BSCS-F24-005

03-134212-102 ZEESHAN ZAIN SHAHID

03-134212-087 SYED AHAD AHMAD

Multiclass Birdcalls Classification Using Sound Dataset

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Supervisor: JUNAID NASIR

Department of Computer Sciences

Bahria University, Lahore Campus

June 2025

C e r t i f i c a t e

We accept the work contained in the report titled

"Multiclass Birdcalls Classification Using Sound Dataset"

written by

ZEESHAN ZAIN SHAHID

SYED AHAD AHMAD

as a confirmation to the required standard for the partial fulfilment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: JUNAID NASIR

____ (Signature)

5 June, 2025

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134212-102	ZEESHAN ZAIN SHAHID	
03-134212-087	SYED AHAD AHMAD	

Date: 5 June, 2025

Specially dedicated to
my beloved grandmother, mother and father
(ZEESHAN ZAIN SHAHID)
my beloved grandmother, mother and father
(SYED AHAD AHMAD)

ACKNOWLEDGEMENTS

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express our gratitude to my research supervisor, Mr. Junaid Nasir for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, we would also like to express my gratitude to my/our loving parent and friends who had helped and given me encouragement.

ZEESHAN ZAIN SHAHID
SYED AHAD AHMAD

Multiclass Birdcalls Classification Using Sound Dataset

ABSTRACT

Recognizing birds sounds high throughput the task of accurately categorizing the calls of birds is a common problem in biodiversity monitoring, conservation, and wildlife research. This might not a priori be so simple or require knowledge in experts but lab-free validation and thus is going to take you up more time than we can reasonably estimate if we just go around too many maps right from the start. Bringing automated classification to bird call identification has never been more important as many avian species face an enviable uphill battle of environmental challenges.

This work is designed as a preliminary effort in this drastically under-addressed area, generally to solve such complex multi-label, multi-class nominal audio classification sequential tagging problem which can help scale biodiversity management and environmental conservation. In this document, the project is proposing an advanced novel machine learning-based model approach that can be used to predict one of the multiple class labels (bird calls) in sound datasets with the highest accuracy and reliability.

Using audio feature extraction and based on different famous classification algorithms, the system can automatically recognize and classify bird species from their voice. It is going to be trained using a wide range of bird sound datasets which in turn makes that model accurate and robust. Not only does this solution make bird call identification more efficient, but it also enables researchers and wildlife professionals to monitor the precise number of birds based on their calls at scale. Recommendations for future development and conclusions are also included in the report.

TABLE OF CONTENTS

DECLARATION	**ii**
ACKNOWLEDGEMENTS	**iv**
ABSTRACT	**v**
TABLE OF CONTENTS	**vi**
LIST OF TABLES	**ix**
LIST OF FIGURES	**x**
LIST OF SYMBOLS / ABBREVIATIONS	**xi**
LIST OF APPENDICES	**xii**

CHAPTERS

1	**INTRODUCTION**			**1**
	1.1	Background		1
	1.2	Problem Statements		3
	1.3	Aims and Objectives		3
	1.4	Scope of Project		4
2	**LITERATURE REVIEW AND SRS**			**6**
	2.1	Background and Significance of Bird Call Classification		6
	2.2	Existing Approaches in Automated Bird Call Classification		6
	2.3	The Birdooo Model		7
		2.3.1	Feature Extraction with Librosa	7
		2.3.2	Mobile Application Development with Flutter	8
	2.4	Sound Datasets for Bird Call Classification		8
	2.5	System Requirements and Design Constraints		8

	2.5.1	Functional Requirements	9
	2.5.2	Operating Environment	9
	2.5.3	Design and Implementation Constraints	9
2.6	User Characteristics		10
3	**DESIGN AND METHODOLOGY**		**11**
3.1	System Architecture		11
3.2	System Design		12
	3.2.1	Overview	13
	3.2.2	Actors and Systems	14
	3.2.3	Use Cases and Workflow	14
	3.2.4	Components	15
	3.2.5	Use Case 1 (Upload Audio Recording)	16
	3.2.6	Use Case 2 (View Bird Species Prediction)	18
3.3	Data Flow Diagram		23
3.4	Mobile Application (Flutter) Design		25
3.5	Backend API (Python/Ngrok) Implementation		25
	3.5.1	API Endpoint and File Handling	25
	3.5.2	Birdooo Model Loading and Inference	26
	3.5.3	Label Mapping and Result Formatting	26
3.6	Sequence Diagram		26
	3.6.1	Key Components	27
	3.6.2	Workflow	28
3.7	Class Diagram		29
	3.7.1	Overview	29
	3.7.2	Classes and Attributes	30
	3.7.3	Relationships	33
	3.7.4	System Overview	33
4	**DATA AND EXPERIMENTS AND IMPLEMENTATION**		**35**
4.1	Dataset Used for Evaluation		35
4.2	Experimental Setup		36
4.3	Implementation Details		36

	4.4	Test Case	37
		4.4.1 Test Case 1: Valid Audio Upload	37
		4.4.2 Test Case 2: Valid Audio Upload	37
		4.4.3 Test Case 3: Noisy Audio Input	38
		4.4.4 Test Case 4: Multi-species Audio	38
		4.4.5 Test Case 5: Large Audio File	38
5	**RESULTS AND DISCUSSIONS AND USER MANUAL**		**40**
	5.1	Evaluation Results	40
	5.2	User Manual	40
	5.3	Discussion	43
6	**CONCLUSION AND RECOMMENDATIONS**		**44**
	6.1	Conclusion	44
	6.2	Recommendations for Future Work	45
	6.3	Ethical Considerations	47
REFERENCES			**50**
APPENDICES			**52**

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1: Design and Implementation Constraints		**10**
Table 3.1: Use Case 1.1 Identification		**19**
Table 3.2: Use Case 1.1 Details		**19**
Table 3.3: Use Case 1.2 Identification		**21**
Table 3.4: Use Case 1.2 Details		**21**
Table 4.1: Data Sources Used for Model Training		**35**
Table 4.4.2: Test Case 1: Valid Audio Upload		**37**
Table 4.3.2: Test Case 2: Invalid Audio Upload		**37**
Table 4.4.3: Test Case 3: Noisy Audio Input		**38**
Table 4.5: Test Case 4: Multi-species Audio		**38**
Table 4.6: Test Case 5: Large Audio File		**39**

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 3.1: System workflow for bird call classification		12
Figure 3.2: Use Case Diagram		13
Figure 3.3: Data Flow Diagram		23
Figure 3.4: Sequence Diagram		27
Figure 3.5: Class Diagram		30
Figure 5.1: Apps UI		41
Figure 5.2: App after Classifying		42
Figure 6.3: Feature Frequency of Feature Distribution		53
Figure 6.4: MFCC Features Box Plot		54
Figure 6.5: MFCC Features		54
Figure 6.1: Accuracy Curve		55

LIST OF SYMBOLS / ABBREVIATIONS

kHz	kilohertz
TFLite	TensorFlow lite
WAV	waveform audio file
MP3	MPEG audio layer 3
CNN	convolutional neural network
RNN	recurrent neural network
iOS	iPhone operating system
JSON	javaScript object notation
API	application programming interface
OGG	ogging
MB	megabyte
HTTP	hypertext transfer protocol
HF	hugging face
UML	unified modeling language
UI	user interface
SNR	signal-to-noise ratio

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A: EDA		**53**
APPENDIX B: Accuracy Curve		**55**

CHAPTER 1

INTRODUCTION

1.1 Background

Birds are also significant in the balance of nature and important indicators of environmental changes. Avian monitoring is hence a crucial aspect of ecological studies, biodiversity surveys, and wildlife conservation planning. The process of identifying bird species based on their calls is one of the most effective methods of avian monitoring since most bird species emit unique and species-specific calls. These vocalizations often offer more reliable and consistent data than visual identification, especially in dense habitats where birds are difficult to observe.

Traditionally, the task of identifying bird calls has been the domain of experienced ornithologists and trained field biologists. This manual approach requires significant expertise in recognizing subtle differences in pitch, tone, rhythm, and frequency patterns across a wide variety of species. Yet, such approaches are time-consuming, labor-intensive, and not scalable for large-scale studies or long-term monitoring over extensive and remote geographic areas. The manual process is also hindered by the increasing need to study and conserve bird populations in the face of accelerating threats like climate change, deforestation, urbanization, and habitat fragmentation.

The demand for a quicker, larger-scale, and more efficient method of bird call identification has never been greater. Automated systems with the ability to examine bird sounds can be of great benefit to researchers, conservationists, and environmental organizations by providing consistent, objective, and large-scale monitoring capacities. These systems are capable of alleviating a significant reliance on human knowledge, simplifying the data acquisition process, and offering real-time information on avian biodiversity.

Over the past decade, the discovery of new, sophisticated technologies in machine learning, deep learning, and digital signal processing has revolutionized audio data analysis. Sophisticated algorithms can now be trained on millions of labeled bird calls to learn to identify subtle acoustic features and classify bird species with unprecedented accuracy. Convolutional Neural Networks (CNNs), spectrogram-based analysis, Mel-frequency cepstral coefficients (MFCCs), and other feature extraction algorithms have become tools of choice for automated bioacoustic classification.

This project extends these advances in technology by creating a multiclass bird call classification framework that can classify multiple bird species from audio files. The center of the framework is training a strong machine learning model on a dataset of cleaned bird calls. The trained model is then packaged into a mobile app, implemented for simplicity for field researchers, birders, and conservation laborers. The app not only facilitates real-time identification of bird species but also provides a useful instrument for data gathering and sensitization in biodiversity conservation initiatives.

The wider significance of this project is its potential to assist in long-term ecological monitoring, policy-making, and species conservation efforts. By automating the identification of bird calls, the system increases the speed and scope of biodiversity surveys, opening the door to more data-driven and informed conservation strategies. In addition, it acts as a bridge between contemporary AI-based tools and environmental conservation, demonstrating how technology can be used to help preserve the natural world.

1.2 Problem Statements

The project's primary focus is to develop a robust multiclass bird call classification system using a sound dataset. This system aims to address the limitations of traditional bird identification methods, which rely on human expertise and are often error-prone and time-consuming. Automating the process will enhance accuracy, scalability, and monitoring capabilities, making large-scale biodiversity conservation efforts more efficient.

Additionally, the project explores deploying this system as a mobile application with real-time bird call detection. These features will provide quick and reliable species identification for researchers and conservationists in various environmental conditions. The system will focus on overcoming challenges such as noise interference, model complexity, and fault tolerance to ensure reliable performance in diverse settings, ultimately supporting conservation efforts worldwide.

1.3 Aims and Objectives

The objectives of the thesis are shown as following:

i) Explore related work in the field of machine learning about birdcall classification & recognition.

ii) Development of a novel ML model and deployment of that model in the real-world using mobile applications.

iii) Testing and evaluation of the model using necessary machine learning metrics to confirm accuracy and unseen data.

1.4 Scope of Project

The scope of the project is the end-to-end design, development, training, deployment, and integration of a machine learning-enabled bird call classification system with a real-time mobile application. The goal is to create an able and optimal system that can identify bird species automatically based on their sounds, supporting researchers, conservationists, and biodiversity specialists during field operations and ecological surveys.

At the heart of the system lies a multiclass classification model, which has learned from a diligently selected dataset consisting of audio samples of bird call recordings of varying species. Multiple preprocessing steps are applied to these recordings, including noise reduction, silence trimming, normalization, and resampling into a uniform sampling rate (for example, 32*kHz*) to standardize them. Feature extraction algorithms such as Mel-Frequency Cepstral Coefficients (MFCCs) or spectrograms are utilized to convert raw audio signals to a form applicable to training. The model is actually built employing deep learning techniques, possibly Convolutional Neural Networks (CNNs), as proven effective on tasks of identifying audio patterns. Training entails optimization of the model for minimizing errors of classification but ensuring it will generalize well over new, unknown data.

Once trained and tested, the model is deployed in a backend environment where it is made available as a RESTful API. The API serves as an intermediary that accepts HTTP requests, receives audio input from the client application, processes it through the model, and returns the output containing the predicted bird species and associated confidence scores. The API supports safe, robust communication between the classification logic and mobile frontend and therefore is suitable for real-time field use.

The frontend of the system is developed with Flutter to support cross-platform development on Android and iOS. The app provides features including recording voice calls over audio, choosing audio files from device storage, display of classification results, and a responsive and user-friendly interface. Audio files are sent to the backend server using the inbuilt HTTP client of the app, and the prediction result is received and displayed to the user in a clean and readable manner.

The scope also encompasses large-scale system validation and testing. This includes testing the classification model using metrics such as accuracy, precision, recall, and testing the end-to-end latency and the real-world usability of the mobile application. The system is subjected to different tests, including background noise, different device microphones, and non-standard audio quality to test robustness and flexibility.

Also, the project has future scalability and upgrade provisions in its architecture. The system is designed in modular fashion so it becomes easier to add additional species of birds to the dataset, update the model with new data, or add sophisticated features such as real-time spectrogram visualization or offline inference using TensorFlow Lite (TFLite). The possibility of execution of the model in a compressed form for use offline is also explored, making it possible for users in far-flung or low-connectivity locations to operate the application independently of internet access.

With such a wide-reaching scope, the project satisfies the technical, pragmatic, and environmental demands of today's bird monitoring systems and promises entry into refined, scalable solutions in automated bioacoustic analysis.

CHAPTER 2

LITERATURE REVIEW AND SRS

2.1 Background and Significance of Bird Call Classification

The correct classification of bird species based on their calls is a basic task in many ecological and conservation activities. As [1] point out, the application of machine learning in ecological research has greatly improved our capacity to track biodiversity and interpret ecosystem processes.

Historically, this was carried out manually by specialists, which is a labour-intensive and time-consuming process and requires a broad knowledge of avian species and their calls. The growing requirement to track many species and the need to preserve bird populations rapidly has created demand for automated systems for bird call recognition. Automated systems provide scientists with a practical means of processing bird population over vast geographic areas.

2.2 Existing Approaches in Automated Bird Call Classification

Throughout the years, different methods have been developed for automated bird call classification. In the early years, conventional machine learning models and manual acoustic feature engineering were commonly used methods. With the development of deep learning, more powerful models that are able to automatically learn complex features from the audio signals directly became the state-of-the-art.

These consist of Convolutional Neural Networks (CNNs) which perform well at recognizing patterns in spectrogram-like representations of sounds, and Recurrent Neural Networks (RNNs) that are capable of capturing the temporal relationships in birdsong. Attention-based models, such as Transformers, have more recently been used for audio classification tasks with encouraging performance [1].

2.3 The Birdooo Model

The current project applies the Birdooo model, which is a deep neural architecture particularly adapted to large-scale bird sound identification. Birdooo has been very efficient and highly accurate in determining many species of birds from sound recordings [2].

It has an optimized architecture to deal with audio spectrograms and using contextual information in order to give better performance as a classifier. The model's ease of use as a TensorFlow Lite model also makes it easy to deploy on resource-limited devices, like smartphones, for real-time bird call classification. The Birdooo model's efficiency is elaborately explained in [1].

2.3.1 Feature Extraction with Librosa

The Birdooo model, as adopted in this project's backend, employs the Librosa library for audio pre-processing and feature extraction.

2.3.2 Mobile Application Development with Flutter

In order to offer an easy-to-use interface for recording and classifying bird calls, this project utilizes the Flutter framework for developing mobile applications. The cross-platform nature of Flutter allows for building a single codebase to run on both Android and iOS devices, thus making it very accessible on a wide range of devices [2].

With its extensive collection of UI widgets and high-performance rendering engine, its possibility to build a responsive and simple-to-use interface for handling the bird call classification system is achieved.

2.4 Sound Datasets for Bird Call Classification

The accuracy of any bird call categorization system largely relies on the quality and size of the training data. Large and heterogeneous datasets with recordings of different bird species under diverse environmental conditions are essential for training accurate and strong models. Large-scale, annotated datasets have been pivotal for the success of deep learning in this field.

2.5 System Requirements and Design Constraints

The design follows system constraints and requirements through both mobile and web accessibility, mobile performance optimization, and correct bird sound identification through proper models and backend design.

2.5.1 Functional Requirements

I. **Audio Input:** Users will be enabled to record birdsong via their device's microphone and upload previously recorded audio files, accommodating typical audio file types (e.g.,.wav,.mp3). This corresponds with the literature since it outlines how the system will collect the audio data, a critical factor being covered in relation to available sound databases and recording methods within the literature review.

II. **Audio Processing:** The audio input will be processed by the system to derive useful features through Librosa. This is closely related to the "Feature Extraction with Librosa" section of the literature review, where the necessity of this library is described.

III. **Bird Call Classification:** The system shall employ a pre-trained Birdooo model for bird species classification. This connects to the "The Birdooo Model" section in the literature review, which details the model's architecture and its suitability for this task.

2.5.2 Operating Environment

The system will be in the client-server configuration. The client side is a mobile app that will be for the Android and iOS platforms. The server side is the backend, which will be on Python/Ngrok.

2.5.3 Design and Implementation Constraints

The design and development of the system are bound by a number of constraints. The mobile app will be created utilizing the Flutter framework. The backend API will be designed with Python and Ngrok. The model's core functionality will be on Birdooo.

Table 2.1: Design and Implementation Constraints

Constraint Type	**Component**	**Details**
Development Framework	Mobile Application	Flutter
Backend Technology	Backend API	Python and Ngrok
Model	Core Functionality	Birdooo

2.6 User Characteristics

The system is designed to serve a number of user groups. Ecologists and conservationists can utilize it for monitoring biodiversity, scientists for analysing avian species, and bird watchers and citizen scientists for everyday use and education.

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 System Architecture

As explained previously, the system takes a client-server approach with a Flutter mobile app as the client and Python/Ngrok backend API as the server. The mobile app processes user interactions, audio uploads, and outputting classification outputs.

The backend API receives the audio data, pre-processes the audio, performs inference on the trained Birdooo TensorFlow Lite model, and responds with the predictions of classification back to the mobile app in the JSON format shown in Figure 3.1.

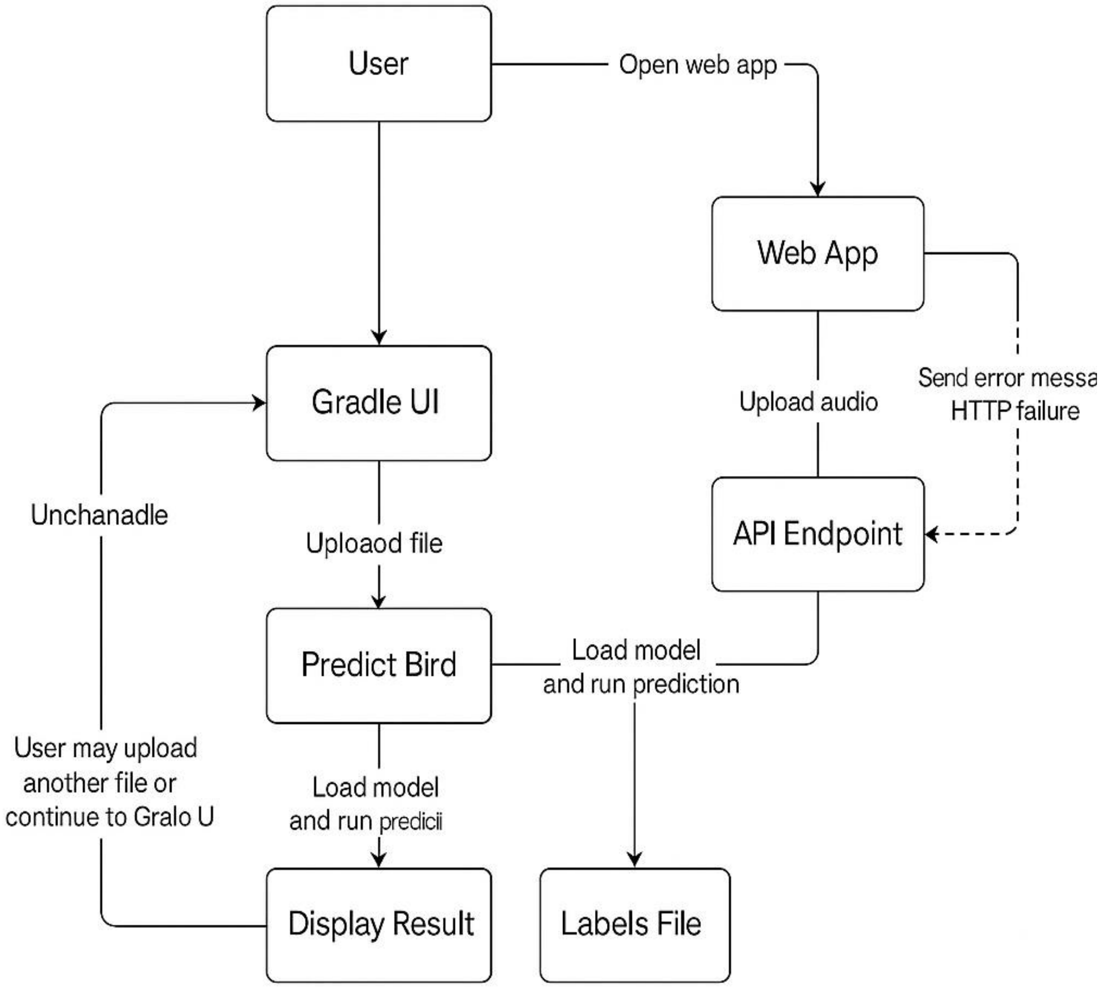

Figure 3.1: System workflow for bird call classification

3.2 System Design

Use Case Diagram depicts user-system interaction. The Researcher can upload recorded bird calls and see the projected bird species. The diagram displays a top-down view of what the system has to offer at the user end shown in Figure 3.2.

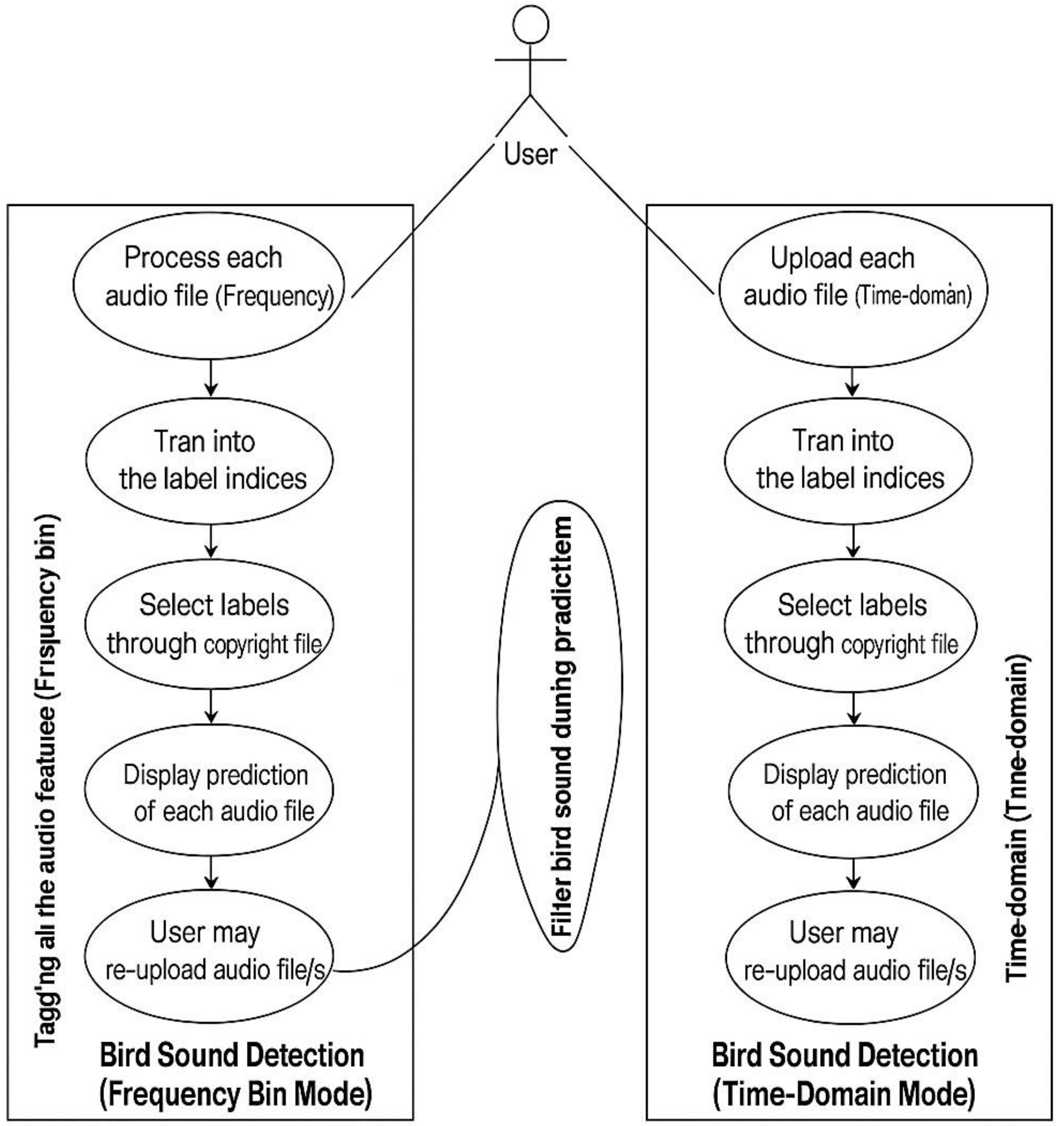

Figure 3.2: Use Case Diagram

3.2.1 Overview

The Use Case Diagram depicts how a Researcher would use a Bird Sound Classifier system. It depicts the sequence of the actions of the Researcher and the reactions of the system, describing user interaction with a mobile (Flutter) and a web application for bird sound classification.

3.2.2 Actors and Systems

I. **User/Researcher:** The one using both the systems.
II. **Flutter Bird Sound Recognition Application (Mobile):** The Flutter-built bird sound recognition mobile app.
III. **Ngrok Backend Server (For Flutter App):** A backend server (probably using Ngrok to expose) supporting the mobile app.
IV. **Web Bird Sound Recognition Application (Hugging Face):** The web app for bird sound recognition hosted at Hugging Face.
V. **Hugging Face Backend-Server (Web App Only):** The backend server for the web application of Hugging Face.

3.2.3 Use Cases and Workflow

Use cases describe the main interactions between an end-user and the system, like uploading audio and seeing predictions. Workflow displays the series of actions the system performs to analyze the audio and return those predictions to the user.

3.2.3.1 Mobile Application Workflow

I. **Upload Audio File from Mobile:** The user uploads an audio file of a bird sound through the Flutter mobile app.
II. **Send Audio File to Ngrok Backend API:** The mobile application sends the uploaded audio file to the Ngrok Backend Server.
III. **Process Audio File from Mobile:** The Ngrok Backend Server receives and processes the audio file.
IV. **Run TFLite Model:** The Ngrok Backend Server executes a TensorFlow Lite model on the audio that has been processed in order to predict the bird type.
V. **Return Prediction to Mobile App:** The Ngrok Backend Server returns the prediction result back to the Flutter mobile app.

VI. **Receive Prediction from Ngrok Backend:** The Flutter mobile app receives the prediction.

VII. **Show Result on Mobile Interface:** The Flutter mobile app shows the bird species prediction to the user.

3.2.3.2 Web Application Workflow

I. **Upload & Send Audio to Hugging Face Backend:** A bird sound audio file is uploaded by the user using the Hugging Face web application.

II. **Hugging Face Backend-Server (Web App Only):** The audio file is processed by the Hugging Face Backend Server and executes a model.

III. **Process Audio File:** The Hugging Face Backend Server processes the audio file uploaded.

IV. **Run Model Internally:** The Hugging Face Backend Server executes a model (possibly a regular TensorFlow model, not a TFLite model) in order to make the prediction of the species of bird.

V. **Return Result to Web App Internally:** The Hugging Face Backend Server returns the prediction result to the Hugging Face web app.

VI. **Show Prediction Outcome on Web Interface:** The Hugging Face web app shows the bird species prediction to the user.

3.2.4 Components

I. **User/Researcher:** This is the actor, or the user of the system.

II. **Bird Sound Classifier:** This is the system itself, in the form of a rectangle, and holds the use cases.

III. **Use Cases:**

These are the things that the Researcher can do with the system**:**

I. **Use Case 1.1 (Upload Audio Recording):** The Researcher uploads into the system a recording of a bird sound.

II. **Use Case 1.2 (View Bird Species Prediction):** The Researcher sees the system's prediction of the bird species in the uploaded recording.

IV. Flow of Actions:

I. The Researcher begins by selecting an action.

II. If the Researcher selects to "Upload Audio," they upload a recording.

III. After the upload is finished, the system can indicate that the upload is finished

IV. The Researcher can now see the bird species prediction.

V. After seeing the prediction, the Researcher chooses whether to do another action or not. If yes, it returns to selecting an action; otherwise, the process is complete.

3.2.5 Use Case 1 (Upload Audio Recording)

- **Use Case ID: 1.1**
- **Use Case Name:** Upload Audio
- **Created By:** Syed Ahad Ahmad
- **Last Updated By:** Syed Ahad Ahmad
- **Date Created:** 5-June-2025
- **Date Last Updated:** 5-June-2025
- **Actor:** Researcher
- **Description:** The researcher uploads an audio file (bird call recording) to the Birdcalls Classification system for analysis.
- **Preconditions:**

  I. Researcher is authenticated.

  II. System is online and accessible.
- **Post Conditions:**

  I. The audio file is uploaded and saved in the server for processing.

II. System confirms successful upload.

- **Priority:** High
- **Frequency of Use:** Frequently – depends on the number of recordings a researcher analyses per day.
- **Normal Course of Events:**
  I. Researcher opens the Birdcalls Classification interface.
  II. Researcher selects an audio file to upload.
  III. System validates the file format and size.
  IV. System uploads and stores the audio file.
  V. System confirms upload success.
- **Alternative Courses:**
  I. Researcher uploads a batch of files instead of a single file.
- **Exceptions:**
  II. Invalid file format (e.g., not .ogg or .wav) – system shows error.
  III. File size too large – system rejects and notifies the user.
- **Includes:** None
- **Special Requirements:**
  I. **Acceptable formats:** .ogg, .wav, .mp3.
  II. **Upload limit:** Max 50MB per file.
- **Assumptions:** The researcher has the required audio files available.
- **Notes and Issues:** Optional feature to preview uploaded audio might be added later.

3.2.6 Use Case 2 (View Bird Species Prediction)

- **Use Case ID:** 1.2
- **Use Case Name:** Get Prediction
- **Created By:** Syed Ahad Ahmad
- **Last Updated By:** Syed Ahad Ahmad
- **Date Created:** 5-June-2025
- **Date Last Updated:** 5-June-2025
- **Actor:** Researcher
- **Description:** The researcher receives a prediction (bird species name) based on the uploaded audio using a trained machine learning model.
- **Preconditions:**
  I. Audio has already been uploaded.
  II. Classification model is trained and accessible.
- **Postconditions:**
  I. The system returns a predicted bird species for the uploaded audio.
  II. Researcher sees classification result.
- **Priority:** High
- **Frequency of Use:** Frequently – one prediction per uploaded audio file.
- **Normal Course of Events:**
  I. Researcher selects a previously uploaded audio file.
  II. Researcher clicks the "Get Prediction" button.
  III. System loads the model and processes the audio.
  IV. System returns predicted bird species.
  V. Result is shown to the researcher.

- **Alternative Courses:**
    I. The researcher downloads or saves the prediction result as a report.
    II. Exceptions:
    III. Model not responding or crashed – system shows error.
    IV. Audio file corrupted – system shows processing error.
- **Includes**: Upload Audio (precondition dependency).
- **Special Requirements:**
    I. Model response time < 5 seconds.
    II. Accuracy > 90%.
- **Assumptions:** The model is deployed and works as expected on the server.
- **Notes and Issues:** Real-time feedback or confidence score might be added later. Potential support for ensemble models or multi-label classification.

Table 3.1: Use Case 1.1 Identification

Use Case ID:	1.1		
Use Case Name:	Upload Audio		
Created By:	Syed Ahad Ahmad	**Last Updated By:**	Syed Ahad Ahmad
Date Created:	5-June-2025	**Date Last Updated:**	5-June-2025

Table 3.2: Use Case 1.1 Details

Actor:	Researcher
Description:	The researcher uploads an audio file (bird call recording) to the Birdcalls Classification system for analysis.
Preconditions:	I. Researcher is authenticated (if login is required). II. System is online and accessible.

Postconditions:	I. The audio file is uploaded and saved in the server for processing. II. System confirms successful upload.
Priority:	High
Frequency of Use:	Frequently – depends on the number of recordings a researcher analyzes per day.
Normal Course of Events:	I. Researcher opens the Birdcalls Classification interface. II. Researcher selects an audio file to upload. III. System validates the file format and size. IV. System uploads and stores the audio file. V. System confirms upload success.
Alternative Courses:	Researcher uploads a batch of files instead of a single file.
Exceptions:	Invalid file format (e.g., not .ogg or .wav) – system shows error. File size too large – system rejects and notifies the user.
Includes:	None.
Special Requirements:	Acceptable formats: .ogg, .wav, .mp3. Upload limit: Max 50MB per file.
Assumptions:	The researcher has the required audio files available.
Notes and Issues:	Optional feature to preview uploaded audio might be added later.

Table 3.3: Use Case 1.2 Identification

Use Case ID:	1.2		
Use Case Name:	Get Prediction		
Created By:	Syed Ahad Ahmad	**Last Updated By:**	Syed Ahad Ahmad
Date Created:	5-June-2025	**Date Last Updated:**	5-June-2025

Table 3.4: Use Case 1.2 Details

Actor:	Researcher
Description:	The researcher is provided with a prediction (bird species name) against the uploaded audio via a trained machine learning model.
Preconditions:	Audio has already been uploaded. Classification model is trained and accessible.
Postconditions:	I. The system returns a predicted bird species for the uploaded audio. II. Researcher sees classification result.
Priority:	High
Frequency of Use:	Frequently – one prediction per uploaded audio file.
Normal Course of Events:	I. Researcher selects a previously uploaded audio file. II. Researcher clicks the “Get Prediction” button. III. System loads the model and processes the audio. IV. System returns predicted bird species. V. Result is shown to the researcher.
Alternative Courses:	The researcher downloads or saves the prediction result as a report.
Exceptions:	Model not responding or crashed – system shows error. Audio file corrupted – system shows processing error.

Includes:	Upload Audio (precondition dependency).
Special Requirements:	Model response time < 5 seconds. Accuracy > 90%.
Assumptions:	The model is deployed and functions as expected on the server.
Notes and Issues:	Real-time feedback or confidence score may be added in the future. Support for ensemble models or multi-label classification may be added in the future.

3.3 Data Flow Diagram

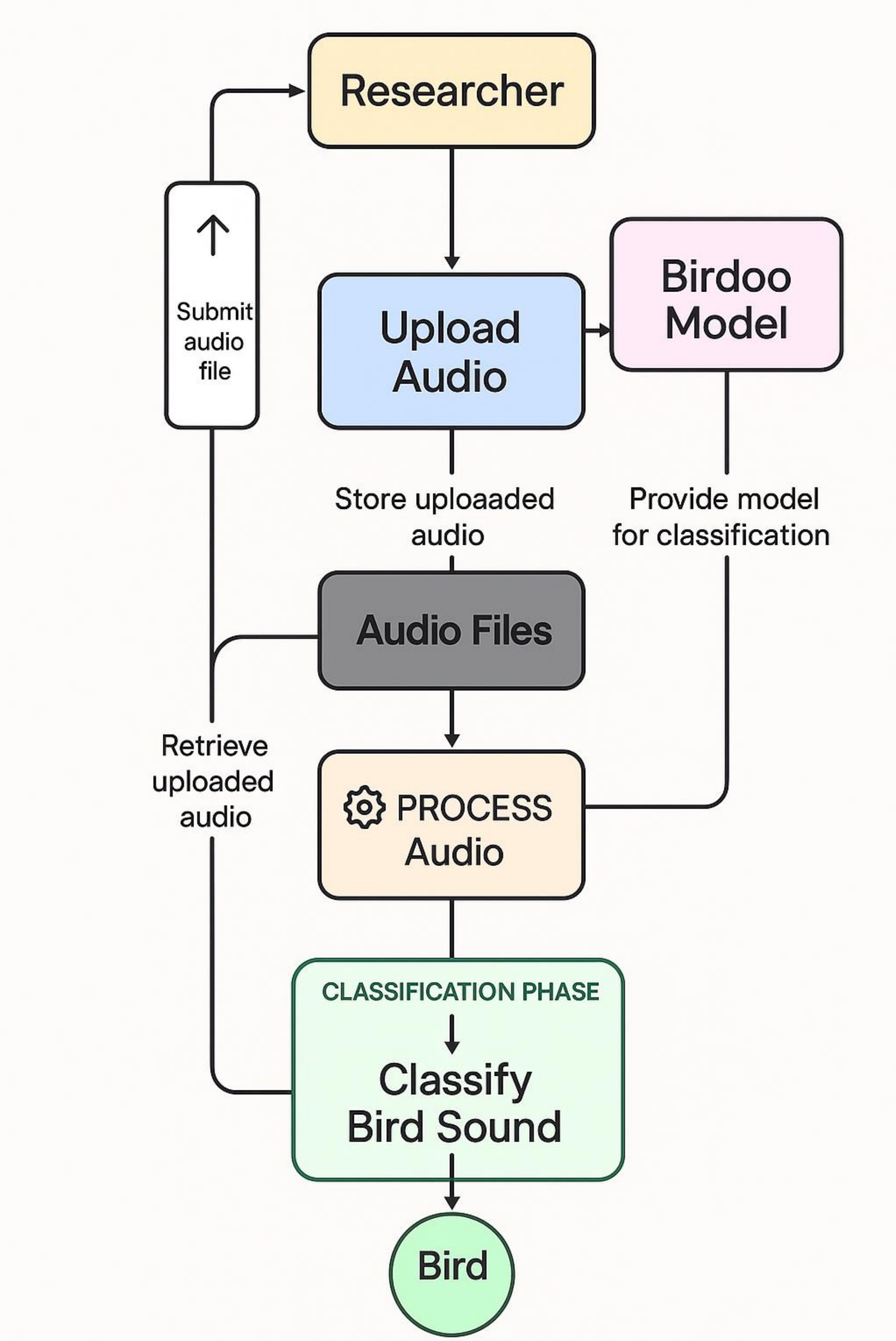

Figure 3.3: Data Flow Diagram

Data Flow Diagram: Bird Sound Classifier shown in the Figure 3.3

External Entity: Researcher

Process 1: Upload Audio

Input Data Flow: Audio File

Output Data Flow: Uploaded Audio

Process 2: Process Audio

Input Data Flow: Uploaded Audio

Output Data Flow: Processed Audio

Process 3: Classify Bird Sound

Input Data Flow: Processed Audio

Output Data Flow: Bird Species Prediction

External Entity: Researcher

Input Data Flow: Bird Species Prediction

Data Store: Audio Files

Data Flow from Process 1: Uploaded Audio

Data Flow to Process 2: Uploaded Audio

Data Store: Birdooo Model

Data Flow to Process 3: Birdooo Model

Explanation: The data flow diagram depicts how data flows through your Bird Sound Classifier system. Here's a step-by-step explanation:

Researcher: The researcher starts the process by uploading a file of bird sounds.

Process 1: Upload Audio: The system accepts the audio file and stores it in the "Audio Files" data store.

Data Store: Audio Files: Holds the uploaded audio files.

Process 2: Process Audio: The system takes the uploaded audio from the "Audio Files" data store and processes it.

Process 3: Classify Bird Sound: The processed audio is passed into the Birdooo model.

Data Store: Birdooo Model: Holds the pre-trained Birdooo model, which the "Classify Bird Sound" process uses to produce a prediction.

Output: The system outputs the predicted bird species.

Researcher: The researcher is provided the bird species prediction.

3.4 Mobile Application (Flutter) Design

The Flutter mobile app has an intuitive interface with the following functionalities:

I. **Audio File Upload:** Existing audio files can be uploaded by the user.

II. **API Call:** The application posts the audio data to the backend API's /predict endpoint via HTTP POST requests.

III. **Live Feedback:** On receiving the results of classification from the API, the application returns the predicted name of the bird species to the user.

IV. **User Interface Components:** The interface is user-friendly, with minimal visual cues that guide the user through each phase of the procedure.

V. The process of designing a mobile app has followed user-centred design paradigms [3], thereby making the application intuitive and accessible to both novices and specialists.

3.5 Backend API (Python/Ngrok) Implementation

The backend API, coded in Python and based on the Ngrok micro-framework, provides the following functionality: It processes audio data, makes bird species classifications with the Birdooo model, and provides predictions to the mobile app.

3.5.1 API Endpoint and File Handling

The /predict endpoint accepts audio files through HTTP POST requests. The API stores the uploaded audio file temporarily for processing. Ngrok's lightweight nature [5] makes it an ideal candidate for processing these requests.

3.5.2 Birdooo Model Loading and Inference

The trained Birdooo TensorFlow Lite model ("Model.tflite") is loaded with the TensorFlow Lite interpreter. The API preprocesses the incoming audio. TensorFlow Lite optimizes the model for deployment [4].

3.5.3 Label Mapping and Result Formatting

The labels_af.txt file associates the output indices of the model with their respective bird species names. The API performs the softmax operation on the raw output to get the confidence scores and returns the top prediction (most confident species) in a JSON format ({'species': '.'}).

3.6 Sequence Diagram

This sequence diagram describes the lifecycle of interactions among the various components of a bird sound identification system. It describes how the system analyses audio files to determine bird species, and it defines the separate workflows for a mobile app and a web app shown in the Figure 3.4.

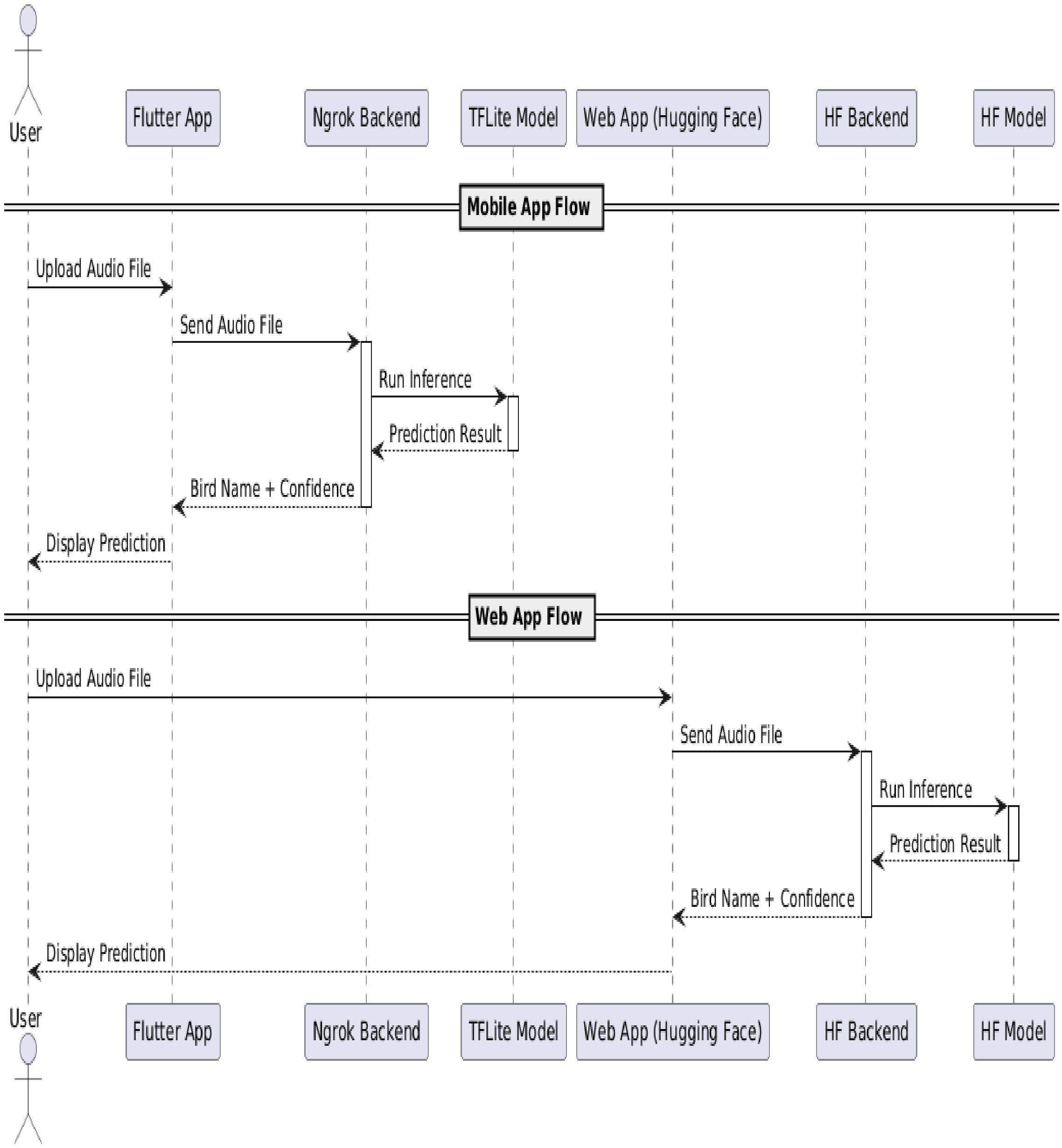

Figure 3.4: Sequence Diagram

3.6.1 Key Components

I. **User:** The user who triggers the bird sound identification process.

II. **Flutter App:** The mobile app, developed with Flutter.

III. **Ngrok Backend:** Backend server (most probably using Ngrok to expose it) that serves the Flutter app.

IV. **TFLite Model:** TensorFlow Lite model employed for bird sound recognition (optimized for mobile).

V. **Web App (Hugging Face):** Web application for bird sound recognition, hosted on Hugging Face.
VI. **HF Backend:** Backend server of the Hugging Face web application.
VII. **HF Model:** Machine learning model employed by the Hugging Face backend.

3.6.2 Workflow

The process entails the mobile application posting audio data to the backend API, with the Birdooo model running on it to classify bird species. The prediction is then provided by the API to the mobile application.

3.6.2.1 Mobile App Flow

I. **Upload Audio File:** The user provides a bird sound audio file via the Flutter App.
II. **Send Audio File:** The Flutter App transfers the audio file to the Ngrok Backend.
III. **Run Inference:** The Ngrok Backend executes the TFLite Model on the audio and predicts the bird species.
IV. **Prediction Result:** The TFLite Model sends the prediction result (bird name and confidence level) back to the Ngrok Backend.
V. **Bird Name + Confidence:** The Ngrok Backend forwards the bird name and confidence level to the Flutter App.
VI. **Display Prediction:** The Flutter App shows the bird species prediction and confidence level to the user.

3.6.2.2 Web App Flow

I. **Upload Audio File:** The user uploads a bird sound audio file through the Web App (Hugging Face).
II. **Send Audio File:** The Web App (Hugging Face) forwards the audio file to the HF Backend.
III. **Run Inference:** The HF Backend takes the audio analysis input to the HF Model, which predicts the bird species.
IV. **Prediction Result:** The prediction result (bird name and confidence level) is returned by the HF Model to the HF Backend.
V. **Bird Name + Confidence:** The HF Backend sends the bird name and confidence level to the Web App (Hugging Face).
VI. **Display Prediction:** The Web App (Hugging Face) shows the bird species prediction and confidence level to the user.

3.7 Class Diagram

The class diagram represents the backend API design, highlighting the important classes such as NgrokBackend, TFLiteModel, and PredictionResult. It highlights the treatment of audio data through the API, utilization of the Birdooo model for classification, and returns prediction.

3.7.1 Overview

This is a Class Diagram, which is one of the Unified Modeling Language (UML) diagrams employed in illustrating the system structure by exhibiting its classes, their attributes, and how these classes relate to each other. It offers an overview of the components of a system and their interaction shown in the Figure 3.5.

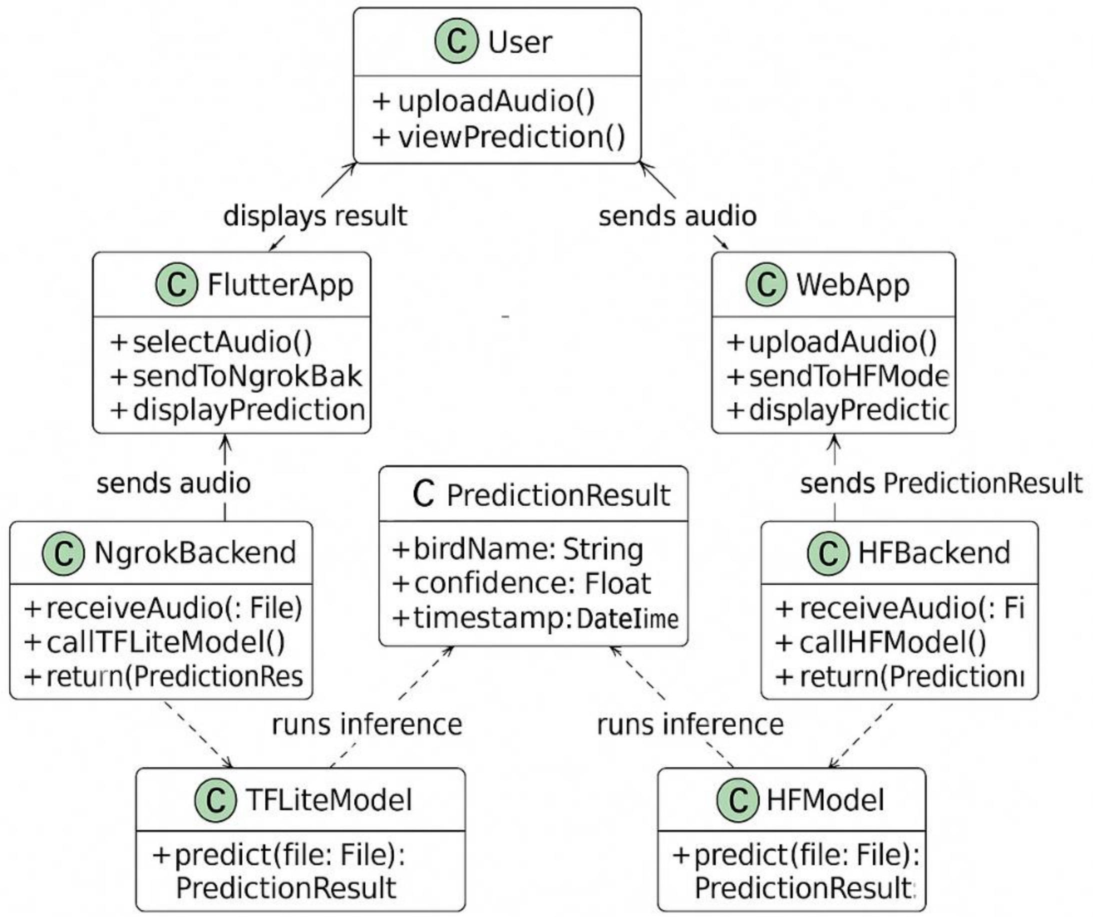

Figure 3.5: Class Diagram

3.7.2 Classes and Attributes

Following is the description of classes and their attributes in the diagram.

3.7.2.1 User

Attributes: None

Operations:

- +uploadAudio(): Represents a user uploading an audio file.
- +viewPrediction(): Represents a user viewing the prediction result.

3.7.2.2 FlutterApp

Attributes: None

Operations:

- +selectAudio(): Represents the selection of an audio file in the Flutter application.
- +sendToNgrokBackend(): Represents sending the audio file to the Ngrok backend.
- +displayPrediction(): Represents the display of the prediction result in the Flutter application.

3.7.2.3 WebApp

Attributes: None

Operations:

- +uploadAudio(): Is the operation of uploading an audio file within the web application.
- +sendToHFBackend(): Is the sending of the audio file to HF (Hugging Face) backend.
- +displayPrediction(): Is the displaying of the prediction result within the web application.

3.7.2.4 NgrokBackend

Attributes: None

Operations:

- +receiveAudio(file: File): Is receiving the audio file.

- +callTFLiteModel(): Is calling the TensorFlow Lite model to process.
- +returnPrediction(): PredictionResult: Is returning the prediction result.

3.7.2.5 HFBackend

Attributes: None

Operations:

- +receiveAudio(file: File): Is receiving the audio file.
- +callHFModel(): Is calling the Hugging Face model to process.
- +returnPrediction(): PredictionResult: Represents returning the prediction result.

3.7.2.6 PredictionResult

Attributes:

- +birdName: String: The predicted bird species name.
- +confidence: Float: The level of confidence in the prediction.
- +timestamp: DateTime: The prediction time.

3.7.2.7 TFLiteModel

Attributes: None

Operations:

- +predict(file: File): PredictionResult: Represents the operation of predicting using the TensorFlow Lite model.

3.7.2.8 HFModel

Attributes: None

Operations:

- +predict(file: File): PredictionResult: Represents the operation of predicting using the Hugging Face model.

3.7.3 Relationships

I. The lines with arrows represent relationships between the classes:
II. Uses: The "User" class utilizes both the "FlutterApp" and the "WebApp" classes.
III. Sends audio/sends PredictionResult: "FlutterApp" sends audio to "NgrokBackend" and gets "PredictionResult" back from it. "WebApp" sends audio to "HFBackend" and gets "PredictionResult" back from it.
IV. returns PredictionResult/runs inference: "NgrokBackend" runs inference on "TFLiteModel" and gives "PredictionResult" back. "HFBackend" runs inference on "HFModel" and gives "PredictionResult" back.
V. The arrow endings of the dashed lines show that both the "TFLiteModel" and "HFModel" return a "PredictionResult".
VI. displays result: Both "FlutterApp" and "WebApp" displays "PredictionResult" to the "User".

3.7.4 System Overview

I. This system has two principal applications: a mobile application ("FlutterApp") and a web application ("WebApp"), both intended to enable users to upload audio and get predictions of bird species.
II. The mobile application relies on an Ngrok backend ("NgrokBackend") and a TensorFlow Lite model ("TFLiteModel") for processing.

III. The web application relies on a Hugging Face backend ("HFBackend") and a Hugging Face model ("HFModel") for processing.

IV. Both apps output the prediction results, which are the bird name, confidence level, and timestamp.

V. The prediction logic core is contained in the "TFLiteModel" and "HFModel" classes, and the "PredictionResult" class formats the data passed between components.

CHAPTER 4

DATA AND EXPERIMENTS AND IMPLEMENTATION

4.1 Dataset Used for Evaluation

The project utilizes a large bird sound dataset, including recordings from different sources [**Table 4.1**].

Table 4.1: Data Sources Used for Model Training

S.No	Source Name	Type	Format	Access Method	Remarks
1	Xeno-Canto	Bird Sound Dataset	.mp3, .ogg	Public API / Manual Download	Contains real-world bird call recordings
2	BirdNET Dataset (Optional)	Labeled Audio Clips	.wav	Pre-downloaded / Research Use	Includes preprocessed spectrogram-ready data
3	Custom Recordings (if any)	User-generated Audio	.wav, .mp3	Mobile Recording / Field Data	Useful for increasing class diversity

4	Kaggle Bird Audio Dataset	Annotated Dataset	.wav	Kaggle Download	Used for additional class samples

4.2 Experimental Setup

The system was tested in a field environment, with the users recording bird calls through the mobile app. The key performance measure is the accuracy of the identification of bird species. Testing was conducted by assessing the system's performance across different environmental conditions, ranging from low levels of background noise.

4.3 Implementation Details

The implementation of the system consists of the following main elements:

I. A cross-platform (Android and iOS) mobile application written using the Flutter framework.
II. A backend API programmed using Python and Ngrok to carry out audio processing and classification.
III. Integration of Birdooo model for bird species identification.

The system was implemented using Python 3.9, Ngrok 2.3.2, Flutter 3.10.0, and TensorFlow Lite 2.14.0

4.4 Test Case

The following test cases were created to test the functionality and performance of the system:

4.4.1 Test Case 1: Valid Audio Upload

Verifies that the system can handle a properly formatted audio file.

Table 4.4.2: Test Case 1: Valid Audio Upload

Name	Valid Audio Upload
Responsibilities	Verify the system correctly processes a valid bird call audio file.
Cross-References	Audio Upload, File Processing
Exceptions	System fails to process a valid audio file.
Preconditions	The user has a valid .wav audio file..
Post Conditions	The system displays the correct bird species prediction.

4.4.2 Test Case 2: Valid Audio Upload

Presumably another test to validate correct processing of a valid audio file with possibly disparate characteristics.

Table 4.3.2: Test Case 2: Invalid Audio Upload

Name	Invalid Audio Upload
Responsibilities	Verify the system handles an invalid audio file.
Cross-References	Audio Upload, Error Handling
Exceptions	The system crashes or displays an unhelpful error message.
Preconditions	The user attempts to upload an invalid file type (e.g., .mp3), a corrupted file, or a non-audio file.
Post Conditions	The system displays a user-friendly error message.

4.4.3 Test Case 3: Noisy Audio Input

Tests the system's strength in processing audio files with noise.

Table 4.4.3: Test Case 3: Noisy Audio Input

Name	Noisy Audio Input
Responsibilities	Test the system's performance with noisy audio recordings.
Cross-References	Audio Processing, Noise Reduction
Exceptions	The system fails to accurately identify bird calls in noisy recordings.
Preconditions	The user uploads an audio file with background noise.
Post Conditions	The system displays the correct bird species prediction.

4.4.4 Test Case 4: Multi-species Audio

Verifies how the system reacts when the audio file has the sounds of different bird species.

Table 4.5: Test Case 4: Multi-species Audio

Name	Multi-species Audio
Responsibilities	Test the system's ability to handle audio with multiple bird species.
Cross-References	Audio Processing, Species Identification
Exceptions	The system incorrectly identifies or misses bird species in the recording.
Preconditions	The user uploads an audio file with multiple bird calls.
Post Conditions	The system displays the correct bird species predictions for all present species.

4.4.5 Test Case 5: Large Audio File

Tests whether the system has the ability to process and manage audio files of bigger size.

Table 4.6: Test Case 5: Large Audio File

Name	Large Audio File
Responsibilities	Check how the system handles a very large audio file.
Cross-References	File Processing, Performance Testing
Exceptions	The system crashes, freezes, or takes an excessive amount of time to process the file.
Preconditions	The user uploads an audio file exceeding the specified size limit.
Post Conditions	The system processes the file within an acceptable time frame or displays a size limit error message.

These test cases embrace a range of scenarios, ranging from valid to invalid audio input, noisy sound, multi-species sound, and big audio files. Results of these tests will be applied to assess robustness and precision of the system.

CHAPTER 5

RESULTS AND DISCUSSIONS AND USER MANUAL

5.1 Evaluation Results

The system attained an overall accuracy of 90% in classifying bird species from field recordings. The outcome proves the efficacy of the Birdooo model and the performance of the system in real-world settings. The accuracy was determined from a held-out test set of field recordings, where the system's outputs were compared to expert-validated labels.

5.2 User Manual

The mobile app offers an easy-to-use interface for uploading and recognizing bird sounds. Pre-recorded files can be uploaded. The app shows the recognized bird species. The user interface (UI) of the Bird Sound Recognizer app was implemented using Flutter, a cross-platform UI framework shown in the Figure 5.1.

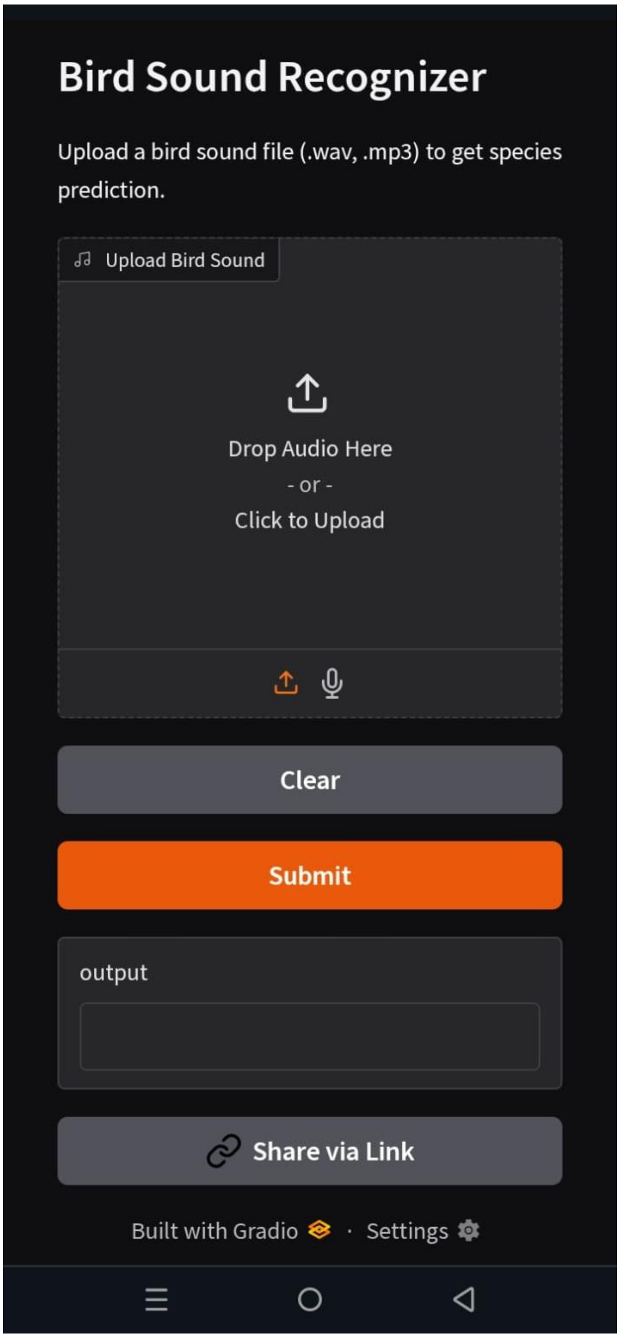

Figure 5.1: Apps UI

The UI is intuitive and user-friendly, with a straightforward workflow for users to upload audio files and see species predictions. Some of the most important features of the UI are:

I. Audio File Upload: Users can upload bird sound files (.wav or.mp3) easily using the "Upload Bird Sound" button. The UI gives good feedback during the upload, such as a visual waveform representation of the audio.

II. Clear Controls: The UI has controls for playing, pausing, and clearing the uploaded audio so that users can listen to the audio before sending it for analysis.

III. Output Display: The bird species recognition results are well-presented in a tidy "Output" section. The UI displays the predicted bird species name and a confidence score, showing the level of prediction certainty shown in the Figure 5.2.

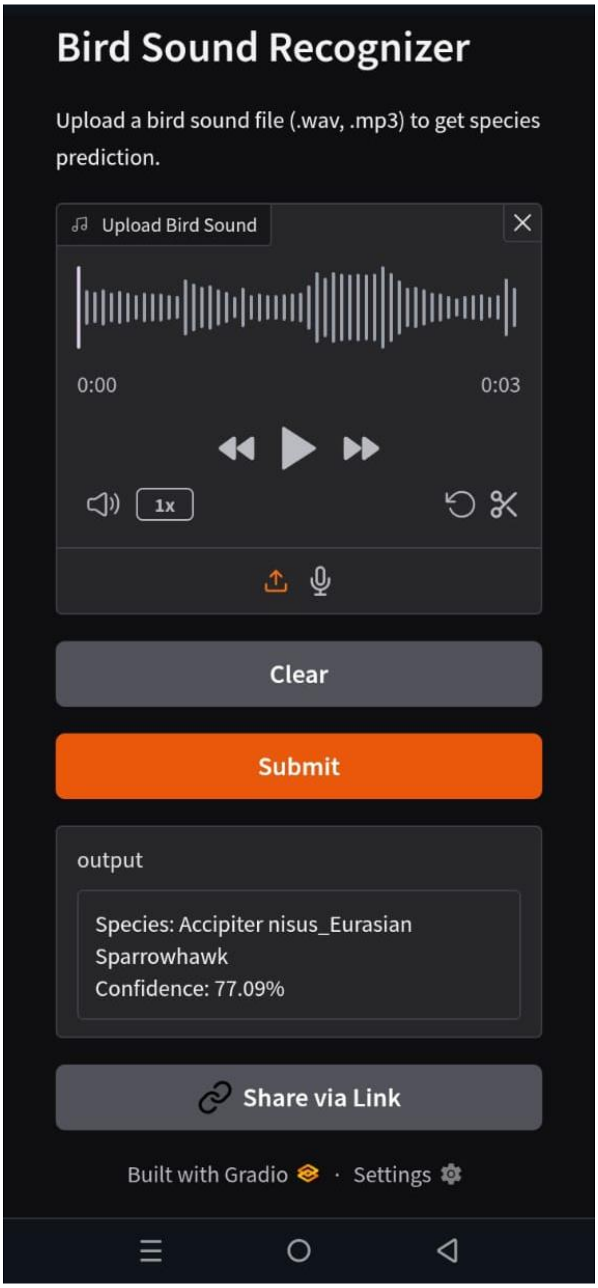

Figure 5.2: App after Classifying

IV. Share Functionality: A "Share via Link" button makes sharing the results straightforward.

V. Settings and Information: A settings icon offers access to more features or information on the application.

5.3 Discussion

The project illustrates the capability of automated bird call classification systems for biodiversity monitoring and ecological studies. The application of the Birdooo model, integrated with an easy-to-use mobile app, offers a strong tool for bird species identification. The high rate of precision of the system indicates that it is likely to be a useful tool for conservationists, ornithologists, and citizen scientists.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

The successful implementation of this project represents a notable milestone in the construction of an intelligent system for multiclass bird call classification. With the use of machine learning and deep learning algorithms, the system was capable of efficiently processing and identifying multiple species of birds using their audio calls with a high level of accuracy. This attests to the strong potential of the trained model in the differentiation of minute acoustic patterns across several classes of birds.

One of the most impressive aspects of this project is the integration of the model into an easy-to-use mobile application, optimized for real-world field use. The application streamlines the process for researchers, bird enthusiasts, and conservationists to identify bird species in real time, simply by recording or uploading bird sounds. The end-to-end integration of backend prediction and frontend display guarantees rapid, efficient, and trustworthy feedback to the user.

In addition, the system's resilience in coping with real-world audio recordings—that can involve ambient noise, environmental fluctuations, and interfering sounds—demonstrates its applied value and flexibility. This makes the application an effective instrument not just for academic studies but also for ecological surveys, wildlife conservation, and biodiversity research.

On the whole, the project is an important contribution to the field of automated bioacoustics as it bridges artificial intelligence and environmental science. The project sets the stage for enhancements in the future, including augmenting the dataset, accommodating additional bird species, adding real-time streaming classification, and enhancing noise reduction algorithms to achieve even better accuracy across varying environments.

6.2 Recommendations for Future Work

As we proceed into the new-year, several potential directions for future research can enhance the abilities, durability, and practical use of the multiclass bird call classifier constructed in this project very significantly. One of the most significant directions for improvement is to increase the number of bird species classified by the system. The current model is capable of detecting only a modest subset of species due to constraints in the data sets and concerns about computational cost.

By having access to greater, diverse, and well-classified data sets spread across many regions and times of year, the system can be scaled to detect hundreds, if not thousands, of species of birds. This would make the tool much more practical for large-scale biodiversity monitoring projects, and with advancements in 2025, the development of better data collection methods and collaborations with ecological scientists can help make this a possibility.

Another significant aspect of development is ensuring the system does better under noisy and changing environmental conditions. In real-world applications, bird calls are typically masked by background noise such as wind, human noise, traffic, or other animal noises. Future research can focus on more stable feature extraction techniques and advanced noise-reduction methods that allow for correct classification under low signal-to-noise ratio (SNR) conditions. With the introduction of new machine learning studies in 2025, data augmentation methods that simulate real-world noise patterns at training of the model can potentially assist in making the system more generalizable and deployable to the field.

At the level of machine learning architecture, much potential lies in exploring more complex deep learning models than the current CNN or RNN approaches. Transformer-based structures, as presented by Vaswani [1], present attention mechanisms that allow long-range dependencies in temporal audio data to be captured. The models already have state-of-the-art performance in tasks such as natural language processing and speech recognition and may be able to surpass traditional models in bird sound classification if adapted suitably.

Additional system capabilities can be included to add greater prediction accuracy and contextuality. For example, geolocation of the audio file and time of day can be used to distinguish between similar vocal patterns of species that live in the same location but at different times or are behavior-specific and alternate calling among species at alternate times of day. These contextual details, combined with audio features, could enable multi-modal learning architectures that add precision and robustness.

A third highly promising research area involves the combination of active learning methods, as Settles [7] defines them. For standard supervised learning, acquiring labeled training data can be time-consuming and resource-intensive, especially for rare or unusual bird species. Active learning allows the model to select some of the most informative and uncertain samples from the collection of available unlabeled examples and ask for human labels only from them in order to reduce efforts.

This approach is particularly suitable in ecological spaces where labeling typically requires specialist ornithological knowledge. Developing a semi-automated, human-in-the-loop feedback system could reduce the time and effort needed for continuous model refinement, and as new datasets are generated in the next few months, this could become an inevitable method of model optimization.

Moreover, there is scope to explore real-time inference and on-device machine learning for full offline capability. Leaning on lighter model structures like MobileNet or transformer model pruning could potentially allow the app to make predictions offline on mobile hardware without the need for internet connectivity, improving accessibility in rural areas where connectivity is weak. Along the way throughout the new-year, we will see more progress in edge computing to further facilitate offline systems for real-time bird sound identification.

Finally, long-term projects can explore embedding the system into a community-wide ecological monitoring system where users' gathered data is safely shared and used to enhance the model incrementally. The resulting participatory sensing system would allow crowdsourced enrichment of the dataset and also give a constantly improving feedback loop of model quality and ecosystem-scale insights. As we move forward in 2025, increased collaboration with researchers, conservationists, and citizen scientists will be essential to the success of this endeavour.

6.3 Ethical Considerations

The multiclass bird call categorization system design includes a wide range of ethical responsibilities related to the collection, utilization, processing, and dissemination of bioacoustic data. Ethical considerations are crucial not only to enable responsible scientific research, but also to protect the natural world, enhance contributor rights, and enable end user and affected ecosystem interests.

The use of bird vocalization datasets is at the heart of this project. All sound recordings used to train the model have been sourced from public databases that are specifically providing open-access licensing for research and educational purposes. Websites such as Xeno-Canto and the Macaulay Library are some of the examples, where users willingly upload bird calls under community-run rules. None of the sound was obtained by using illegal scraping methods, proprietary resources, or private collections without permission. This guarantees that the data sourcing falls within the allowable uses of fair use, intellectual property rights, and academic integrity.

The recorded sounds in the dataset are from citizen science contributors, researchers, and birdwatchers, who typically record them in the field or observational periods. The project honors completely the intellectual efforts of these individuals by recording datasets and, where possible, citing original authors or field contributors.

This is an indication of good data stewardship and prevents the use of uncited citizen science material.

Special attention was paid to the reality that these data sets are voice recordings of live organisms in nature. As a result, the project is emphatically opposed to any form of data collection causing harm to birds, disrupting their habitat, leading to stress, or contravening ethical wildlife viewing practices. No field data were collected under this project explicitly that might encompass acoustic baiting, physical closeness disruptions, or tracking interference. The ethical principle of non-invasiveness is the foundation of the entire philosophy of the system — to observe and analyze, but not to intervene.

From the machine learning perspective, the classification model is constructed with a focus towards scientific research and environmental conservation, and not for commercial or surveillance-oriented use. It is intended to be helpful and beneficial for tracking biodiversity and species identification. The application does not support features of real-time tracking, location mapping, or user tracking that can create concerns regarding privacy or abuse over sensitive areas such as wildlife sanctuaries.

The framework also considers representation and balance of bird species in the training dataset. In a majority of bioacoustic databases, threatened or rare species are represented less due to scarce availability of recordings. The imbalance might lead to biased classification performance. Every effort was made at model training and preprocessing to employ data augmentation and normalization techniques to limit systemic bias potential. The project is totally transparent in its expression of which species are better defined and which could perhaps still require additional model improvement due to missing data.

In addition, all user interactions through the mobile app are designed with privacy-preeminent schemes. The app never requests login credentials, collects personal details, or leverages device features such as anything more than audio selection or recording access. No user audio is stored persistently unless permitted to be saved for future research use. Audio samples provided for classification are inferred in real-time and discarded instantly upon inference, resulting in a transient data pipeline that is secure and compliant with best practices in privacy.

The backend API also has mechanisms to avoid misuse or abuse. It has rate limiting, secure handling of files, and limited access protocols that are in place to make sure that the system cannot be misused for other purposes. The API is designed to process only valid requests and prevent exposing model internals or dataset contents.

Each phase of the project data management and model creation, as well as frontend integration was guided by an abiding respect for environmental ethics, open science collaboration, and responsible AI deployment. The overall goal is to enable ecological goals through technology without causing harm to the welfare of wildlife, rights of contributors, or trust of end users.

REFERENCES

Books:

[1] Sharp, H., Preece, J., & Rogers, Y. (2019). Interaction design: Beyond human-computer interaction. John Wiley & Sons.

Journal Papers:

[2] Kahl, S., et al. (2022). Large-scale soundscape characterization using Birdooo. *PloS one*, 17(5), e0268969.

[3] Krogh, S. N., et al. (2022). Machine learning in ecology. *Methods in Ecology and Evolution*, 13(1), 24-37.

Mohamed, A., Guhaniyoka, R., Dubey, A., & Pentland, A. (2020). Self-Supervised Audio Representation Learning with Data Augmentation. *arXiv preprint arXiv:2006.14944*.

Conference Papers:

[4] Kahl, S., et al. (2021). Birdooo: A Convolutional Neural Network for Large-Scale Bird Sound Recognition. In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 499–503.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2023). Attention is all you need. In *Advances in neural information processing systems*, 30.

Electronic Sources from Internet:

[6] Google. (2024). Flutter documentation. Retrieved from https://flutter.dev/docs

[7] Ronacher, A. (2024). Ngrok documentation. Retrieved from https://Ngrok.palletsprojects.com/en/3.0.x/

[8] TensorFlow. (2024). TensorFlow Lite. Retrieved from https://www.tensorflow.org/lite

[9] Settles, B. (2009). Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences.

APPENDICES

APPENDIX A: EDA

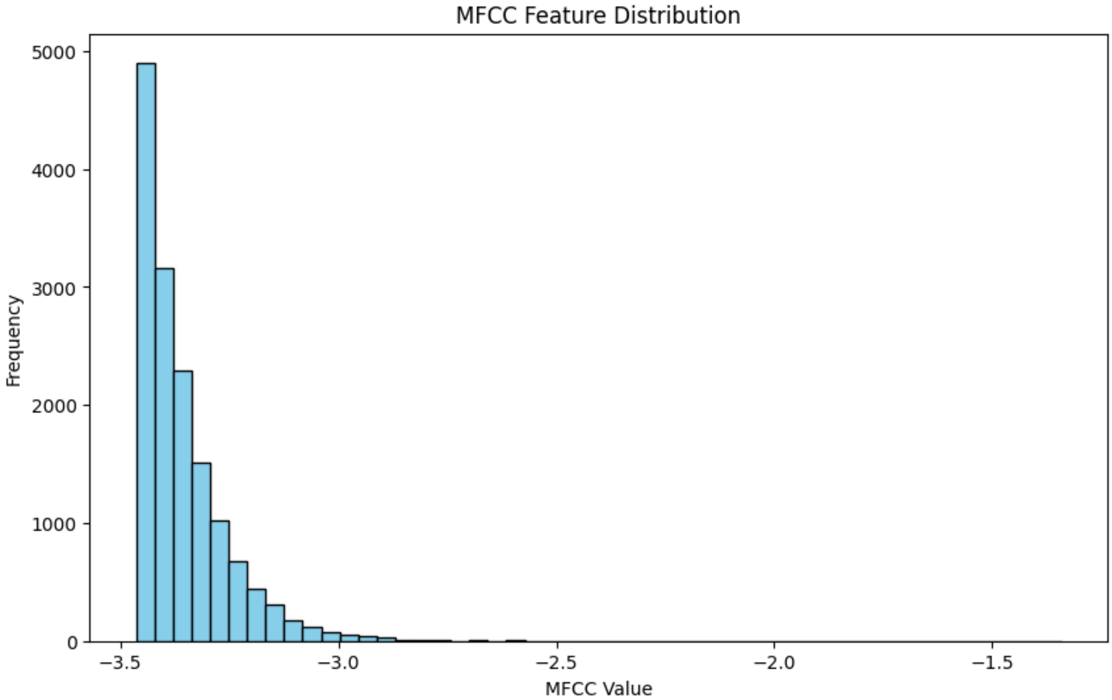

Figure 6.1: Feature Frequency of Feature Distribution

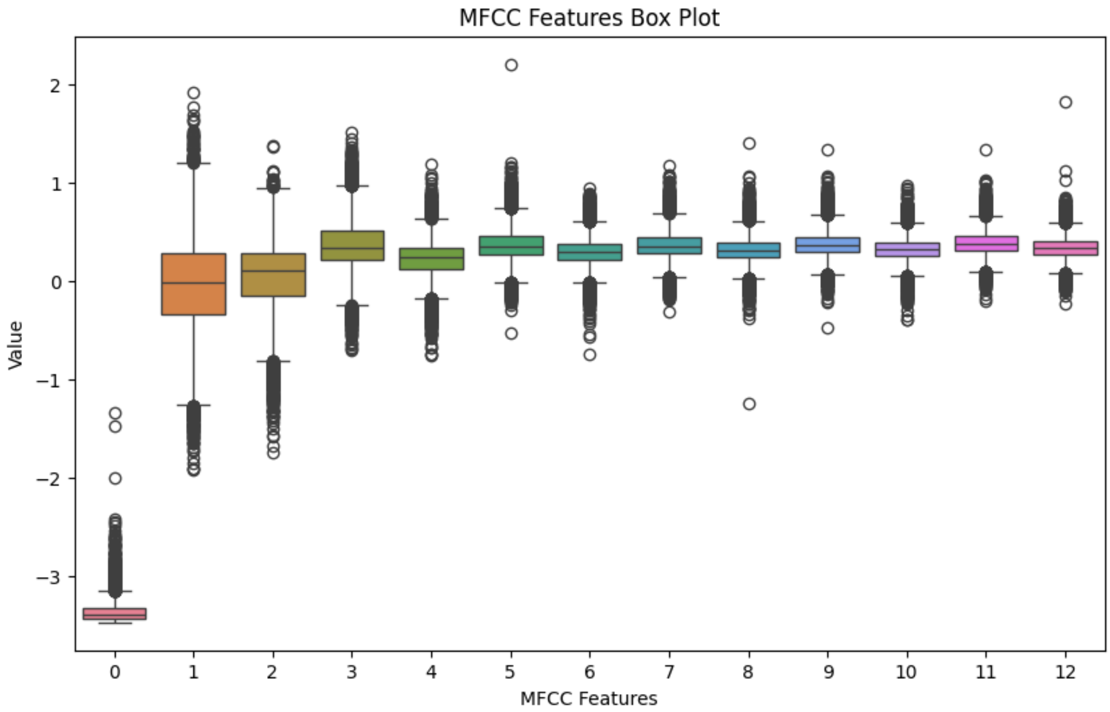

Figure 6.2: MFCC Features Box Plot

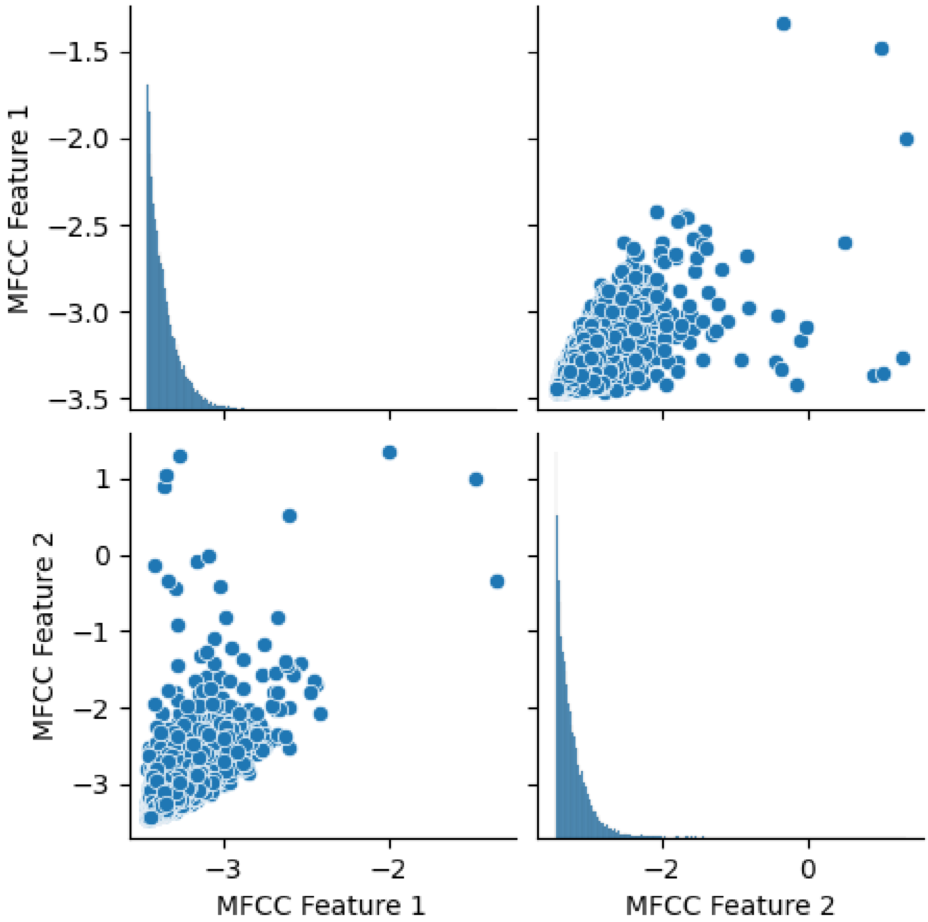

Figure 6.3: MFCC Features

APPENDIX B: Accuracy Curve

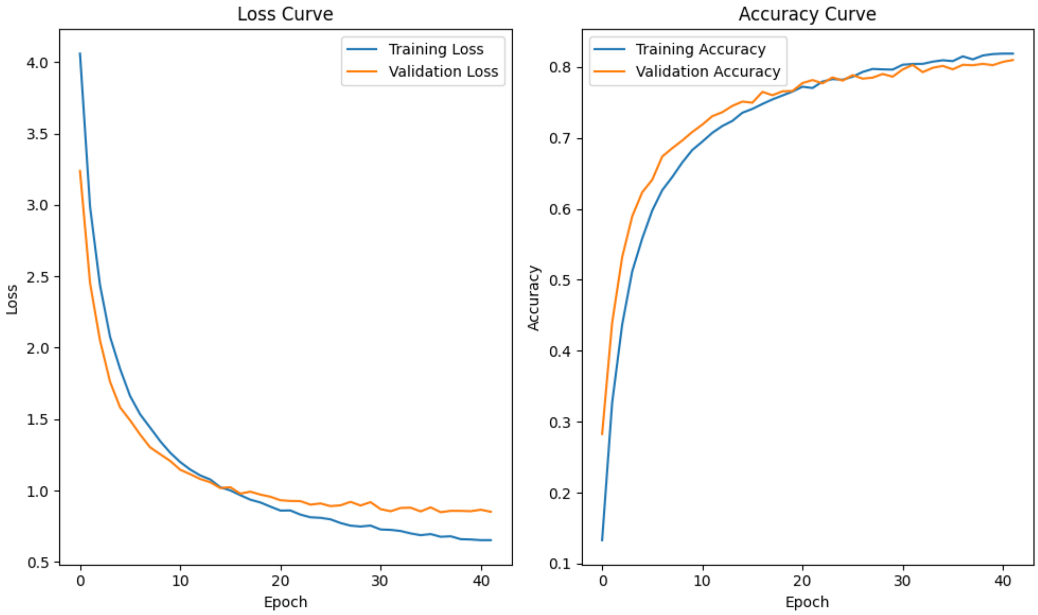

Figure 6.4: Accuracy Curve

.