> <li>14. The alarm engine turns on or off the event and return true</li> </ol>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog
<b>Postcondition</b>	The App Enable/disable the selected reminder
<b>Exception Paths</b>	The user may close the app
<b>Other</b>	The reminders are stored in app storage and alarm engine wait for its trigger

6.1.5.1 Receive Audio Use Case

<b>Use Case Name</b>	Receive Audio
<b>XRef</b>	Section 4.5.2.1 Receive Audio
<b>Trigger</b>	The User has started the application and the app detects one of the classified sounds.
<b>Precondition</b>	The User has already logged in to the app and is connected to the server.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system will activate the mic to receive sound input</li> <li>2. The system would store the sound as a mono sound channel in wav files of 3-5 sec duration.</li> </ol>

Sound Sense  
(Smart Visual Alert System for Deaf)

	<p>3. The system would apply a function on wav file to know the presence of audio in the file over a certain loudness level.</p> <p>4. In case there is no sound present in the file or is not up to the defined threshold it would be discarded, and no further step would be taken.</p> <p>5. The system would convert the sound to a tensor using a coded function which could be sent as input to the Ai algorithm for classification.</p>
<b>Alternative Paths</b>	if the communication with the server failed then display the no response dialog
<b>Postcondition</b>	The App then sets wav file ready for classification for audio
<b>Exception Paths</b>	The Reader may close the app
<b>Other</b>	The wav file waits for the other wav files to be processed.

6.1.5.2 Classify Audio:

<b>Use Case Name</b>	Classify Audio
<b>XRef</b>	Section 4.5.2.2 Classify Audio
<b>Trigger</b>	The application receives input wav file to classify.
<b>Precondition</b>	The User has already logged in to the app and is connected to the server.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system would send the preprocessed audio file to the Ai algorithm</li> <li>2. The Ai algorithm would be a neural network developed using transferred learning techniques embedding the YAMnet model.</li> <li>3. The Ai model will classify the sound into respective sound events.</li> <li>4. The output would then trigger the LED light to visually alert the user.</li> </ol>
<b>Alternative Paths</b>	If the communication with the server failed, then display the no response dialog
<b>Postcondition</b>	The App then displays classified sounds to the user through LED lights.
<b>Exception Paths</b>	The Reader may close the app
<b>Other</b>	The LED lights will be a trigger to visually alert the user.

6.1.5.3 Display Results:

<b>Use Case Name</b>	Display Results
<b>XRef</b>	Section 4.5.2.3 Display Results
<b>Trigger</b>	The application receives classified results from the AI model
<b>Precondition</b>	The User has already logged in to the app and is connected to the server.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system would check for the color affiliated with the classified output.</li> <li>2. The system would turn the LED light on with that respective color for a defined interval of time.</li> </ol>
<b>Alternative Paths</b>	If the communication with the server failed, then display the no response dialog
<b>Postcondition</b>	The App then displays classified sounds to the user through LED lights.
<b>Exception Paths</b>	The Reader may close the app
<b>Other</b>	The LED lights will be a trigger to visually alert the user.

6.1.5.4 Speech-to-text Converter:

Sound Sense  
(Smart Visual Alert System for Deaf)

<b>Use Case Name</b>	Speech-to-Text Converter
<b>XRef</b>	Section 4.6 Speech-to-Text Converter
<b>Trigger</b>	The application receives classified results from the AI model
<b>Precondition</b>	The User has already logged in to the app and is connected to the server.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The User selects the Speech-to-Text option.</li> <li>2. The App collects the audio input.</li> <li>3. The App then implies Google API. The API then converts the audio to text.</li> <li>4. The App then displayed the converted text to the user interface.</li> </ol>
<b>Alternative Paths</b>	If the communication with the server failed, then display the no response dialog
<b>Postcondition</b>	The converted text will be then displayed on the interface.
<b>Exception Paths</b>	The User may close the app
<b>Other</b>	The User will be able to read the text to understand the conversation

## 5.2.2 Non-Functional Requirements:

### 5.2.3.1 Performance Requirements

The proposed scheme for app is to make the sound detection operations light weight and faster. The sound detection should be better and more accurate.

#### 5.2.3.1 Safety Requirements

The system should not allow any unauthorized user to view logs and history.

#### 5.2.3.3 Security Requirements

1. Encryption of login credentials would be provided.
2. The saved logs and history data would be protected.
3. Authentication tokens would be used up to date.

#### 5.2.3.4.1 Software Quality Attributes

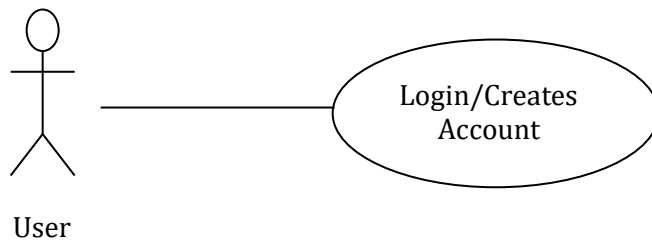
- The system should be running perfectly with all the features mentioned above.
- It would not malfunction or crash while running.
- It would be tested.
- The system would be available to user all the time.

## 5.3 Use Cases

### 5.1.3.1 Account Login (App's perspective)

**Diagram:**





### **Brief Description**

The User accesses the App, and creates a new account or login to an existing one

### **Initial Step-By-Step Description**

This use case requires certain actions to be taken before it can be initiated. The User has already installed the required App.

1. The User opens the App.
2. The App displays the login screen to the user.
3. The User Enter his Name and Password if user already has an account.
4. Else the User clicks on Register button.
5. The App displays the Register screen to the user..
6. The User Enter his Name and new Password for the account.
7. The User then clicks on Login button.
8. The App proceeds with the login procedure

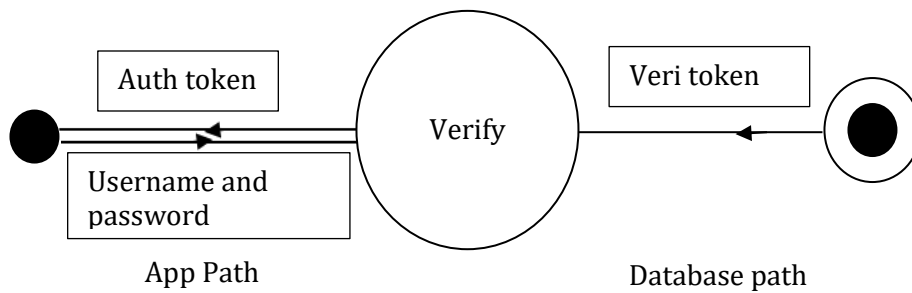
Any new user would be able to create a new account which would provide him the ability to start classifying the sounds around him. Any user can access the website from which they can register. The registered data would be stored in database and login credentials would be provided to the user.

#### *5.1.3.1.1 Functional Requirements*

Any user can make the request of making the account. The request would be inspected by the system to check for its validity of the request. In this feature, the data would be checked whether the requirements are fulfilled. The register button would send the data to the database, and the user would be able to log in with the credentials specified in the registration section.

### 5.1.3.2 Account Verification (Servers' perspective)

**Diagram:**



**Figure 1 – User verification Process**

**Brief Description**

The App sends the login information, and the server returns the authorization token

**Initial Step-By-Step Description**

This use case requires certain actions to be taken before it can be initiated. The User has already installed the required App.

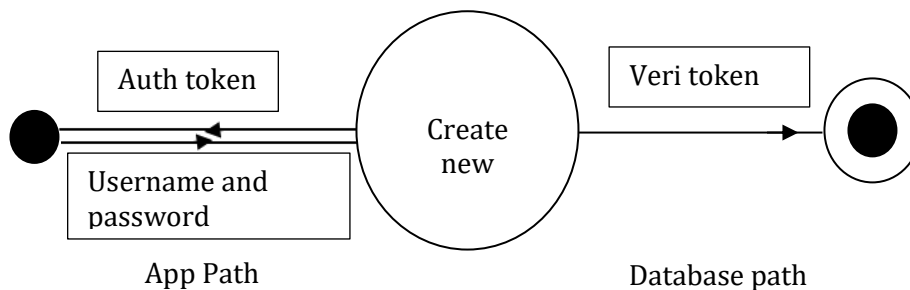
1. The App scans the username and password.
2. The App attempts to connect with the server.
3. The App encrypts the password and send username and encrypted password to server.
4. Else the App throws No response Error.
5. The Server receives the Username and Password and attempt to verify it using verification token from database.
6. The Server return an Authorization token if verification is successful.
7. Else Server returns No User message.
8. The App receives the auth token and store it.
9. The App proceeds with the login procedure.
10. Else the App throw No User error

### 5.1.3.2.1 Functional Requirements

The App need to identify the User that is using the app. The login interface permits registered users to access all the features associated with their account. Upon entering their username and password and clicking submit, the system validates their credentials. If the entered information is correct, users are granted access to the application. However, if the information is incorrect, an error message is displayed.

### 5.1.3.3 Account Creation (Server's perspective)

**Diagram:**



**Figure 3 – User Creation Process**

#### **Brief Description**

The App sends the Registration information, and the server returns the authorization token

#### **Initial Step-By-Step Description**

This use case requires certain actions to be taken before it can be initiated. The User has already installed the required App.

1. The App matches the entered Password and reentered Password
2. The App scans the username and password.
3. The App attempts to connect with the server.
4. The App encrypts the password and send username and encrypted password to server.
5. Else the App throws No response Error.
6. The Server receives the Username and Password and verify if the user already does not exist.
7. The Server generate new Verification token and stores it in the database.

Sound Sense  
(Smart Visual Alert System for Deaf)

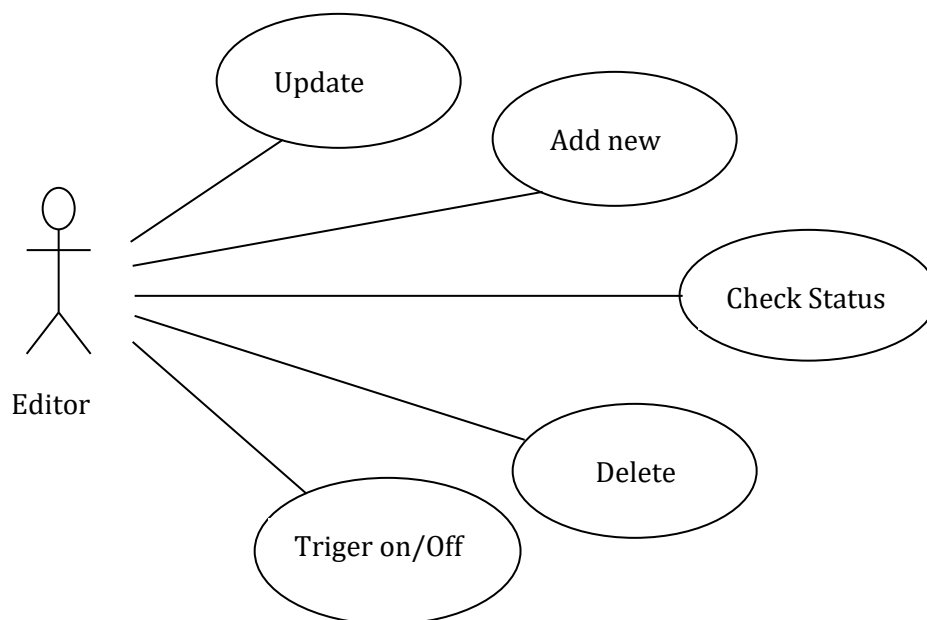
8. The Server return an Authorization token if user creation is successful.
9. Else Server returns User already exist message.
10. The App receives the auth token and store it .
11. The App proceeds with the login procedure.
12. Else the App throw User already exist error

*5.1.3.3.1 Functional Requirements*

The App need to identify the User that is using the app. In case the user does not possess an account, clicking on the register button will redirect them to the registration screen. In the registration screen a new user type in all their information and clicks submit, the data is then validated to make sure there is not an existing user with those credentials. If there is an existing user, then the user is asked to enter a new username. If there is no conflict with the credentials, then the user is registered.

*5.1.3.4 Use Case: App Reminders (App’s perspective)*

**Diagram:**



**Figure 4 – App Reminders System**

**Brief Description**

The App Reminders System has the following sets of use cases:

**Initial Step-By-Step Description**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

1. The User selects the reminders section in the app.
2. The App displays reminders section to the User.
3. The App waits for user input.

*5.1.3.4.1 Functional Requirements*

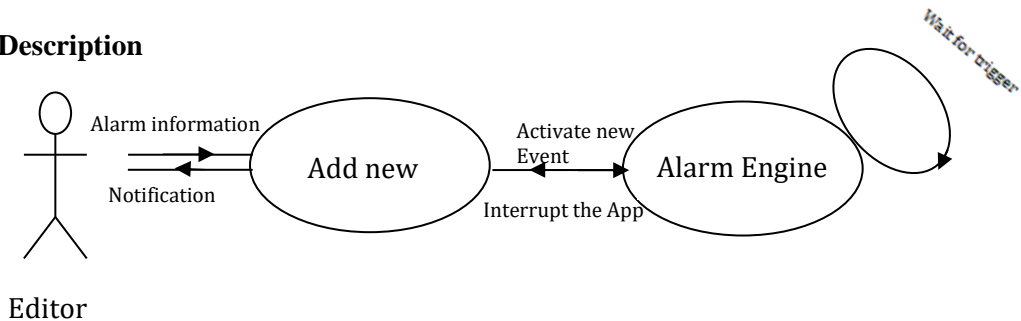
The system provides the facility to display the alarms and reminders using the light signals. A user can add, remove, update, trigger on and off, and check status of the reminder

*5.1.3.4.2 Use Case: Add Reminder*

**Diagram:**

**Figure 5 – App Reminders System**

**Brief Description**



**Brief Description**

The Add reminder allow user to add new Alarm:

**Initial Step-By-Step Description**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

1. The User selects the Add reminder option
2. The App asks for the time of reminder trigger, reminder Description, and Alarm settings

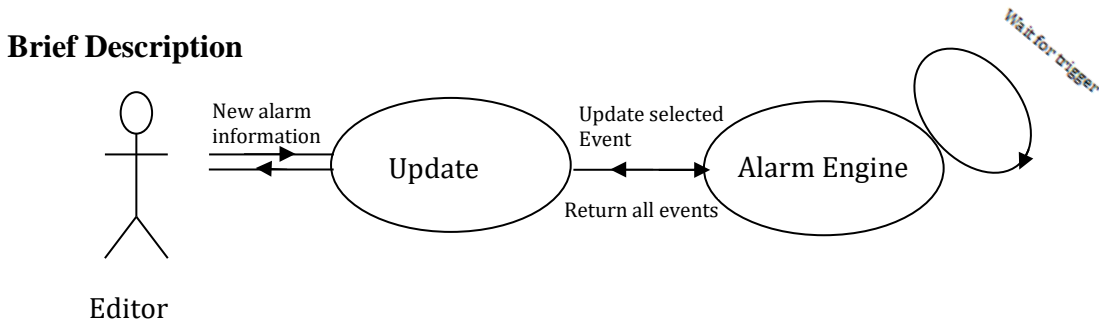
Sound Sense  
(Smart Visual Alert System for Deaf)

3. The user selects the time, enters the Reminder description and configure the alarm settings.
4. The App creates a new Event and push it to Alarm Engine.
5. The Alarm Engine checks through all events.
6. The Alarm Engine waits for the event trigger.
7. When the event is triggered the alarm engine Interrupts the app
8. The App respond to the interrupt and display notifications and trigger the alarm

5.1.3.4.3 Use Case: Update Reminder

**Diagram:**

Figure 6 – Update Reminders System



**Brief Description**

The Update reminder allow user to update a selected Alarm:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

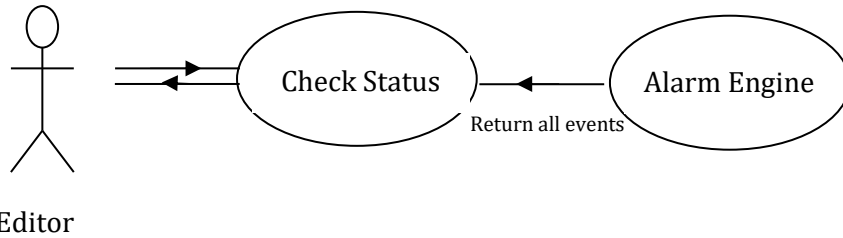
1. The User selects the View reminder option.
2. The App displays all the added reminders.
3. The User selects a reminder.
4. The App asks for the time of reminder trigger, reminder Description, and Alarm settings
5. The user selects the time, enters the Reminder description and configure the alarm settings.
6. The App creates a Update Event and push it to Alarm Engine.

5.1.3.4.4 Use Case: Check status Reminder

**Diagram:**

**Figure 7 – View Reminders System**

**Brief Description**



**Brief Description**

The Update reminder allow user to update a selected Alarm:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

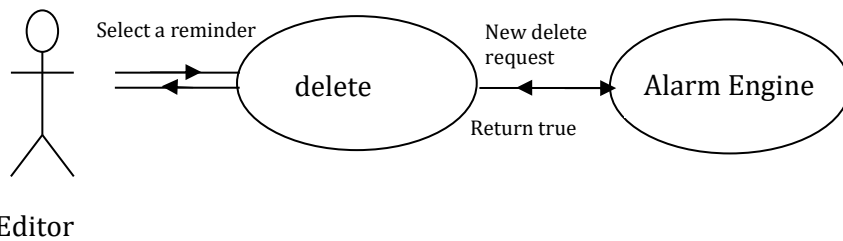
1. The User selects the view reminder option.
2. The App displays all the added reminders.

*5.1.3.4.5 Use Case: Delete Reminder*

**Diagram:**

**Figure 8 – Delete Reminders System**

**Brief Description**



**Brief Description**

The Update reminder allow user to update a selected Alarm:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

1. The User selects the View reminder option.

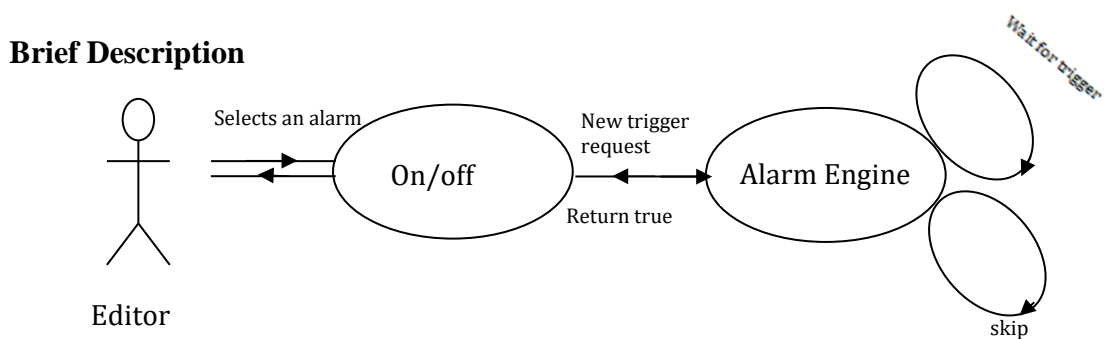
Sound Sense  
(Smart Visual Alert System for Deaf)

2. The App displays all the added reminders.
3. The User selects a reminder.
4. The user clicks on delete button
5. The App send the alarm engine a delete request with the event.
6. The alarm engine deletes the event and return true

5.1.3.4.6 Use Case: Trigger Reminder On or Off

**Diagram:**

**Figure 9 – Update Reminders System**



**Brief Description**

The Update reminder allow user to update a selected Alarm:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The User has already logged in the required App.

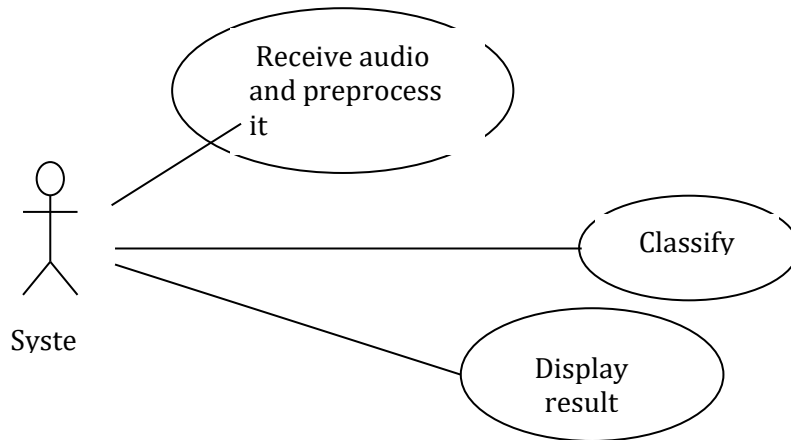
1. The User selects the View reminder option.
2. The App displays all the added reminders.
3. The User selects a reminder.
4. The User toggle the Alarm on/off
7. The App send the alarm engine a trigger request with the option and the event.
8. The alarm engine turns on or off the event and return true

5.1.3.5 Use Case: Sound Classification



Sound Sense  
(Smart Visual Alert System for Deaf)

**Diagram:**



**Figure 10 – Sound Classification System**

**Brief Description**

The App Sound classification has the following sets of use cases:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The system must be connected to the power supply and switched on, and along with that it must be connected to the microphone and LED

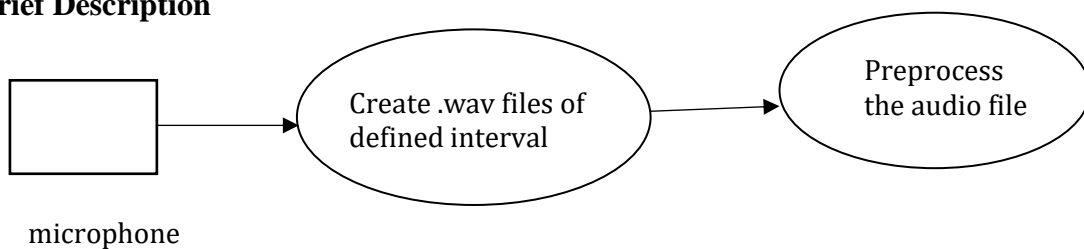
1. The system would receive audio via the mic.
2. The system would convert the input into numbers (tensors) and filter out the absence of sound.
3. The system would send the audio to the Ai classification model which will classify the sound and display the result via LED.

**5.1.3.5.1 Use Case: Receive Audio**

**Diagram:**

**Figure 11 – Receiver System**

**Brief Description**



**Brief Description**

The received audio allows the system to retrieve audio via a mic:

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The system should be connected to a mic.

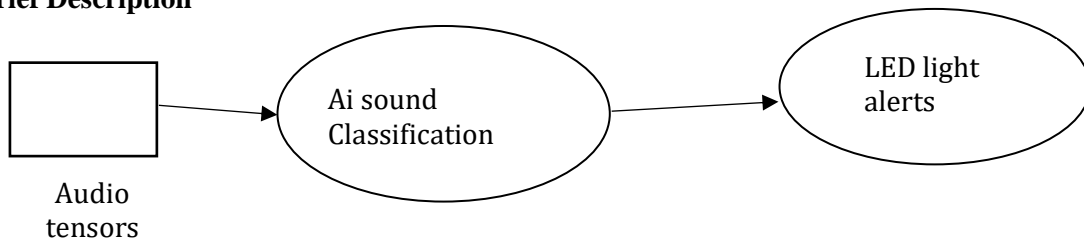
1. The system will activate the mic to receive sound input
2. The system would store the sound as a mono sound channel in wav files of 3-5 sec duration.
3. The system would apply a function on wav file to know the presence of audio in the file over a certain loudness level.
4. In case there is no sound present in the file or is not up to the defined threshold it would be discarded, and no further step would be taken.
5. The system would convert the sound to a tensor using a coded function which could be sent as input to the Ai algorithm for classification.

*5.1.3.5.2 Use Case: Classify Audio*

**Diagram:**

**Figure 12 – classify audio**

**Brief Description**



**Brief Description**

The sound classifier classifies the input sound event.

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The system shall take input from the mic and preprocess it

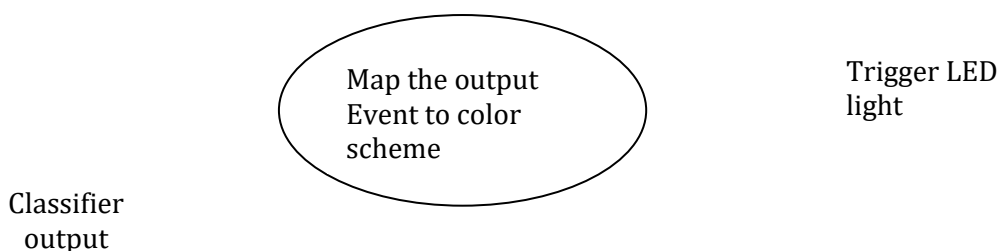
Sound Sense  
(Smart Visual Alert System for Deaf)

1. The system would send the preprocessed audio file to the Ai algorithm
2. The Ai algorithm would be a neural network developed using transferred learning techniques embedding the SAR model.
3. The Ai model will classify the sound into respective sound events.
4. The output would then trigger the LED light to visually alert the user.

*5.1.3.5.3 Use Case: Display Results*

**Diagram:**

**Figure 13 – display the result**



**Brief Description**

The output of the classifier would be mapped to the color scheme to display the respective color via LED light

**Initial Procedural Overview**

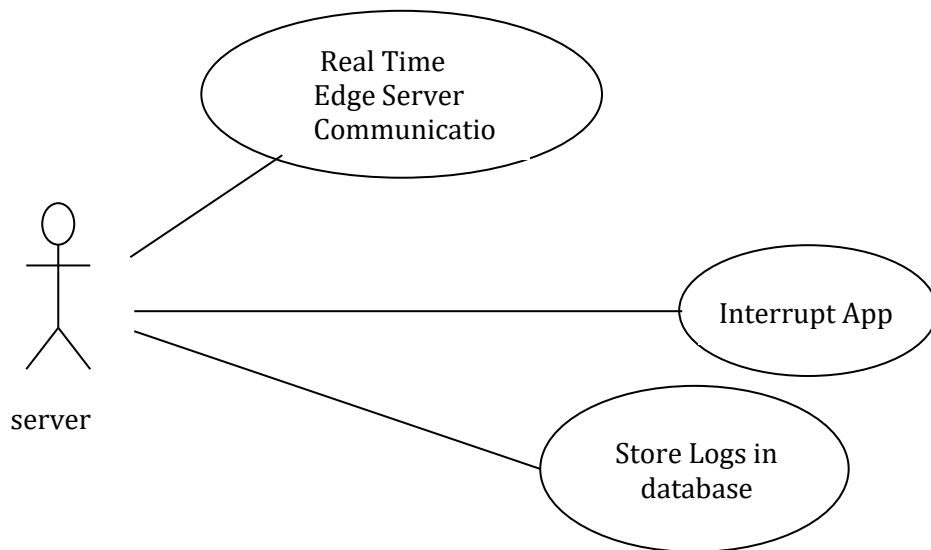
This use case requires certain actions to be taken before it can be initiated. The system shall process the input and classify the sound event.

- 1 The system would check for the color affiliated with the classified output
- 2 The system would turn the LED light on with that respective color for a defined interval of time.

*5.1.3.6 Use Case: Server-End*

Sound Sense  
(Smart Visual Alert System for Deaf)

**Diagram:**



**Figure 14 – Server use cases**

**Brief Description**

The server keeps records and provide communication between attached devices and mobile app

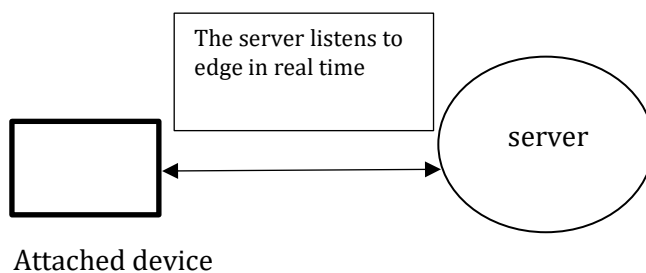
**Initial Procedural Overview**

The server must be hosted and Online

**5.1.3.6.1 Use Case: Real Time Edge Server Communication**

**Diagram:**

**Figure 15 – Poll attached devices**



**Brief Description**

The Server poll attached device for recognition and status.

**Initial Procedural Overview**

Sound Sense  
(Smart Visual Alert System for Deaf)

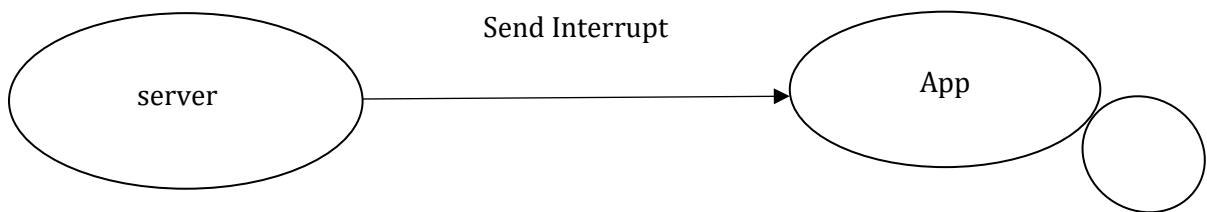
The server must be hosted and online

1. The Server will listen connected node in realtime
2. when a new device is connected it will request for recognition
3. The Server while polling will pick this request and register the attached device.
4. When server creates a new Response when device changes its own state.

*5.1.3.6.2 Use Case: Interrupt App*

**Diagram:**

**Figure 16 – Interrupt App**



**Brief Description**

The Server will interrupt the app to handle the response.

**Initial Procedural Overview**

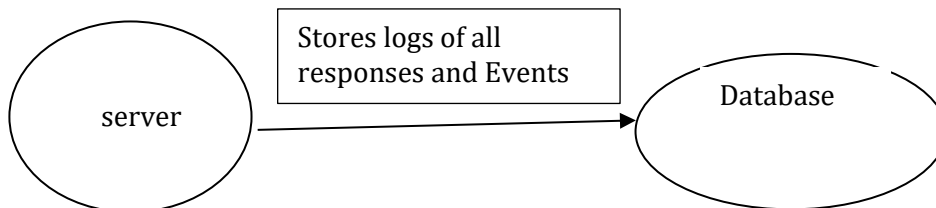
This use case requires certain actions to be taken before it can be initiated. The Server has already polled status of attached devices and created a response

1. The Server Generate an interrupt and send the interrupt to App
2. The App picks the interrupt and activate the interrupt handler.
3. The interrupt handler receives response from the server and activate the reminder system

*5.1.3.6.3 Use Case: Store Logs in database*

**Diagram:**

**Figure 13 – Store Logs in database**



**Brief Description**

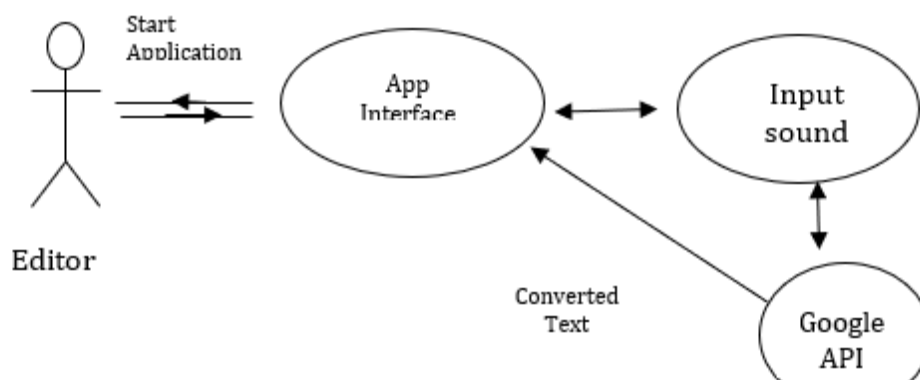
The server will keep records of all responses and events related to a user

**Initial Procedural Overview**

This use case requires certain actions to be taken before it can be initiated. The system shall process the input and classify the sound event.

- 1 The Server copies the response and push it to the database
- 2 The server serves the list of all responses when App requests it

**5.1.3.7 Use Case: Speech-to-text**



**5.1.3.7.1 Process**

**Brief Description**

The Speech-to-Text Converter converts speech audio to text.

**Initial Procedural Overview**

Sound Sense  
(Smart Visual Alert System for Deaf)

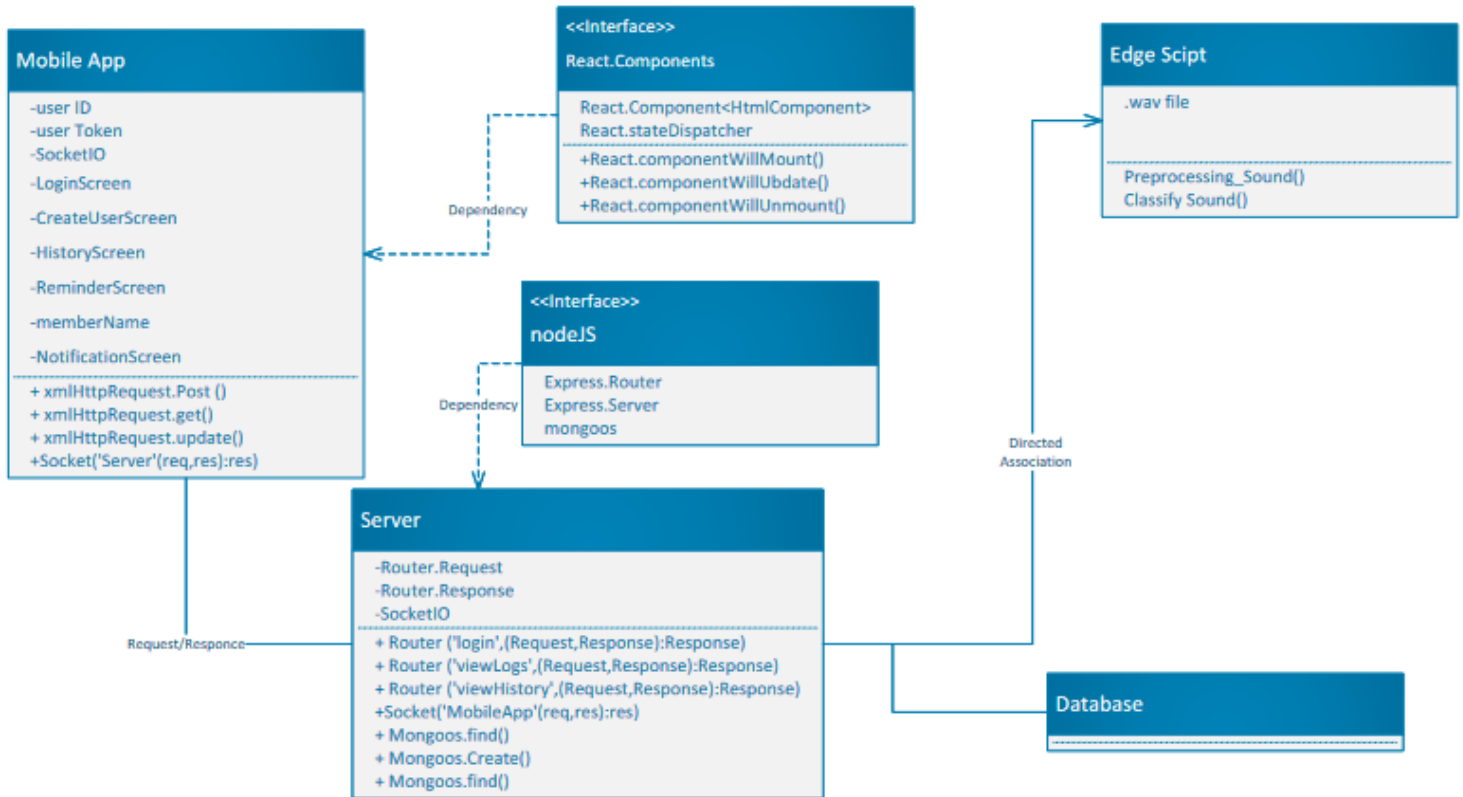
This use case requires certain actions to be taken before it can be initiated. The User would select this module and click on the record input button to start gathering audio to convert to text.

1. The User selects the Speech-to-Text option.
2. The App collects the audio input.
3. The App then implies Google API. The API then converts the audio to text.

The App then displayed the converted text to the user interface.

# Sound Sense (Smart Visual Alert System for Deaf)

## 5.4 Class Diagram

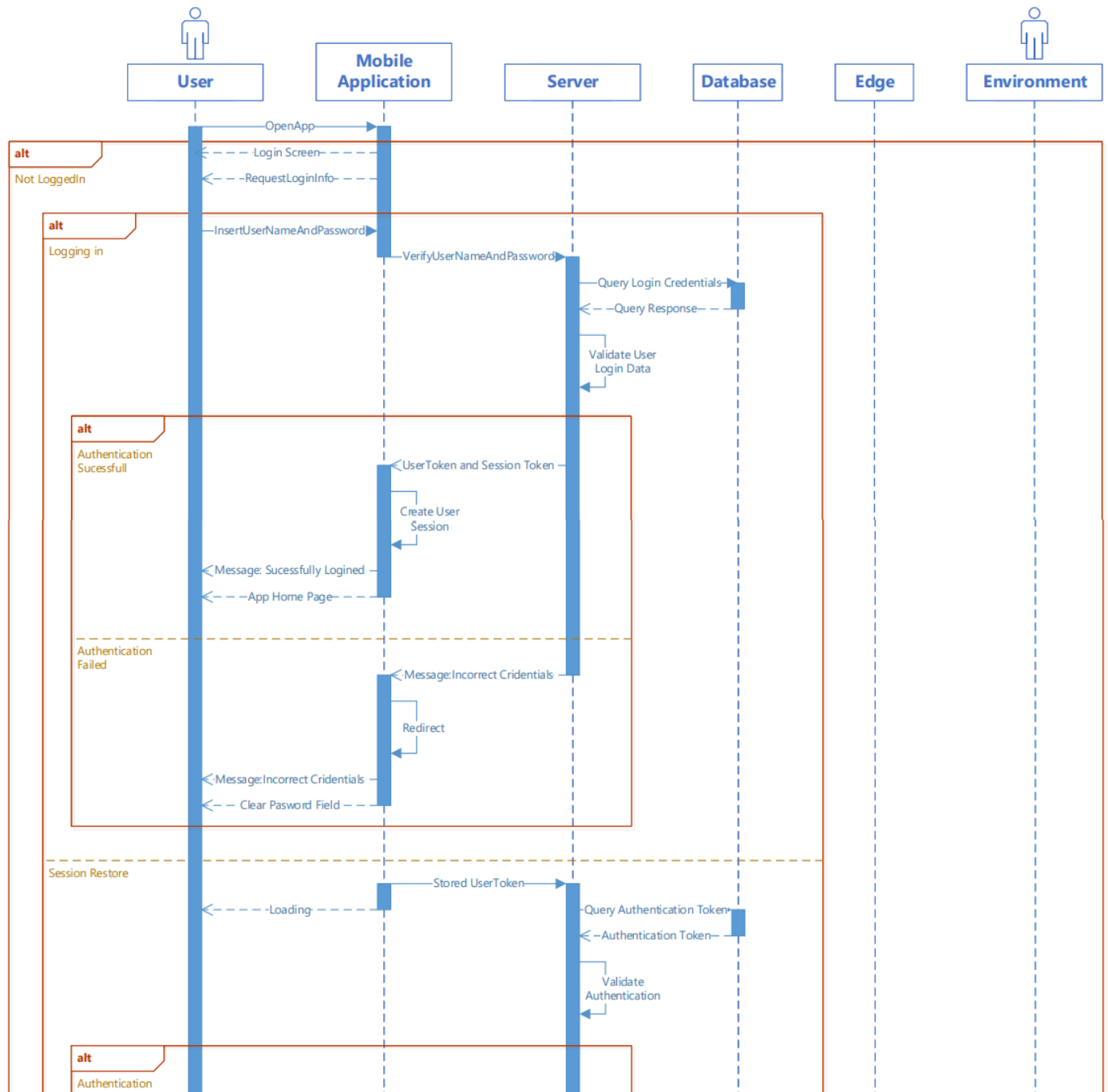




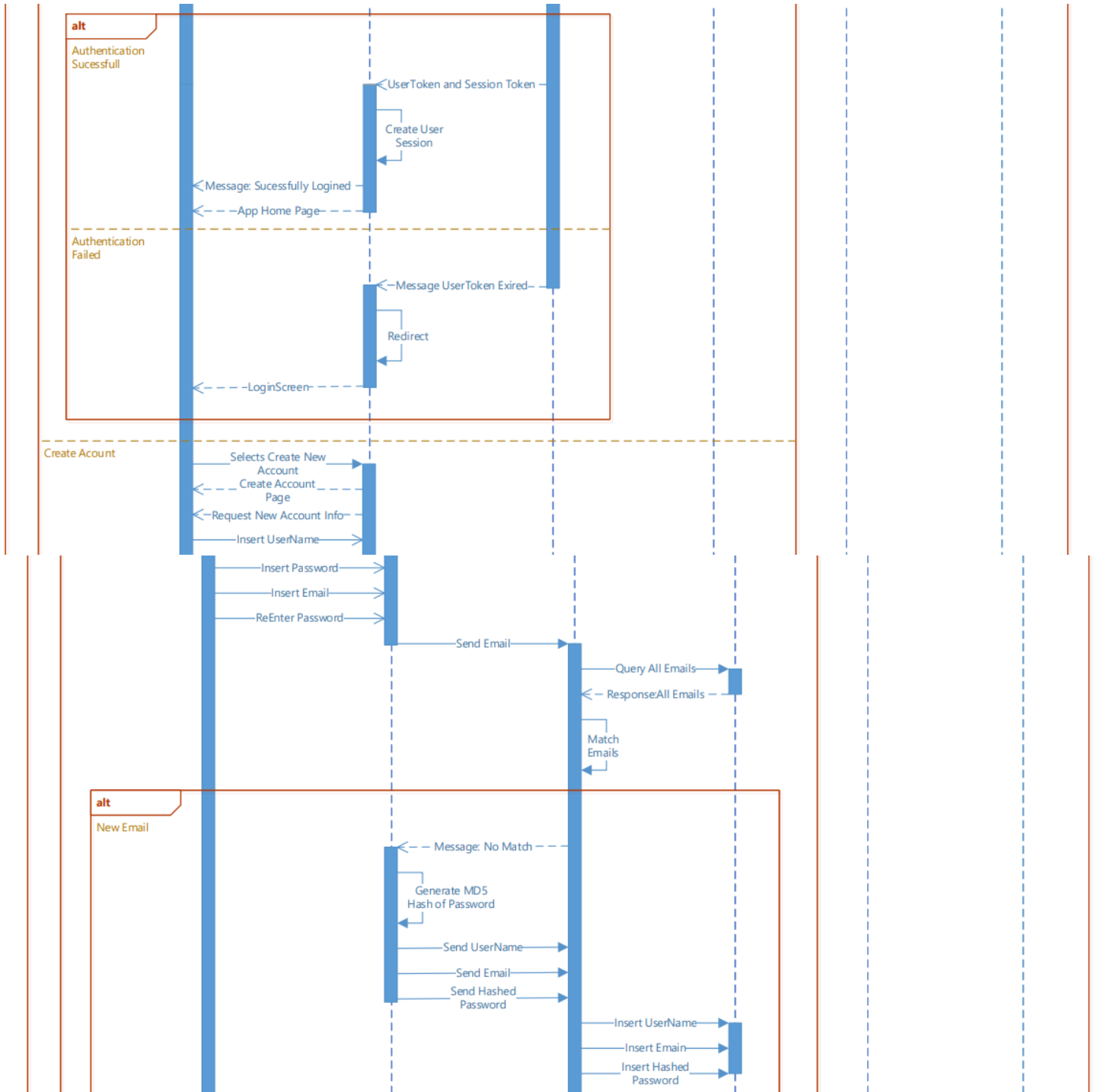
## 5.5 Detail System Design

### 5.5.1 Sequence Diagram

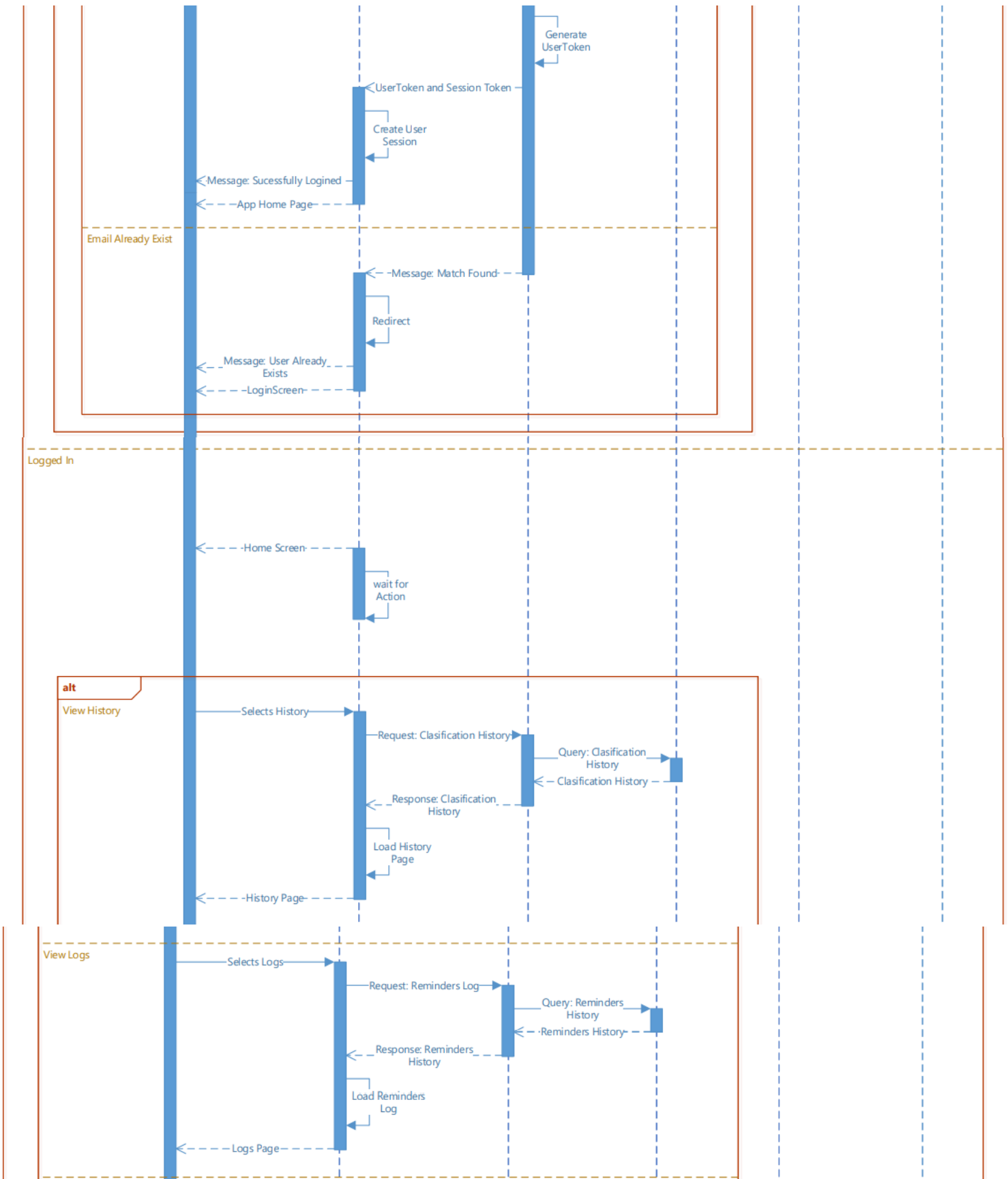
Following sequence diagrams show the sequence of activities performed in all the use cases delivered in SRS document version 1.2



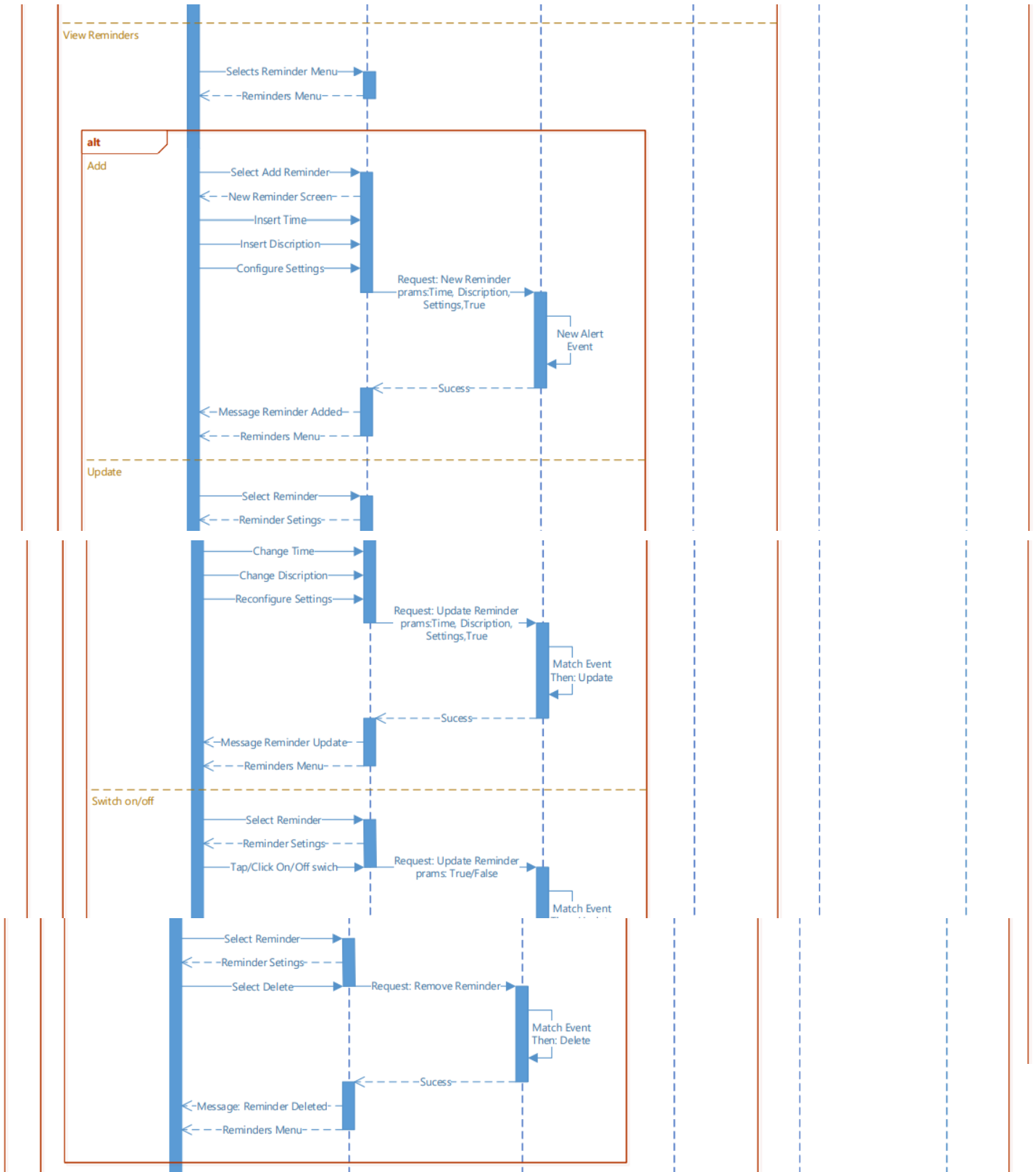
# Sound Sense (Smart Visual Alert System for Deaf)



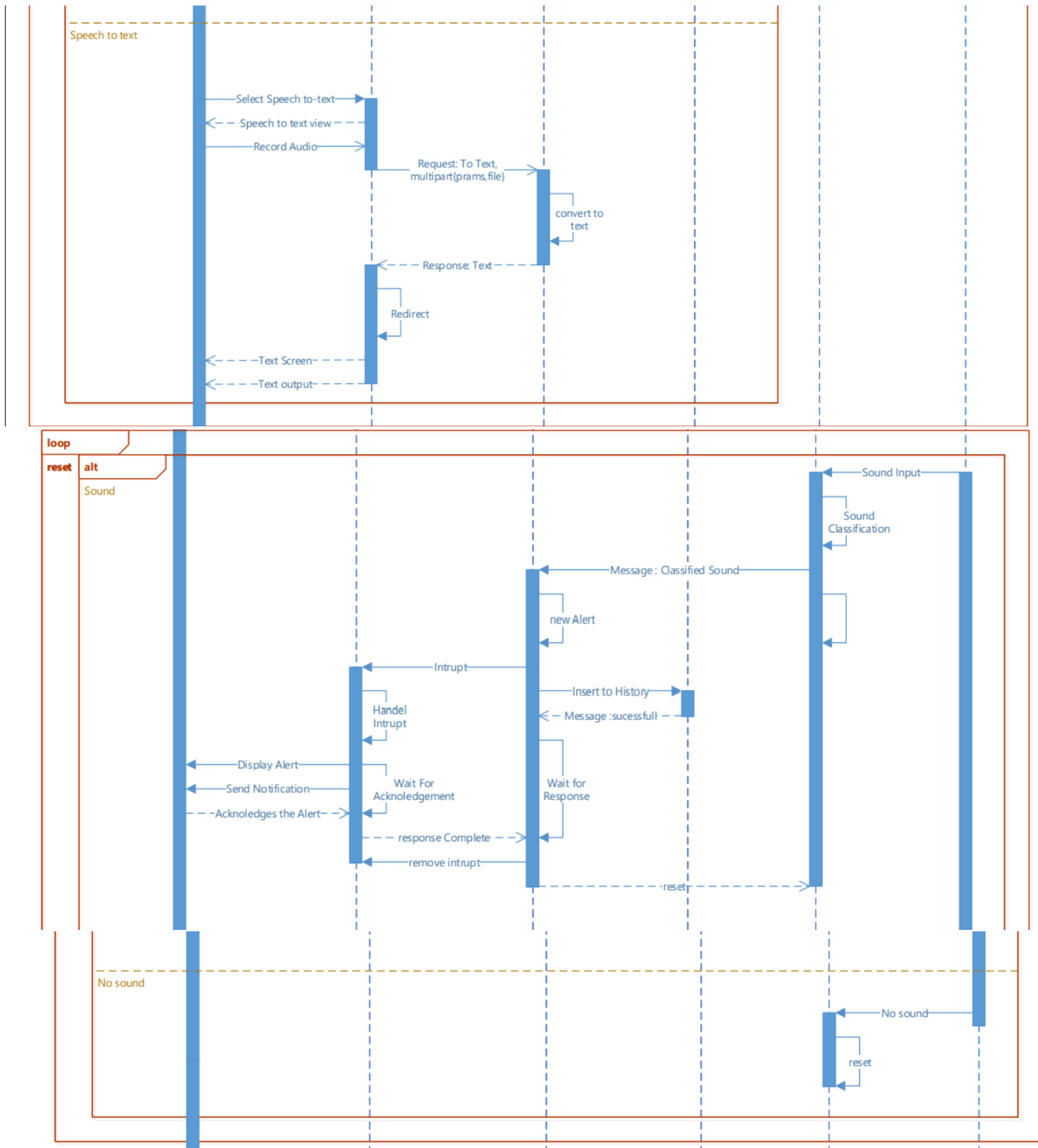
# Sound Sense (Smart Visual Alert System for Deaf)



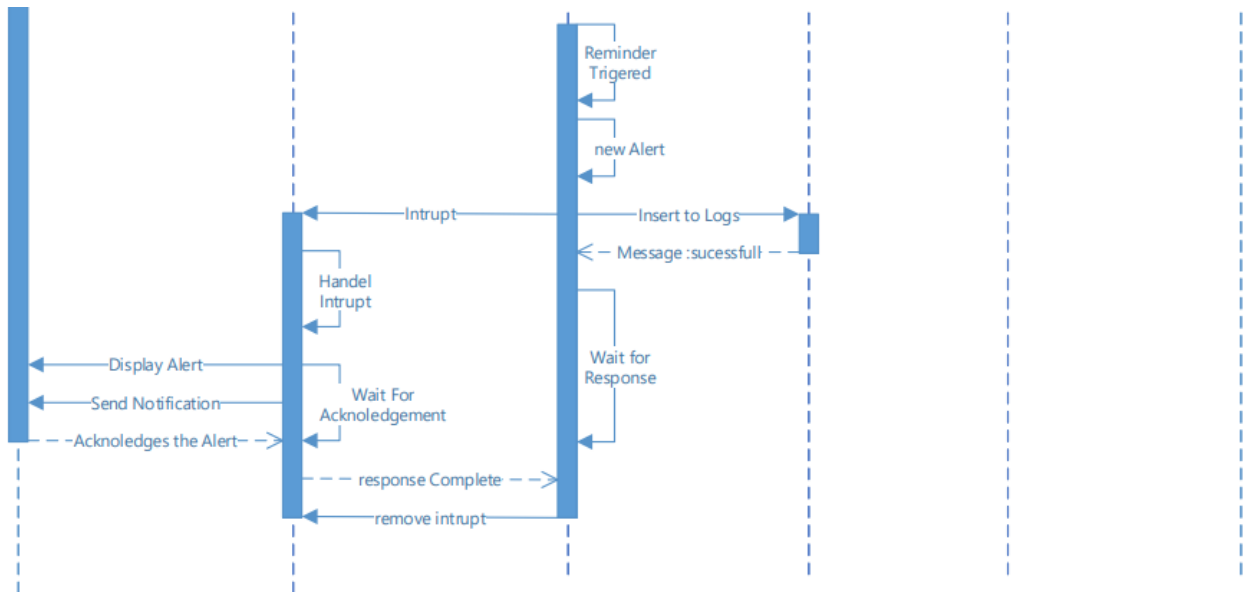
# Sound Sense (Smart Visual Alert System for Deaf)



# Sound Sense (Smart Visual Alert System for Deaf)



## Sound Sense (Smart Visual Alert System for Deaf)



### 5.5.2 Dynamic View (Activity Diagram)

In the activity diagram the dynamic view of the system is shown. All the activities are shown concurrently with their respective start and end state. There are three activity diagrams attached below:

1. Activity Diagram 1 displays the activities of Edge Computing Device
2. Activity Diagram 2 displays the activities of the sever
3. Activity Diagram 3 displays the activities of the application

Sound Sense  
(Smart Visual Alert System for Deaf)

5.5.2.1 Activity Diagram 1 (Edge Computing Device)

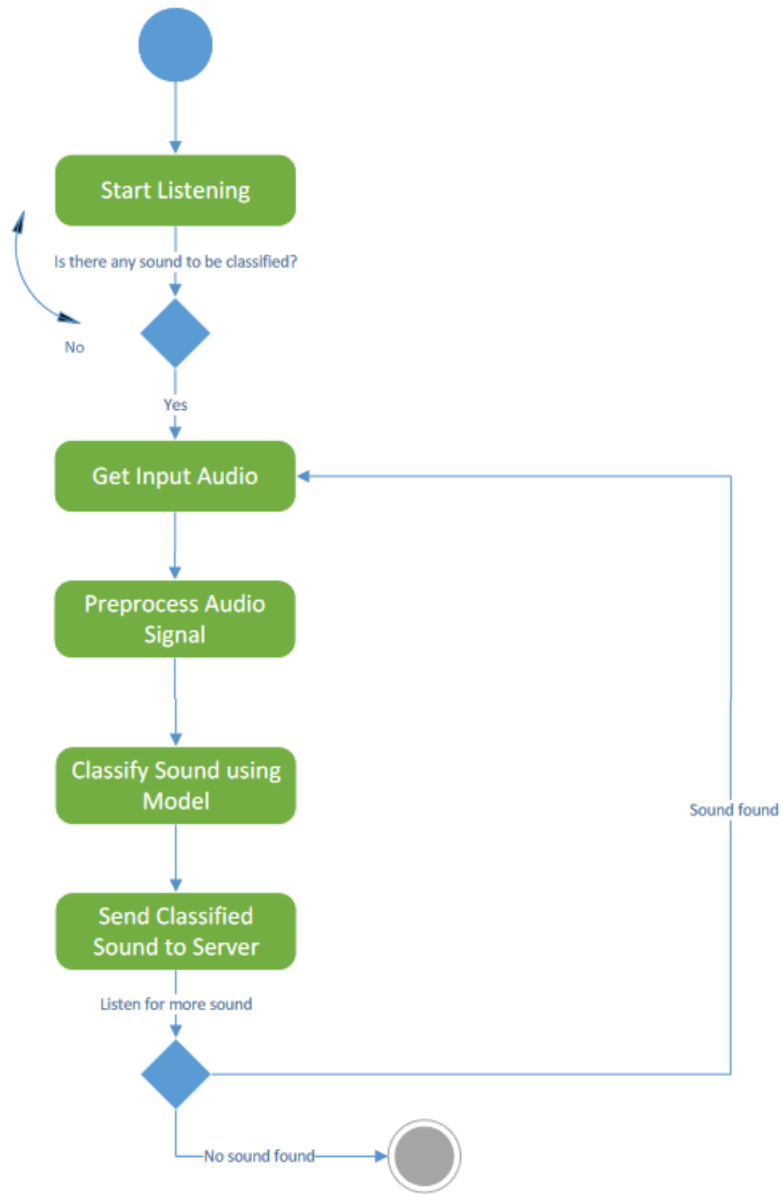
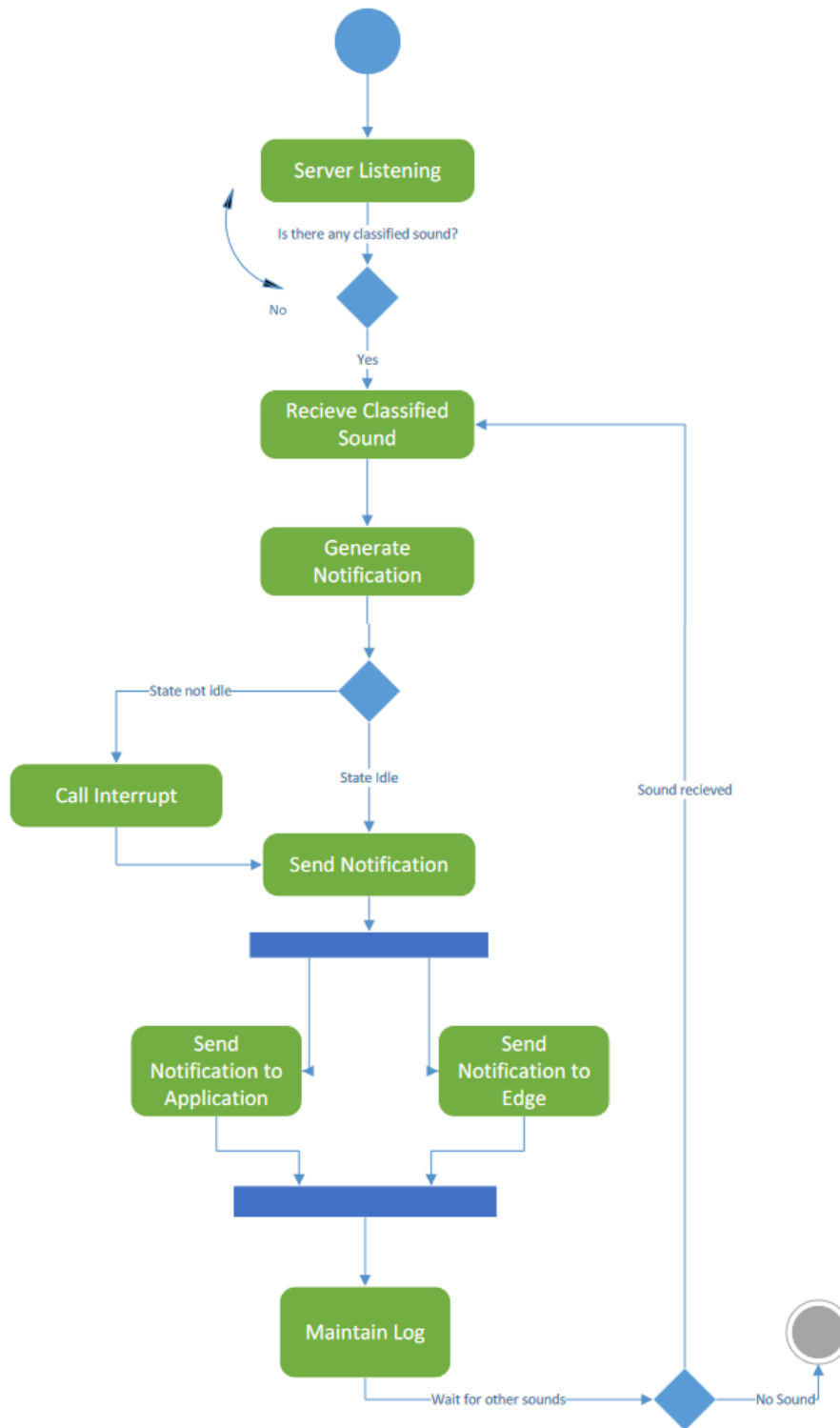


Figure 5.5.2.1

Sound Sense  
(Smart Visual Alert System for Deaf)

5.5.2.2 Activity Diagram 2 (Server)



Figure

5.5.2.2



Sound Sense  
(Smart Visual Alert System for Deaf)

5.5.2.3 Activity Diagram 3 (Application)

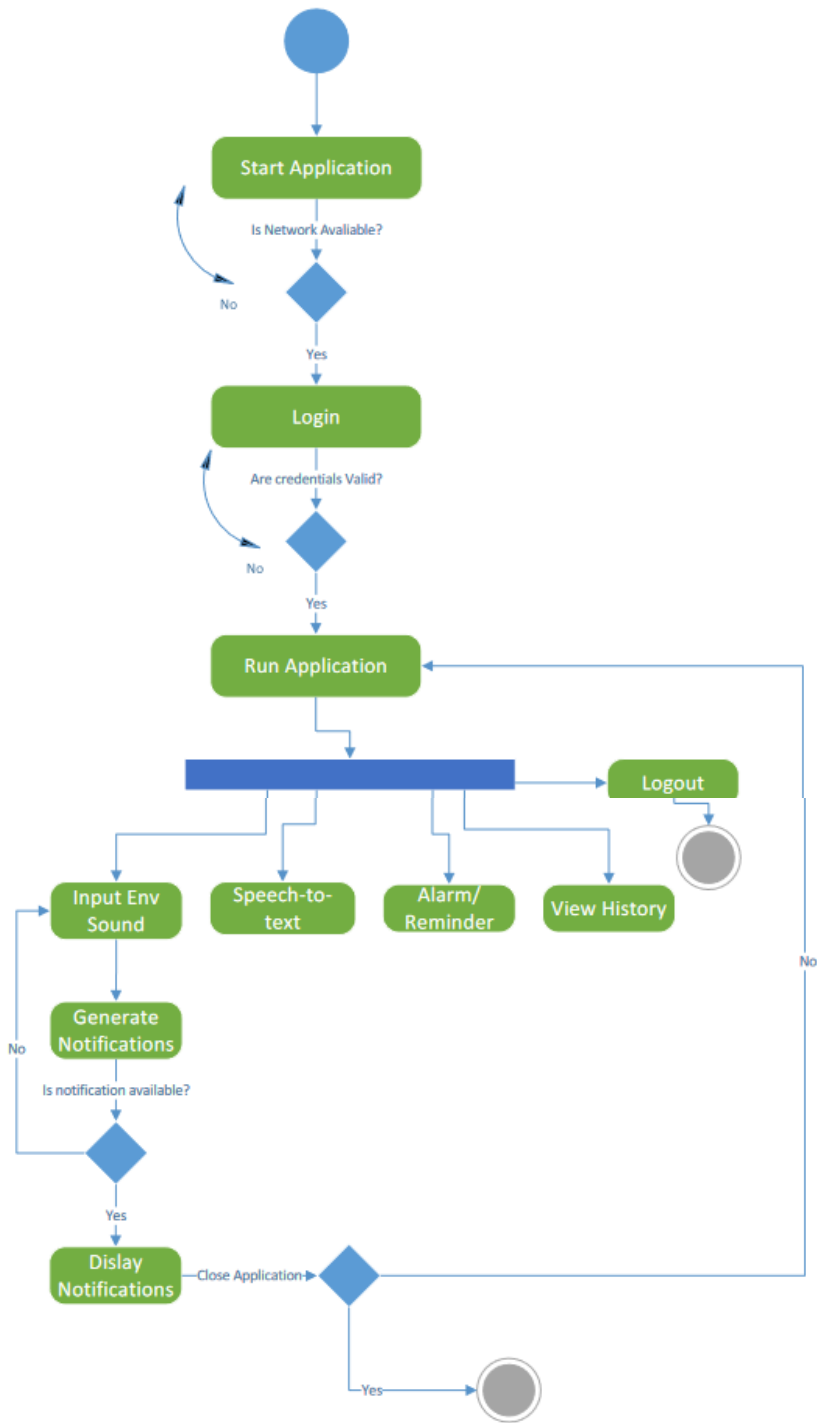


Figure 5.5.2.3

### 5.5.3 Logical View (State Diagram)

Following is the state diagram of Smart Visual Alert System showing all the states that the system has during the course of action. There are three activity diagrams attached below:

1. State Diagram 1 displays the activities of Edge Computing Device
2. State Diagram 2 displays the activities of the sever.
3. State Diagram 3 displays the activities of the application.

#### 5.5.3.1 State Diagram 1 (Edge Computing Device)

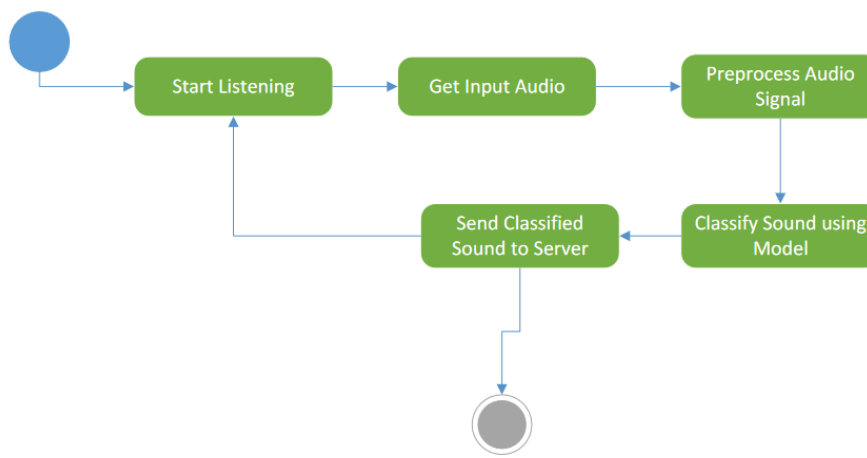


Figure 5.5.3.1

Sound Sense  
(Smart Visual Alert System for Deaf)

5.5.3.2 State Diagram 2 (Server)

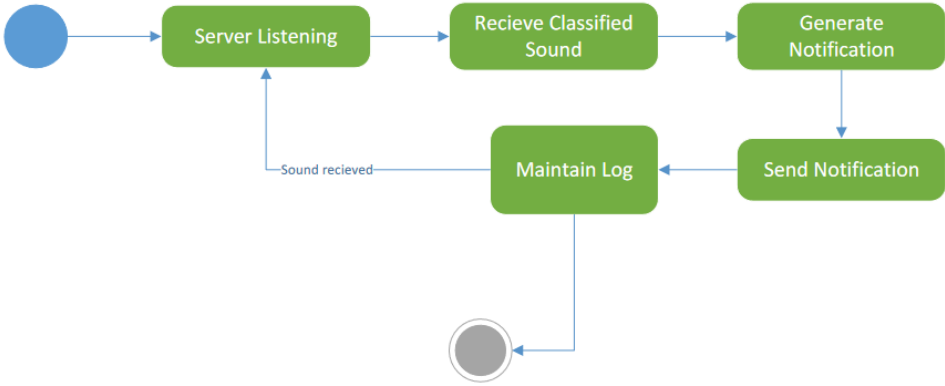


Figure 5.5.3.2

Sound Sense  
(Smart Visual Alert System for Deaf)

5.5.3.3 State Diagram 3 (Application)

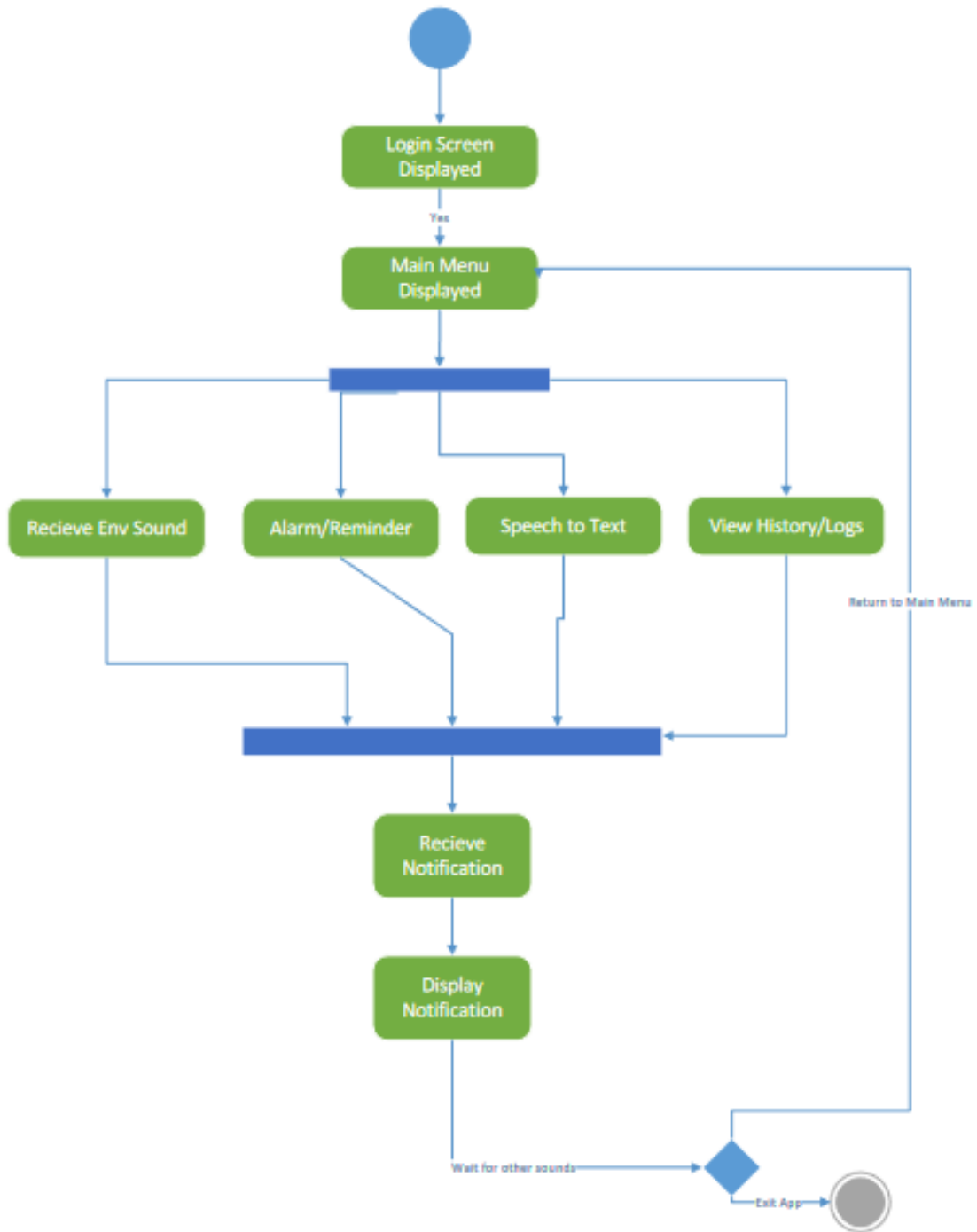


Figure 5.5.3.3

#### 5.5.4 Design Rationale

Smart Visual Alert System is a complex system incorporating different hardware and software components furthermore it is an application of Distributed computing. Distributed Computing is a field of computer science that focuses on the design and development of systems that involve multiple computers working together to solve a problem or perform a task.

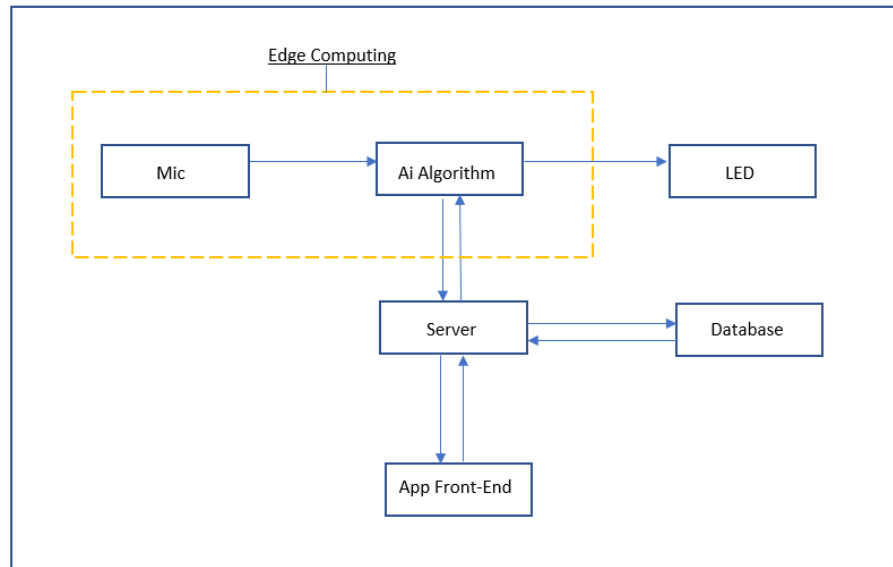
At high level the Smart Visual Alert System can be thought of as comprising three modules Sound detection and classification on Edge, Server, Mobile Application Front-End. Each component has been tasked with a specific responsibility to perform. Mobile application is responsible for interacting with the user while server-side handles the back end.

Layered Architecture has been implemented at the Edge computing device that receive sound, pre-process it, classify it and send the result to the server all the steps are performed in defined sequence.

For the Mobile Application it is suitable that we use Model View Control (MVC) as the design pattern. Model is basically the database that logs all the details needed or supplied by different functionalities provided in application. View is basically the Mobile Application Front-End that interact with the user and Controller is the main connector component that connects the front-end, database and input signal from the edge computing device and hold all the business logic for the features implemented.

The components are assigned individual tasks and can perform their work independently, yet they follow a certain flow in terms of data and control, resulting in high cohesion. Additionally, once a component finishes its task, it communicates its state to other components with minimal interaction. Consequently, that component will come into action. That led us to low coupling.

## Sound Sense (Smart Visual Alert System for Deaf)



### 5.5.5 Data Description

#### 5.5.5.1 Data Description

The Simple Audio Recognition (SAR) model is a deep learning model that is trained on a large dataset of audio samples. The dataset used to train the model typically consists of thousands or even millions of audio samples. Each audio sample in the dataset is typically labeled with a specific category or class, such as a particular keyword or environmental sound.

The audio samples are usually preprocessed to extract relevant features such as the frequency content of the signal, which are then used as input to the SAR model. The model architecture typically consists of several layers of convolutional neural networks (CNNs) that extract increasingly complex features from the input audio signal. These layers are followed by one or more fully connected layers that perform the final classification of the audio sample.

The SAR model is typically trained using a supervised learning approach, where the model is presented with labeled audio samples and learns to associate specific features of the audio signal with the corresponding class label. The model is trained using an optimization algorithm such as stochastic gradient descent, which minimizes the difference between the predicted class label and the true label for each audio sample in the training dataset.

## Sound Sense (Smart Visual Alert System for Deaf)

Once the SAR model is trained, it can be used to classify new, unlabeled audio samples by extracting the relevant features from the audio signal and passing them through the trained model. The output of the model is a probability distribution over the different possible classes, which can be used to determine the most likely class label for the input audio signal.

In summary, the SAR model is a deep learning model that is trained on a large dataset of labeled audio samples. The model architecture typically consists of several layers of CNNs that extract features from the input audio signal, followed by one or more fully connected layers that perform the final classification. The model is trained using a supervised learning approach and can be used to classify new, unlabeled audio samples by extracting features from the audio signal and passing them through the trained model.

### *5.5.5.2 Data Dictionary*

The command "react-scripts start" initializes the development environment, starts a server, and enables hot module reloading.

**Query** objects are returned when calling many Model methods, providing a chaining API for specifying search terms, cursor options, hints, and other behavior.

**Authentication** is the process of verifying a user's identity by obtaining credentials and confirming the user's identity using those credentials.

**Authorization** is the process of granting or denying system access permissions to authenticated users, allowing them to access resources.

**Validation** is a technical process used by web-forms to check if the information provided by a user is correct. The form either alerts the user of errors that need to be fixed before proceeding or validates the form, allowing the user to continue with their registration process.

**MD5: (string) => string:** generates a md5 hash of a text string

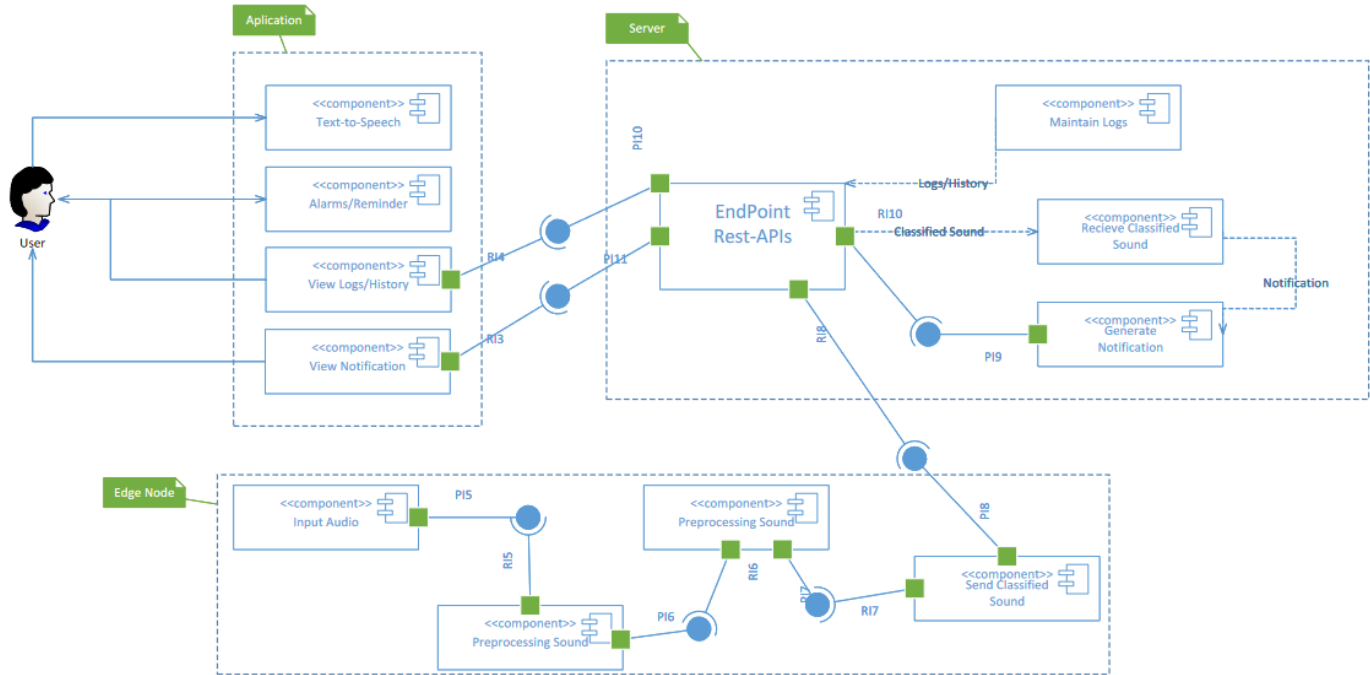
**Alert Event:** An Alert Event is a Triggerable Event that is listened by the server to send notifications to user. An Alert is generated when Alert Event is triggered

**Interrupt:** A processor receives an interrupt to temporarily pause the execution of the current code and attend to a specific event or task in order to ensure timely processing. An interrupt is a request for the processor to interrupt the currently executing code, so that the event can be

# Sound Sense (Smart Visual Alert System for Deaf)

processed in a timely manner.

## 5.5.6 Component Diagram



## 6. Implementation and Testing

### Introduction

The purpose of this document is to offer the necessary test documentation for the Sound Sense project. It will aid in the technical aspect of testing by providing detailed test cases for black box testing. Each test case outlines important information such as the person responsible for conducting the test, the preconditions necessary to carry out the test, the specific item being tested, and the input, expected output, or results. If applicable, the test case also includes procedural steps.

### Test Objectives



## Sound Sense (Smart Visual Alert System for Deaf)

The aim of this document is to provide additional information that is necessary to carry out the required tests, as a supplement to the test plan. By including comprehensive test information, vital components will not be neglected, and test coverage will be enhanced. Testers can utilize the test cases outlined in this document to proceed with testing. The purpose of this document is to supplement the test plan by furnishing specific details essential for conducting the required tests. By including comprehensive test information, vital components will not be neglected, and test coverage will be enhanced. Testers can utilize the test cases outlined in this document to proceed with testing.

### **Test Items**

The testing process should consider the following major modules/functionalities based on the requirements of Sound Sense as outlined in Section 5 of this document.

1. Sound Input
2. sound Preprocessing
3. Spectrogram generation
4. Sound Classification
5. Display Results/Notifications
6. Reminder's timing
7. Speech to text

### **Features to be tested.**

1. The Edge device shall be able to receive sound from the environment.
2. The Edge device shall be able to generate spectrograms of the input sound
3. The Edge device shall be able to classify the input sound using the CNN model.
4. The Edge device shall be able to transfer the classification results from the edge to the server.
5. The server shall be able to maintain logs in the database.
6. The server shall be able to generate alerts on mobile phones.

Sound Sense  
(Smart Visual Alert System for Deaf)

7. The server shall be able to generate an alert when the reminder goes off and furthermore send that alert to mobile phone and edge device.
8. The mobile app (front-end) shall be able to allow users to log in/sign up.
9. The mobile app shall allow the user to view the history of occurred events.
10. The mobile app shall allow users to set up a reminder.
11. The mobile app shall allow a user to record sound that is to be converted into text.
12. The mobile app shall allow user to view all the connected edge devices.

## **Detailed Test Strategy**

The Sound Sense project consists of several components, which necessitates the independent development of system modules, followed by their integration. The comprehensive approach includes White Box and Black Box testing through Unit Testing, with Integration Testing conducted to effectively integrate the system.

## **Unit Testing**

At the code or source level, Unit Testing is carried out to identify programming errors specific to the language, such as faulty syntax or logical errors, and to test functions or code modules. A possible rephrased sentence could be: "The design of unit test cases aims to verify the accuracy and correctness of the program's functionality."

## **White box testing**

White box testing involves bypassing the user interface and testing inputs and outputs directly at the code level. The results are then compared against the requirements. Unlike black box testing, this method does not focus on the program's intended functionality, but rather evaluates its code and structure. The generated test cases must execute each condition at least once. To achieve this, we will use Basis (alternative) Path Testing,

which is practical for the program's relatively straightforward functionality. The goal is to ensure the program's correctness.

## **Black Box Testing**

Black box testing is a software testing technique where the tester doesn't have access to the internal workings of the system being tested. It involves evaluating the system based on its external behavior and output without knowledge of its internal implementation details. This approach involves running tests with various inputs to verify that the system produces the correct output, as a user who interacts with the software/application would experience it.

## **Integration Testing**

Integration testing involves testing the combined functionality of previously tested modules to ensure they operate correctly when integrated.

## **Incremental Testing**

The complete application testing requires the integration of eight primary modules. An incremental testing strategy will be used to integrate the upcoming modules with the already integrated components. This means that each module will be integrated and tested incrementally until the entire system is complete. The development team will be responsible for performing the integration testing.

1. Real time Audio Acquisition
2. Preprocessing and Classification
3. LED light control
4. Edge-Server Communication
5. Real time Alert Display on the Application
6. Log Maintenance

Sound Sense  
(Smart Visual Alert System for Deaf)

7. Reminder System
8. Speech-to-text

### Item Pass/Fail Criteria

The criteria for determining the pass or fail status of a test item are outlined as follows, and the test cases' details are specified in the Test Deliverables section. The item's pass or fail status would depend on the principles described below:

1. Preconditions are met.
2. Inputs are executed as specified.
3. The output functions as described in the output specification => Pass.
4. The system does not perform as stated in the output specification => Fail.

### Suspension Criteria and Resumption Requirements

The testing process will be halted if a defect is detected that prevents any further testing. Once the defect has been resolved, the testing will resume.

### Testing Deliverables

#### Test case 1 – Acquire Audio

Test Case Name	Acquire Audio
Test case number	1
Description	The test case is aimed at checking the audio acquisition functionality of Sound Sense. The system shall record audio in real-time using the mic attached in the edge device
Testing technique used	White box testing
Precondition	mic to be properly connected and turned on

Sound Sense  
(Smart Visual Alert System for Deaf)

Input	Audio from the environment in real-time
Steps	The audio recording initiates as we run the python script on the edge device
Expected Output	A sound file in .wav format with 5 sec of audio
Alternative Path	N/A
Actual Output	Confirmed

**Test case 2 – Audio Preprocessing**

Test Case Name	Preprocesses acquire audio
Test case number	2
Description	The test case is aimed at checking the audio preprocessing functionality of Sound Sense that includes generating spectrogram images of the input audio.
Testing technique used	White box testing
Precondition	Test case 1 is satisfied
Input	A 5 sec audio file in .wav format
Steps	Using TensorFlow library we generate spectrogram of input audio.
Expected Output	Spectrogram image
Alternative Path	N/A
Actual Output	Confirmed

**Test case 3 – Audio Classification**

Test Case Name	Audio Classification.
Test case number	3

Sound Sense  
(Smart Visual Alert System for Deaf)

Description	This test case checks how the model performs and gives its output.
Testing technique used	Black box testing.
Precondition	Test cases 1 and 2 are satisfied.
Input	Spectrogram image of the input audio.
Steps	The spectrogram is fed into the model
Expected Output	Classification results in form of a string classifying the input audio.
Alternative Path	N/A
Actual Output	Confirmed

**Test case 4 – Edge-Serve communication using Sockets.**

Test Case Name	Edge-Server Communication using Sockets
Test case number	4
Description	The test case is aimed at checking the connection between server and edge device
Testing technique used	White box testing
Precondition	Server and Edge device is to be connected to same network
Input	IP address and port number
Steps	A client-server connection to be established between edge device and server
Expected Output	Successful connection between Edge device and server
Alternative Path	N/A
Actual Output	Confirmed

**Test case 5 – Sending and Receiving of messages between Edge device and**

Sound Sense  
(Smart Visual Alert System for Deaf)

**Server.**

Test Case Name	Sending and receiving of messages between Edge device and Server
Test case number	5
Description	The test case is aimed at checking the successful sending and receiving of messages between Edge device and Server
Testing technique used	White box testing
Precondition	Test Case 4 is satisfied.
Input	N/A
Steps	The Edge device be able to send the classification results to server and receive alerts from the server.
Expected Output	Successful communication between Edge device and Server
Alternative Path	N/A
Actual Output	Confirmed

**Test case 6 – Authentication**

Test Case Name	Authentication
Test Case Name	6
Description	This test case checks if the user is able to successfully authenticate using valid credentials.
Testing technique used	Black box testing
Precondition	The user has valid credentials for authentication
Input	Username and password
Steps	<ul style="list-style-type: none"> <li>• Navigate to the login page.</li> <li>• Enter the username and password in the respective fields.</li> <li>• Click on the login button</li> </ul>

Sound Sense  
(Smart Visual Alert System for Deaf)

Expected Output	The user is successfully authenticated and redirected to the home page
Alternative Path	If the user enters invalid credentials, an error message is displayed and the user remains on the login page
Actual Output	Confirmed.

**Test case 7 – Alarm Activation**

Test Case Name	Alarm Activation
Test case number	7
Description	This test case checks if the alarm activates at the set time.
Testing technique used	Black box testing
Precondition	The alarm is set for a specific time.
Input	Time
Steps	<ul style="list-style-type: none"> <li>• Set the alarm for a specific time.</li> <li>• Wait for the set time to be reached.</li> </ul>
Expected Output	The alarm activates at the set time and produces an audible alert.
Alternative Path	If the alarm does not activate at the set time, the test case fails.
Actual Output	Confirmed

**Test case 8 – React. render()**

Test Case Name	React. render()
Test case number	8
Description	This test case checks if the React.render() method correctly renders a React element into the DOM
Testing technique used	White box testing.
Precondition	A valid React element and a container DOM element are available



Sound Sense  
(Smart Visual Alert System for Deaf)

Input	React element and container DOM element.
Steps	<ul style="list-style-type: none"> <li>• Call the React.render() method with the React element and container DOM element as arguments.</li> <li>• Observe the container DOM element.</li> </ul>
Expected Output	The React element is rendered correctly into the container DOM element.
Alternative Path	If the React element is not rendered correctly or an error occurs, the test case fails.
Actual Output	Confirmed

**Test case 9 –Serve Sockets.**

Test Case Name	Server Sockets
Test case number	9
Description	This test case checks if the server can successfully create and bind a socket to listen for incoming connections.
Testing technique used	White box testing
Precondition	The server is running and the network is available
Input	IP address and port number
Steps	<ul style="list-style-type: none"> <li>• On the server, create a socket using the given IP address and port number.</li> <li>• Bind the socket to the IP address and port number.</li> <li>• Start listening for incoming connections on the socket.</li> </ul>
Expected Output	The server successfully creates and binds the socket and starts listening for incoming connections
Alternative Path	If the server is unable to create or bind the socket or an error occurs, the test case fails.
Actual Output	Confirmed

## 6.1 Environment Needs

This visual representation illustrates an overview of the application, displaying its different modules, their interconnections, and the flow of data between them:

### 6.1.1 Hardware

1. Raspberry-Pi

## Sound Sense (Smart Visual Alert System for Deaf)

2. Microphone
3. LED Lights
4. Raspberry-Pi power supply
5. Mobile phones

### 6.1.2 Software

1. IDE: Jupyter Notebook and Visual Studio Code
2. Python Libraries: - Pandas, NumPy, ScikitLearn, TensorFlow, Scipy
3. Application Frontend: - HTML, CSS, Bootstrap, JavaScript, React
4. For Backend Server: - Nodejs
5. Database: - Mongo DB

## Responsibilities

All members of the development team are responsible for carrying out both nits testing and integration testing tasks.

## Staffing and Training Needs

Developers are required to have a fundamental understanding of testing strategies and techniques, such as Black Box testing and Integration testing, in order to test the project. Each developer will test their colleagues' work and contribute to the development and testing of the project concurrently.

## Risks and Contingencies

Although efforts have been made to minimize the chances of failure, unforeseen circumstances such as network issues, corrupt input data, or system failure may still occur. The team will implement thorough error handling procedures to address these issues, but there may still be some unexpected challenges.

## Schedule Risk

To avoid delays in project delivery, it may be necessary to allocate additional hours per day. However, this decision will depend on the feasibility and availability of resources.

## Budget Risk

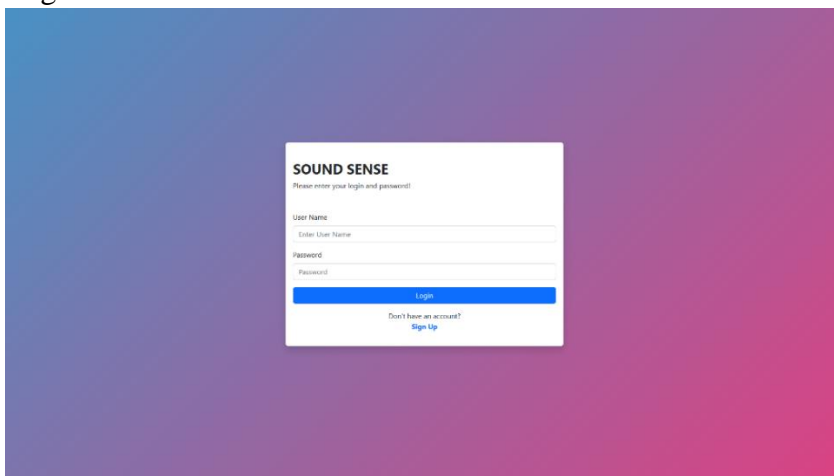
The project's budget will be managed by utilizing more cost-effective alternatives to ensure it stays within the allocated budget.

## 6.2 User Interface Design

### 6.2.1 Desktop Application

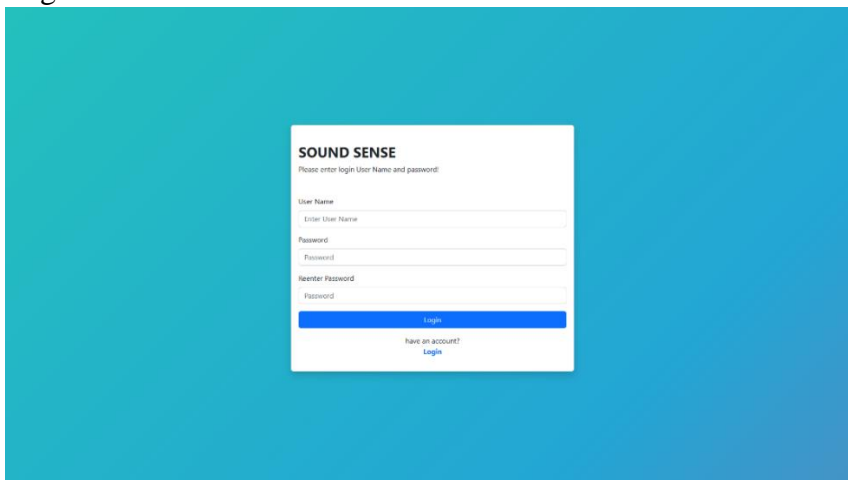
Introducing our sleek and user-friendly desktop application designed to make your life easier. With its modern interface and intuitive features, you can effortlessly manage your tasks, stay organized, and boost your productivity. Our powerful desktop app is optimized for performance and packed with all the essential tools you need to get things done quickly and efficiently.

Login:



The screenshot shows a login form for 'SOUND SENSE' centered on a purple-to-pink gradient background. The form is white with a thin border and contains the following elements: the title 'SOUND SENSE', a sub-header 'Please enter your login and password!', a 'User Name' label with an input field, a 'Password' label with an input field, a blue 'Login' button, and a link 'Don't have an account? Sign Up'.

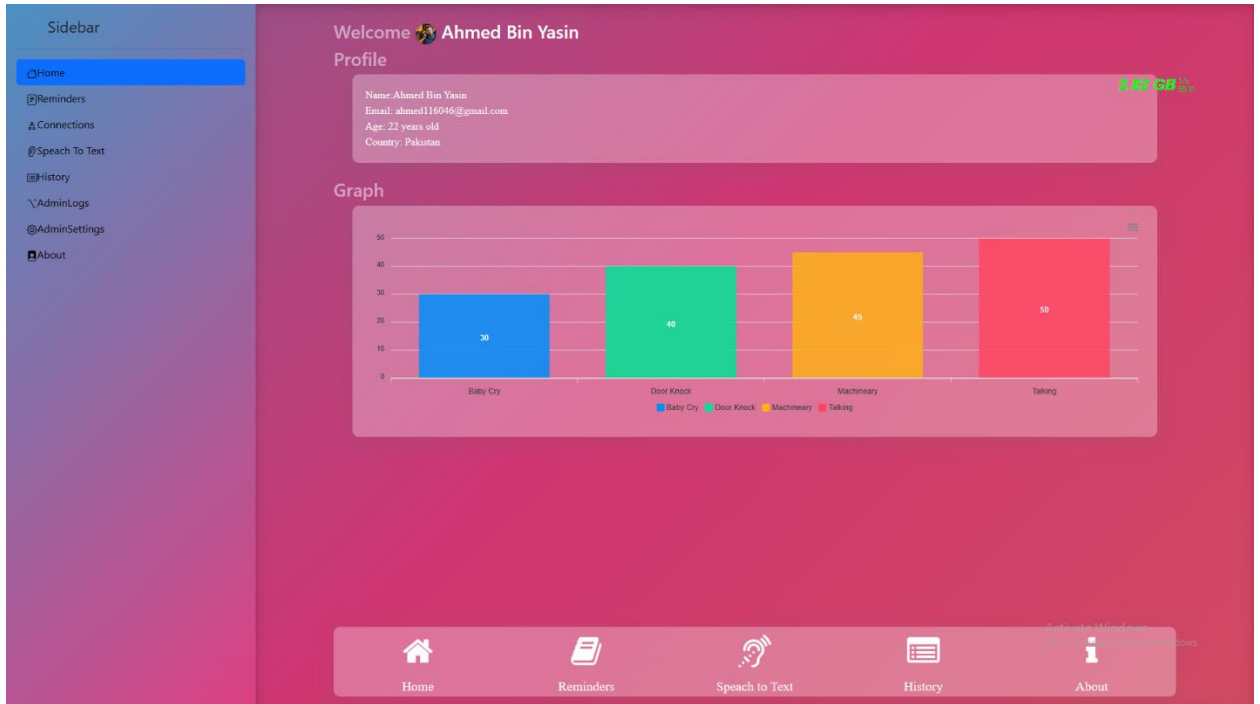
Registration:



The screenshot shows a registration form for 'SOUND SENSE' centered on a teal-to-blue gradient background. The form is white with a thin border and contains the following elements: the title 'SOUND SENSE', a sub-header 'Please enter login User Name and password!', a 'User Name' label with an input field, a 'Password' label with an input field, a 'Reenter Password' label with an input field, a blue 'Login' button, and a link 'Have an account? Login'.

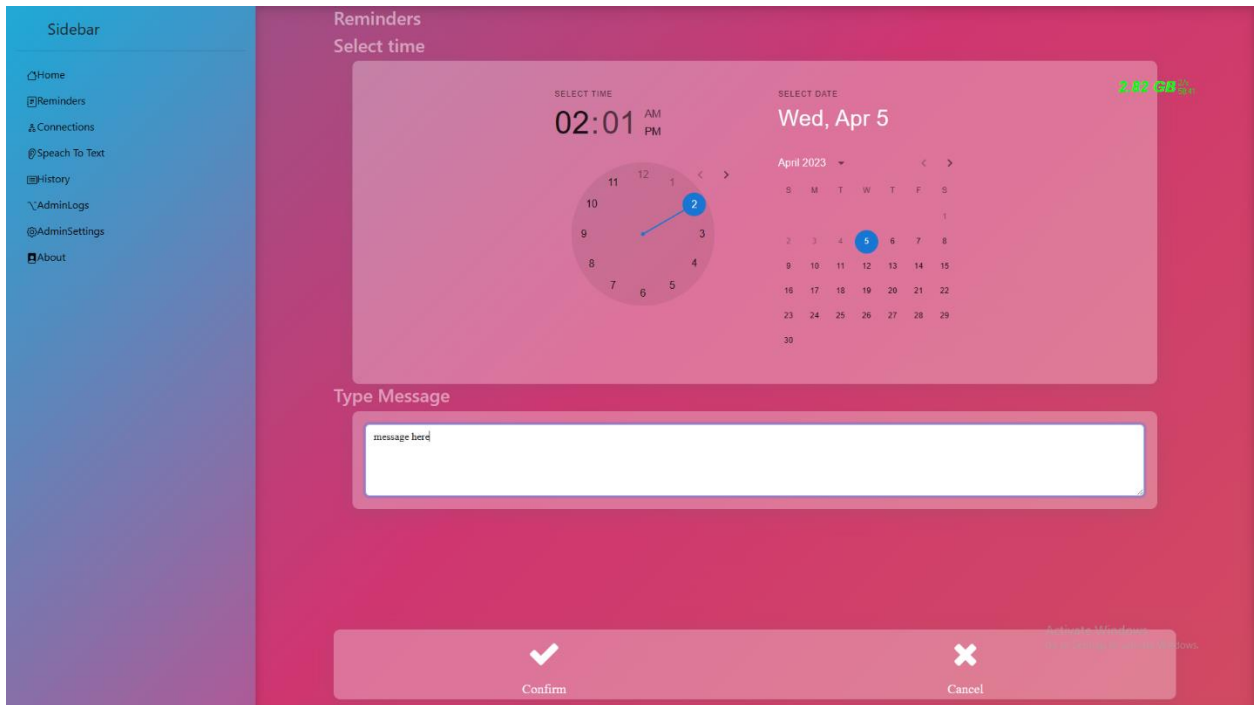
# Sound Sense (Smart Visual Alert System for Deaf)

## Dashboard:

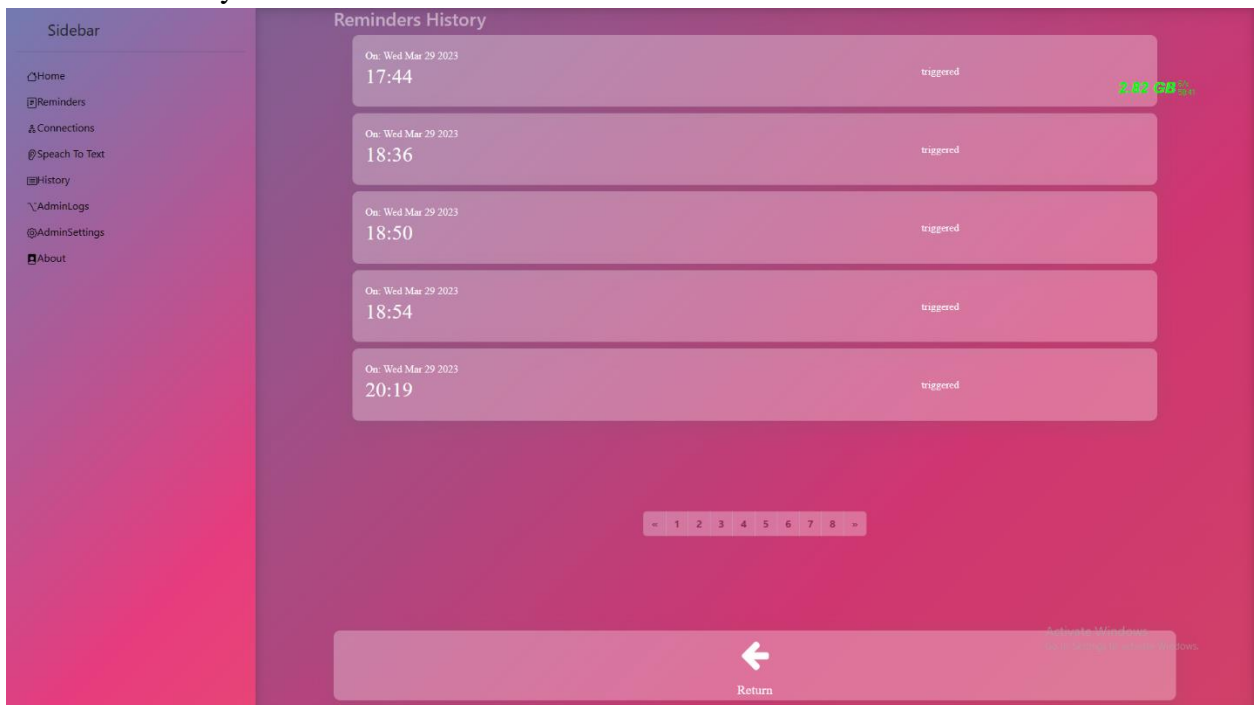


## Reminder Creation:

# Sound Sense (Smart Visual Alert System for Deaf)



## Reminder History:



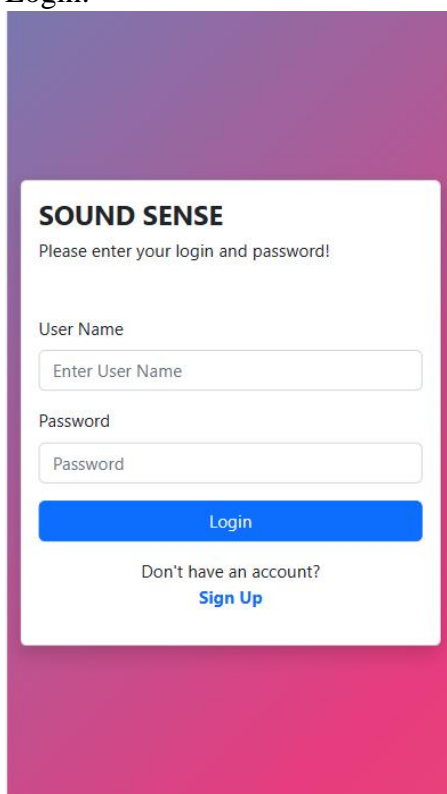
## 6.2.2 Mobile Application

Introducing our versatile and innovative mobile application designed to keep you connected and productive on-the-go. Whether you're running errands, commuting, or traveling, our app makes it easy to manage your tasks, stay organized, and access

## Sound Sense (Smart Visual Alert System for Deaf)

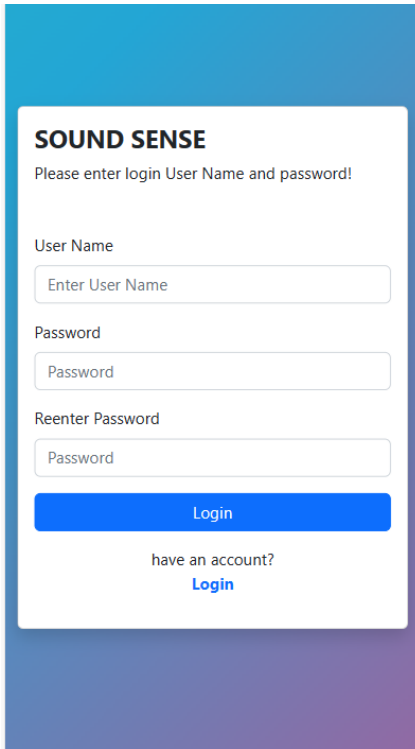
important information from anywhere. With its intuitive interface and powerful features, you can streamline your workflow and accomplish more in less time. Our app is optimized for all mobile devices, ensuring a seamless experience no matter where you are.

Login:

A screenshot of the Sound Sense login interface. The background is a vertical gradient from purple at the top to pink at the bottom. A white rounded rectangle is centered on the screen. At the top of this rectangle, the text "SOUND SENSE" is displayed in bold. Below it, the instruction "Please enter your login and password!" is shown. There are two input fields: "User Name" with a placeholder "Enter User Name" and "Password" with a placeholder "Password". A blue "Login" button is positioned below the password field. At the bottom of the white rectangle, the text "Don't have an account?" is followed by a blue "Sign Up" link.

Registration:

# Sound Sense (Smart Visual Alert System for Deaf)



**SOUND SENSE**  
Please enter login User Name and password!

User Name

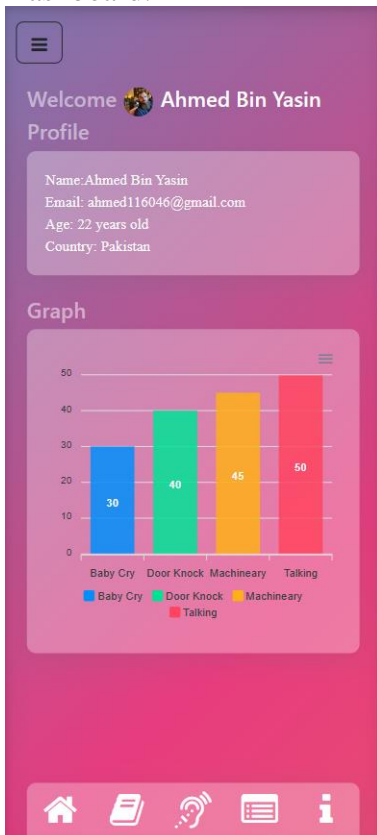
Password

Reenter Password

[Login](#)

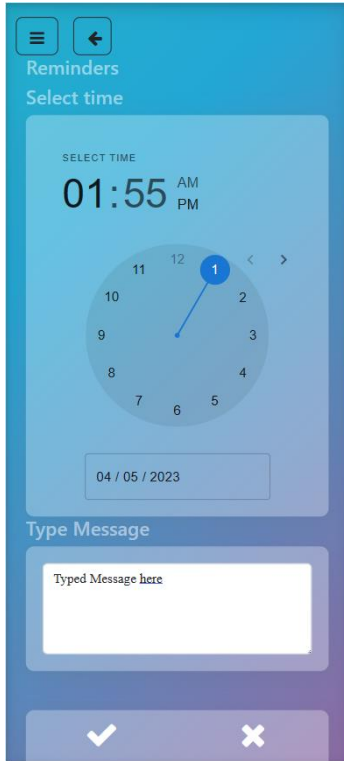
have an account?  
[Login](#)

## Dashboard:

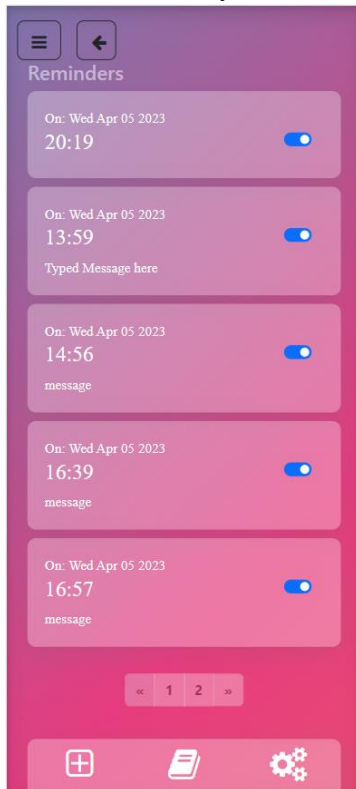


## Reminder Creation:

# Sound Sense (Smart Visual Alert System for Deaf)



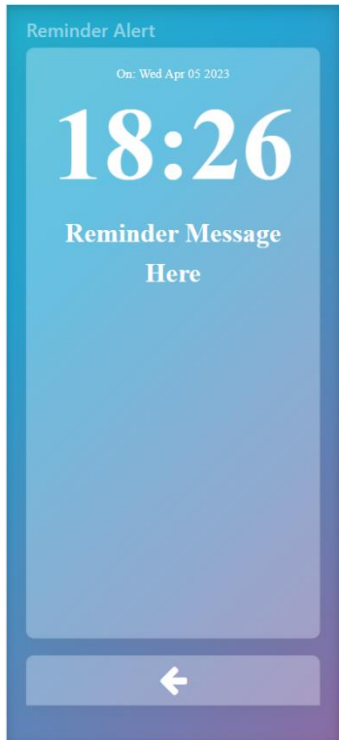
## Reminder History:



## Reminder Alert:



## Sound Sense (Smart Visual Alert System for Deaf)



## 7. Results and Discussion

### 7.1 Overview

In conclusion, this comprehensive document elaborates all facets of the development of Sound Sense – Smart Visual Alert System for Deaf from initial requirement gathering to final testing. Sound Sense is an IOT based sound event classification system that provides us with an effective and accurate sound event classification. Sound Sense is a product of the latest edge computing technology, deep learning, and application development. It has high potential to be further developed into a solution that offers more features and is more accurate and robust.

### 7.2 Objectives achieved.

1. Using deep learning to classify sound events with more than 80% accuracy.
2. Developing Communication between edge device, server and front-end.
3. Provision of Aid to deaf people for their day-to-day life.
4. Exploration of new edge computing technology for AI applications.
5. Development of easy to use and ser friendly mobile application using the latest technologies.

## 8. Future work

There are several potential areas of future work that could be pursued to extend and improve upon the smart visual alert system for deaf people using edge computing and deep learning. Here are a few possibilities:

**Enhancing the accuracy of sound classification:** While the current system uses a deep learning algorithm for sound classification, there is always room for improvement in terms of accuracy. Researchers could explore different deep learning architectures or use additional pre-processing techniques to improve the accuracy of sound classification.

**Adding more sound categories:** The current system is designed to classify sounds into a few categories, such as "Baby Cry", "Door Bell", "Machines" and "Talking". However, there may be other important sounds in the environment that could be added to the system. Researchers could investigate which additional sound categories would be most useful for deaf individuals and develop new models to classify these sounds.

**Improving the LED alert patterns:** While the LED alerts in the current system are customizable, there may be room for improvement in terms of the patterns or colors used to indicate different types of sounds. Researchers could conduct user studies to determine which LED patterns are most effective for different types of sounds and adjust the system accordingly.

**Expanding the mobile app:** While the current mobile app provides some useful features, there is always room for improvement. Researchers could add new features to the app, such as the ability to customize LED patterns or receive alerts through other means (such as vibration or text notifications). Additionally, the app could be expanded to include additional assistive features for deaf individuals, such as real-time transcription or sign language translation.

**Integrating with smart home devices:** As the Internet of Things (IoT) continues to grow, there may be opportunities to integrate the smart visual alert system with other

Sound Sense  
(Smart Visual Alert System for Deaf)

smart home devices. For example, the system could be integrated with smart thermostats, lighting systems, or security cameras to provide additional functionality and improve the overall user experience.