SoundScape

Bird Sound Recording & Identification

v2.1.0

An intelligent cross-platform mobile application combining audio recording, machine learning, and geolocation for automated bird species identification and biodiversity monitoring.

Flutter 3.9 | TensorFlow Lite | GetX | Appwrite

Comprehensive Technical Documentation

github.com/muhammedshabeerop/SoundScape

Table of Contents

1 Introduction

1.1 Overview

SoundScape is an innovative mobile application designed for ecological bioacoustics research and bird watching enthusiasts. The application leverages modern mobile computing capabilities, machine learning, and cloud infrastructure to enable real-time bird sound recording, analysis, and species identification. Built on the Flutter framework, SoundScape provides a cross-platform solution for iOS, Android, Linux, macOS, Windows, and Web platforms.

1.2 Purpose

- Enable citizen scientists to contribute to biodiversity monitoring
- Provide accurate bird species identification using on-device machine learning
- Create a geospatial database of bird vocalizations
- Support ecological research through crowdsourced acoustic data
- Provide real-time analysis and community-driven data collection

1.3 Scope

SoundScape encompasses audio recording and processing, real-time noise level monitoring, on-device ML inference, cloud synchronization, geographic mapping, and community-driven data collection.

1.4 Problem Statement

Traditional bird watching and biodiversity monitoring face challenges including manual identification requiring expert knowledge, limited spatial coverage, difficulty tracking migratory patterns, lack of standardized data collection methods, and no centralized platform for citizen science contributions.

Key Innovation

SoundScape addresses these challenges by providing an accessible, automated, and scientifically rigorous tool for acoustic biodiversity monitoring with on-device ML processing.

2 Project Details

2.1 Project Information

Property	Value
Project Name	SoundScape
Version	2.1.0+1
Platform	Cross-platform (iOS, Android,
Framework	Flutter 3.9.0
Backend	Appwrite (BaaS)
ML API	FastAPI v5.0.0 with BirdNET v2
Language	Dart
Architecture	GetX (MVC with reactive progra

2.2 Key Features

1. **Audio Recording**
   High-quality capture with real-time waveform visualization
2. **Multi-Species ID**
   Simultaneous detection of up to 5 species using BirdNET v2.4
3. **Visual Rankings**
   Medal-based ranking system with confidence indicators
4. **Noise Monitoring**
   Real-time decibel level measurement and tracking
5. **Geolocation**
   GPS-tagged recordings with interactive map visualization
6. **Cloud Sync**
   Automatic backup and sync across devices via Appwrite
7. **Offline Mode**
   Full functionality without internet connection
8. **Web Interface**
   Interactive map with all recordings on web browsers

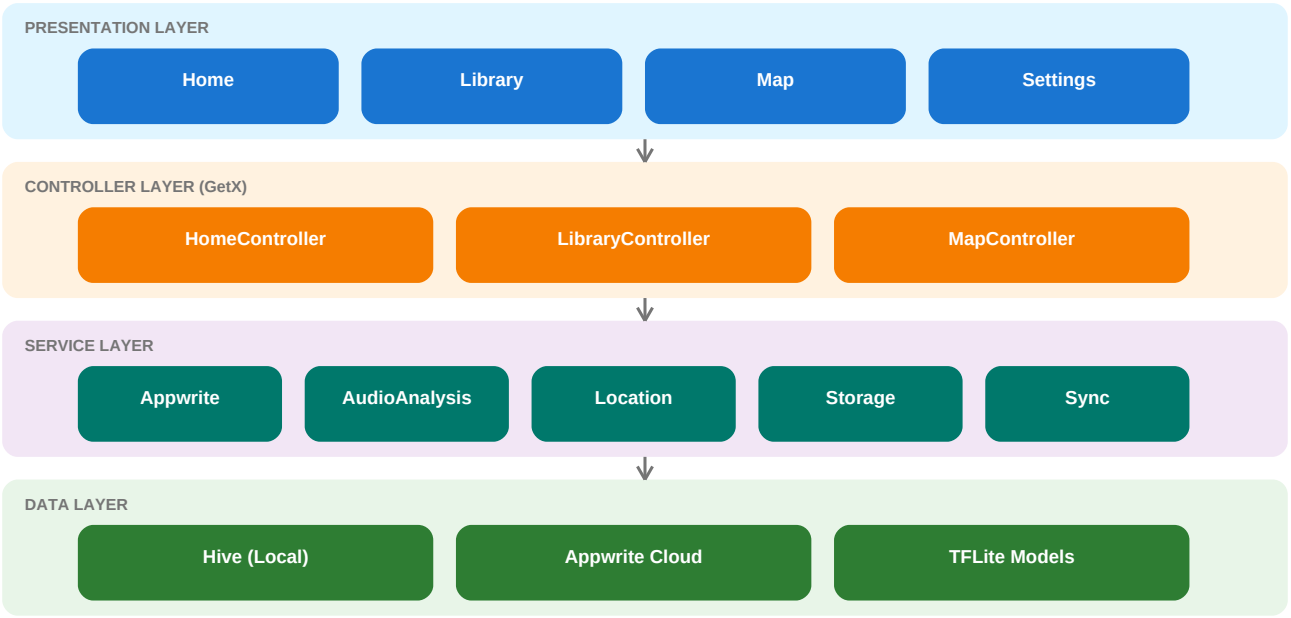
2.3 Target Users

- Bird watching enthusiasts
- Ecological researchers and scientists
- Environmental conservation organizations
- Educational institutions
- Citizen scientists
- Wildlife sanctuary managers

3 System Architecture

3.1 High-Level Architecture

SoundScape follows a layered architecture pattern with clear separation of concerns, using GetX for state management, dependency injection, and routing.



3.2 Component Architecture

Frontend (Flutter): Framework 3.9.0 with Dart SDK, GetX state management, Material Design widgets, GetX navigation and dependency injection.

Backend (Appwrite): User authentication with JWT, NoSQL document store, file storage for audio, serverless functions, and real-time WebSocket updates.

3.3 Data Flow

1. User initiates recording via Home screen
2. Audio captured via FlutterAudioCapture at 16kHz
3. Real-time waveform rendered and GPS coordinates obtained
4. Audio saved as WAV file to local storage
5. ML inference triggered (YAMNet pre-filter → BirdNET classification)
6. Results stored in local Hive database
7. Background sync uploads to Appwrite cloud

4 Tech Stack & Frameworks

4.1 Core Technologies

Component	Technology	Version	Purpose
Framework	Flutter	3.9.0	Cross-platform UI
Language	Dart	3.9.0	Application logic
State Mgmt	GetX	4.7.2	Reactive programming
Backend	Appwrite	20.3.3	BaaS
Local DB	Hive	2.2.3	NoSQL storage
ML Runtime	TFLite Flutter	0.12.1	On-device inference
Maps	Flutter Map	8.2.2	Geographic visualization
Audio	Flutter Sound	9.28.0	Recording/playback

4.2 Audio Libraries

```yaml
flutter_sound: ^9.28.0          # Audio recording/playback
flutter_audio_capture: ^1.1.2   # Low-level audio capture
audio_waveforms: ^2.0.2         # Waveform visualization
```

4.3 Location Services

```yaml
geolocator: ^14.0.2             # GPS positioning
latlong2: ^0.9.1                # Coordinate calculations
flutter_map: ^8.2.2             # Interactive maps
```

4.4 Backend Integration

```yaml
appwrite: ^20.3.3               # Appwrite SDK
dio: ^5.4.0                     # HTTP client
connectivity_plus: ^6.1.1       # Network status
```

4.5 Sensors & Utilities

```yaml
sensors_plus: ^7.0.0            # Accelerometer, gyroscope
flutter_compass: ^0.8.1         # Magnetic compass
permission_handler: ^12.0.1     # Runtime permissions
uuid: ^4.5.2                    # UUID generation
intl: ^0.20.2                   # Internationalization
```

5 Hardware & Software Specifications

5.1 Mobile Device Requirements

Platform	Minimum	Recommended
Android OS	6.0 (API 23)	10.0+
iOS	12.0	15.0+
RAM	2 GB	4 GB
Storage	500 MB	2 GB
Processor	ARM64/x86_64	A9 chip or later

5.2 Sensor Specifications

Sensor	Specification	Purpose
Microphone	20Hz-20kHz, 44.1kHz, 16-bit	Audio capture
GPS	±5-10m accuracy, 1Hz	Location tagging
Accelerometer	±2g to ±16g, 50Hz	Device orientation
Compass	0.1° resolution, ±2°	Recording direction

5.3 Network Requirements

- Minimum: 2G/EDGE for basic data sync
- Recommended: 4G/LTE or WiFi for faster uploads
- Audio upload: ~1 MB/minute (WAV), ~200 KB/minute (compressed)
- Latency: <500ms for real-time features
- Full offline support with background sync when online

5.4 Desktop Requirements

Platform	OS Version	RAM	Storage
Linux	Ubuntu 20.04+	4 GB	1 GB
macOS	10.14 (Mojave)+	4 GB	1 GB
Windows	10 (1809)+	4 GB	1 GB

5.5 Audio Format Specifications

```text
Format: WAV (PCM)
Sample Rate: 44100 Hz (recording), 16kHz/48kHz (ML inference)
Channels: Mono
Bit Depth: 16-bit
Encoding: Linear PCM
```

6 Module Description

6.1 Authentication Module

Purpose: Handle user registration, login, and session management.

- Email/password authentication
- Social login (Google, Apple) support
- Password reset functionality
- Session persistence with automatic token refresh

6.2 Home (Recording) Module

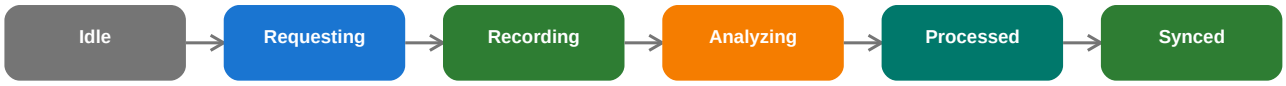
Core audio recording and analysis functionality with real-time processing.

- Real-time waveform visualization and audio level meter
- Recording timer with minimum 15-second enforcement
- Live ML predictions during recording
- Automatic GPS tagging and speech detection

6.3 Recording Pipeline

1. User taps record button → Request microphone permission
2. Initialize audio capture (44.1kHz, mono)
3. Start GPS tracking for location tagging
4. Real-time waveform rendering and noise monitoring
5. Buffer audio for YAMNet speech detection (>70% discards)
6. User stops recording (minimum 15 seconds enforced)
7. Save audio as WAV, trigger BirdNET analysis
8. Store results locally, queue for cloud sync

6.4 Recording State Machine



6.5 Library Module

Browse, search, and manage saved recordings with multi-species indicators.

- List all recordings with primary species and "+X more" indicator
- Search by species name, filter by date/confidence/location
- Swipe-to-delete and bulk selection
- Export recordings and sync status indicator

6.6 Details Module

Display comprehensive information about specific recordings with multi-species support.

- Audio playback with waveform visualization
- Multi-species display with visual rankings (Ø<ßÆØ>ÝHØ>ÝI)
- Detection statistics bar (species count, best match %, confidence)
- Color-coded confidence levels (Green!' Blue!' Orange!' Grey)
- Species information from Wikipedia for all detected species
- Location map and share functionality

6.7 Map Module

Visualize recordings on interactive geographic maps.

- Interactive OpenStreetMap with zoom/pan
- Clustered markers for dense areas
- Filter by species, date, and confidence
- Heatmap visualization for species density

6.8 Noise Monitor Module

Real-time ambient noise level measurement and classification.

dB Range	Classification	Example
0-30	Very Quiet	Whisper, rustling leaves
30-60	Quiet	Normal conversation
60-80	Moderate	Busy office, traffic
80-100	Loud	Factory, lawn mower
100+	Very Loud	Hearing damage risk

6.9 Settings Module

User preferences and app configuration including recording quality, ML model selection, theme settings, and data management.

6.10 Sync Module

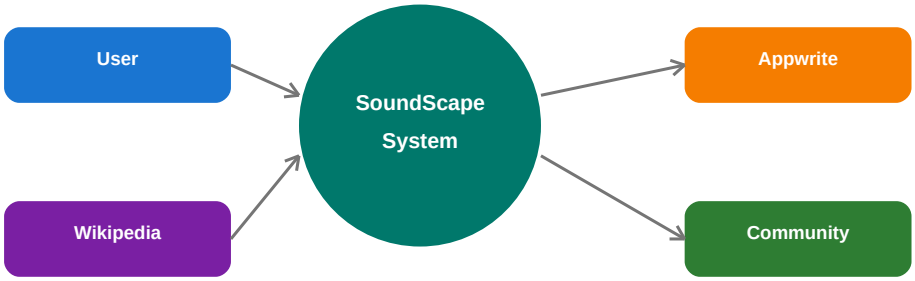
Background synchronization with intelligent network handling.

- WiFi: Auto-upload all pending, download latest models
- Cellular: Upload only if enabled, compress before upload
- Offline: Queue operations, sync when connection restored
- Conflict resolution: Compare timestamps, newer version wins

7 Data Flow Diagrams

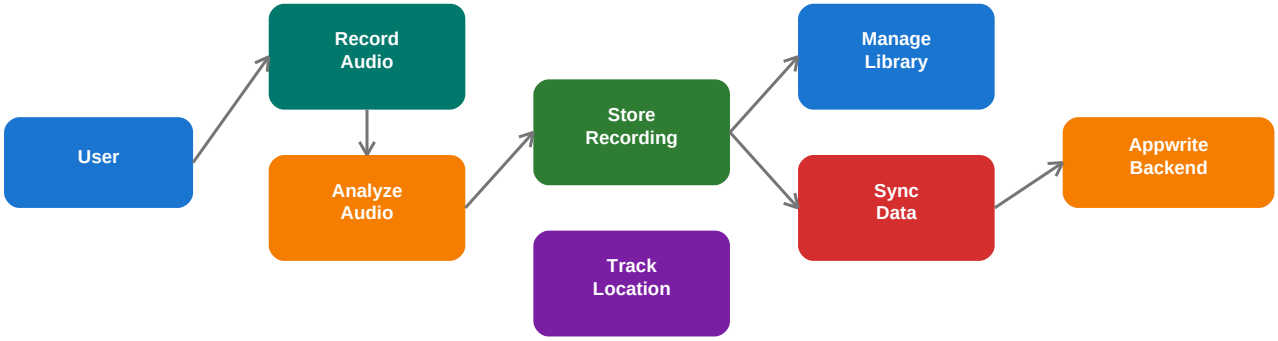
7.1 Context Diagram (Level 0)

The context diagram shows SoundScape system interacting with external entities: User, Appwrite Backend, Wikipedia API, and Community Users.



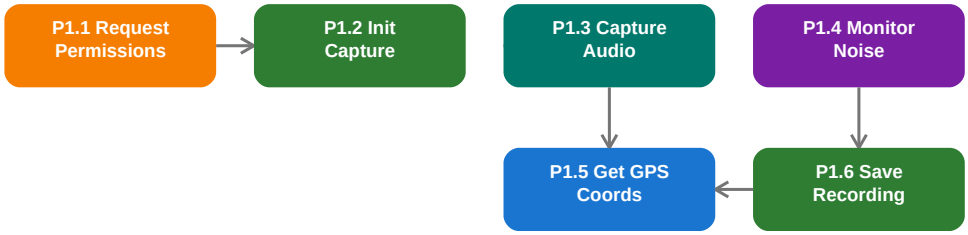
7.2 Main Processes (Level 1)

Level 1 DFD decomposes the system into six main processes that handle recording, analysis, storage, and synchronization.



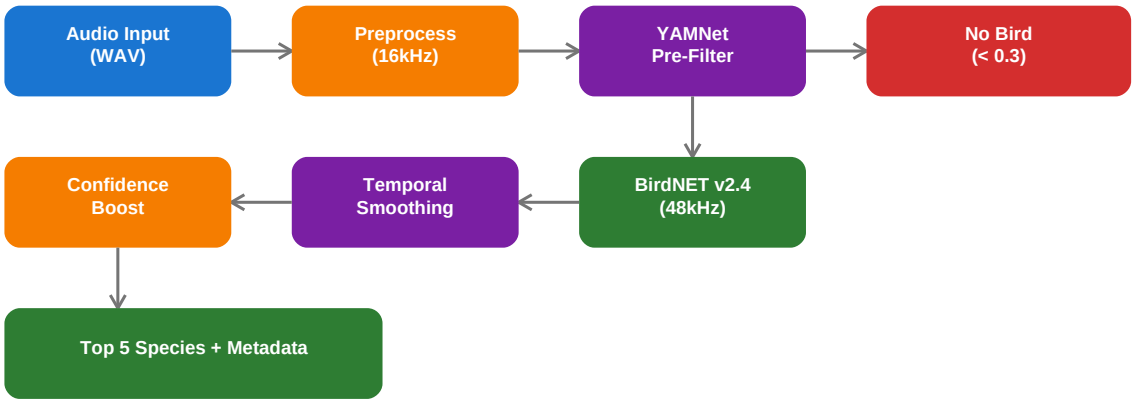
7.3 Recording Process (Level 2)

Detailed breakdown of the recording process from permission request to saving the audio file.



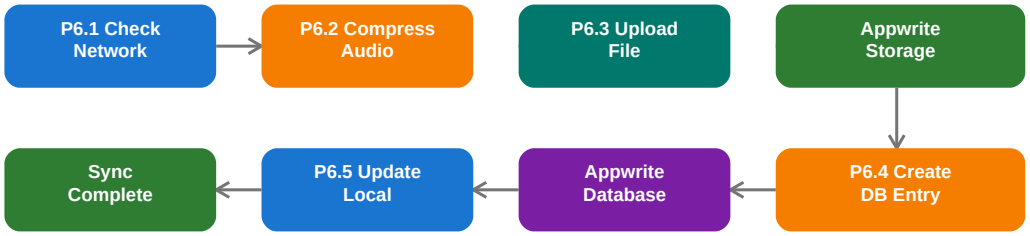
7.4 Analysis Process (Level 2)

The ML pipeline showing audio preprocessing, YAMNet filtering, BirdNET classification, and result generation.



7.5 Sync Process (Level 2)

Data synchronization flow from network check through upload and database entry creation.



7.6 Data Stores

Store	Type	Contents
D1: Local DB	Hive	Recording metadata, preference
D2: Audio Files	File System	WAV recordings, cached images
D3: ML Models	Asset Bundle	YAMNet, BirdNET models
D4: Cloud DB	Appwrite	User accounts, synced recordin

D5: Cloud Storage	Appwrite	Audio files, images

8 Database Design

8.1 Local Database Schema (Hive)

```dart
@HiveType(typeId: 0)
class Recording extends HiveObject {
  @HiveField(0) String id;              // UUID
  @HiveField(1) String filePath;        // Local file path
  @HiveField(2) double latitude;
  @HiveField(3) double longitude;
  @HiveField(4) DateTime timestamp;
  @HiveField(5) int duration;           // milliseconds
  @HiveField(6) String? commonName;     // Primary species
  @HiveField(7) double? confidence;     // 0.0 - 1.0
  @HiveField(8) String status;          // pending/processed/uploaded
  @HiveField(9) String? s3key;          // Cloud storage key
  @HiveField(10) Map<String, double>? predictions; // Top 5
}
```

8.2 Cloud Database Schema (Appwrite)

```json
{
  "$id": "unique()",
  "$collection": "recordings",
  "userId": "string",
  "commonName": "string",
  "scientificName": "string[]",
  "confidence": "float",
  "confidenceLevel": "integer[]",
  "latitude": "float",
  "longitude": "float",
  "timestamp": "datetime",
  "duration": "integer",
  "status": "enum(pending,processed,verified)",
  "s3key": "string",
  "predictions": "json"
}
```

8.3 Database Indexes

- userId_timestamp (userId ASC, timestamp DESC)
- commonName_confidence (commonName ASC, confidence DESC)
- location (latitude, longitude) - Geospatial index
- status_timestamp (status ASC, timestamp DESC)

9 Machine Learning Models

9.1 Multi-Model Architecture (v5.0.0)

SoundScape utilizes a two-stage ML pipeline combining YAMNet for bird detection/filtering and BirdNET v2.4 for species classification.

9.2 YAMNet (Bird Pre-Filter)

Property	Value
Purpose	Fast bird detection to filter
Architecture	MobileNet-based CNN
Source	TensorFlow Hub
Input	Audio waveform @ 16kHz
Output	521 audio event classes
Threshold	0.3 (30% bird confidence)
Processing Time	~0.1-0.2s for 5s audio

9.3 BirdNET v2.4 (Species Classification)

Property	Value
Purpose	High-accuracy bird species ide
Architecture	EfficientNet backbone CNN
Source	Zenodo (Cornell Lab)
Input	3s spectrograms @ 48kHz
Output	6,522 global bird species
Model Size	~40MB (TFLite FP32)
Processing Time	~0.8-1.2s per 3s segment

9.4 Advanced Prediction Algorithms

Overlapping Windows: 50% overlap (2.5s stride) for boundary detection, ensuring birds vocalizing near segment boundaries are detected.

Temporal Smoothing: Dual-method smoothing using moving average convolution and exponential weighted average to reduce prediction jitter.

Confidence Boosting: Up to 10% boost when both YAMNet and BirdNET agree on bird presence, increasing user trust in results.

9.5 Performance Metrics

Metric	v1.0 (BirdNET only)	v2.0 (Combined)	Improvement
Bird Detection Precision	85%	92%	+7%
Species Accuracy (Top-1)	78%	89%	+11%
Species Accuracy (Top-5)	91%	97%	+6%
Non-bird Rejection	80%	96%	+16%
Processing (bird audio)	2.5s	2.8s	-12%
Processing (non-bird)	2.5s	0.3s	+88%

9.6 Latency Analysis (15s audio)

```text
Audio download:        0.5s
Preprocessing (16kHz): 0.2s
YAMNet pre-filter:     0.2s
Preprocessing (48kHz): 0.3s
BirdNET inference:     4.5s (6 chunks)
Temporal smoothing:    0.1s
Post-processing:       0.2s
% % % % % % % % % % % % % % % % % % % % % % % % % % % % %
Total:                 ~6.0s

For non-bird audio:    ~0.9s (85% faster!)
```

10 API Documentation

10.1 Appwrite REST API

Base URL: https://fra.cloud.appwrite.io/v1

```http
Authentication:
POST /account/sessions/email     # Login
GET  /account/sessions/current   # Current session
DELETE /account/sessions/{id}    # Logout

Database:
POST   /databases/{db}/collections/{col}/documents
GET    /databases/{db}/collections/{col}/documents
PATCH  /databases/{db}/collections/{col}/documents/{id}
DELETE /databases/{db}/collections/{col}/documents/{id}

Storage:
POST /storage/buckets/{bucket}/files
GET  /storage/buckets/{bucket}/files/{id}/view
GET  /storage/buckets/{bucket}/files/{id}/download
```

10.2 ML Classification API

```json
POST /classify/combined

Request:
  Content-Type: multipart/form-data
  file: audio file (WAV/M4A)

Response:
{
  "model": "combined",
  "bird_detected": true,
  "predictions": [
    {"class_name": "American Robin", "score": 0.89},
    {"class_name": "House Sparrow", "score": 0.45}
  ],
  "confidence_method": "boosted",
  "processing_time": 5.2,
  "audio_duration": 15.0
}
```

10.3 Third-Party APIs

- Wikipedia API: GET /api/rest_v1/page/summary/{species}
- OpenStreetMap: Tile server for flutter_map

11 Future Enhancements

11.1 Short-term (3-6 months)

- Spectrogram visualization with full frequency spectrum
- Sound event detection for individual calls within recordings
- Noise reduction pre-processing to enhance audio quality
- User profiles with achievements and statistics
- Comments, discussions, and verification system
- Leaderboards for top contributors

11.2 Medium-term (6-12 months)

- Custom model training for personal models
- Federated learning for privacy-preserving training
- Smart watch app for quick recording from wearables
- Web dashboard for browser-based analysis
- eBird integration for observation export
- GBIF export for Global Biodiversity Information Facility

11.3 Long-term (1-2 years)

- Migration tracking to visualize species movement
- Population trends and habitat analysis
- Image recognition combined with audio identification
- Multi-language support (20+ languages)
- Regional models specialized for different continents
- Natural language queries ("Show me robins near water")

11.4 Technical Improvements

- Background processing with queued analysis
- Incremental sync for changed data only
- End-to-end encryption for sensitive recordings
- Two-factor authentication
- CDN integration for faster global downloads
- GraphQL API for efficient data fetching

12 Version Changelog

12.1 Version 2.1.0 "Web Explorer"

Release Date: February 3, 2026

- NEW: Interactive web map interface with full-screen OpenStreetMap
- NEW: Details panel with animated slide-in sidebar
- NEW: Platform-aware routing (kIsWeb detection)
- NEW: Marker system with user location and recordings
- NEW: Playback controls in web details panel
- IMPROVED: All recordings displayed on map with click interaction

12.2 Version 2.0.0 "Multi-Species"

Release Date: February 3, 2026

- MAJOR: Multi-species detection (up to 5 species per recording)
- MAJOR: Two-stage ML pipeline (YAMNet + BirdNET v2.4)
- NEW: Visual rankings with medal system (Ø<ßÆØ>ÝHØ>ÝI)
- NEW: Color-coded confidence levels
- NEW: Detection statistics bar
- NEW: Temporal smoothing and confidence boosting
- IMPROVED: 88% faster processing for non-bird audio
- IMPROVED: +11% species accuracy (Top-1)

12.3 API Version 5.0.0

- NEW: POST /classify/combined endpoint
- NEW: Bird pre-filter with 0.3 threshold
- NEW: Overlapping windows (50% stride)
- NEW: Temporal smoothing algorithms
- NEW: Confidence boosting (up to 10%)
- OPTIMIZED: kaiser_fast resampling (30% speedup)

13 Bibliography

13.1 Academic References

1. Kahl, S., et al. (2021). "BirdNET: A deep learning solution for avian diversity monitoring." Ecological Informatics, 61, 101236.
2. Gemmeke, J.F., et al. (2017). "Audio Set: An ontology and human-labeled dataset for audio events." IEEE ICASSP, 776-780.
3. Stowell, D., et al. (2019). "Automatic acoustic detection of birds through deep learning." Methods in Ecology and Evolution.
4. Vellinga, W.P. & Planqué, R. (2015). "The Xeno-canto Collection and its Relation to Sound Recognition." CLEF Working Notes.

13.2 Technical Documentation

1. Flutter Documentation (2024). Flutter Framework. https://flutter.dev/docs
2. Dart Programming Language (2024). Dart Language Specification. https://dart.dev/guides
3. Appwrite Documentation (2024). Appwrite BaaS. https://appwrite.io/docs
4. TensorFlow Lite (2024). TFLite Guide. https://tensorflow.org/lite/guide
5. GetX Documentation (2024). GetX State Management. https://pub.dev/packages/get

13.3 Online Resources

- Cornell Lab of Ornithology - Macaulay Library: https://macaulaylibrary.org
- Xeno-canto Foundation - Bird Sound Database: https://xeno-canto.org
- TensorFlow Hub - YAMNet: https://tfhub.dev/google/yamnet/1
- Zenodo - BirdNET v2.4: https://zenodo.org/records/15050749

A Appendix A: Installation Guide

A.1 Prerequisites

- Flutter SDK: 3.9.0 or higher
- Dart SDK: 3.9.0 or higher
- Android Studio / Xcode (for mobile development)
- Git for version control

A.2 Clone and Install

bash

```
# Clone repository
git clone https://github.com/muhammedshabeerop/SoundScape.git
cd SoundScape/frontend

# Install dependencies
flutter pub get

# Configure Appwrite (edit appwrite.config.json)
{
  "projectId": "your-project-id",
  "endpoint": "https://fra.cloud.appwrite.io/v1",
  "databaseId": "your-database-id",
  "bucketId": "your-bucket-id"
}
```

A.3 Run Application

bash

```
# Android
flutter run -d android

# iOS
flutter run -d ios

# Web
flutter run -d chrome

# Desktop
flutter run -d linux    # or macos, windows
```

A.4 Build for Production

bash

```
# Android APK
flutter build apk --release

# Android App Bundle
flutter build appbundle --release

# iOS
flutter build ipa --release

# Web
flutter build web --release
```

B Appendix B: Troubleshooting Guide

B.1 Common Issues

Issue	Solution
App crashes on recording	Ensure microphone permissions
Species not identified	Record for at least 15 seconds
GPS showing 0,0	Enable location services and w
Sync failing	Check internet connection and
Low confidence scores	Move closer to bird, reduce ba
Model not loading	Reinstall app to restore bundl

B.2 Performance Optimization

- Close other apps to free RAM for ML inference
- Use WiFi for faster audio uploads
- Enable auto-upload only on WiFi to save mobile data
- Clear app cache periodically for storage management

B.3 Contact & Support

- GitHub Issues: https://github.com/muhammedshabeerop/SoundScape/issues
- Email: support@soundscape.app
- Documentation: COMPREHENSIVE_DOCUMENTATION.md

Generated from COMPREHENSIVE_DOCUMENTATION.md