Senior Design

Bat Lab
Software Requirements Specification

Version 1.1

Revision History

Date	Version	Description	Author
05/Dec/2025	1.0	Group Final Draft	Team
02/Apr/2026	1.1	Revised Draft	Riley Phan

Table of Contents

Software Requirements Specification

1. Introduction

1.1 The Purpose of the *BatLab Acoustic Classification System*

The BatLab Acoustic Classification System (BACS) is a web-based tool that helps bat researchers and wildlife biologists automatically classify bat species from their echolocation recordings. Researchers currently spend significant time manually inspecting spectrograms and listening to calls to identify species and filter out noise. BACS reduces this workload by generating spectrograms from uploaded `.wav` *files, running a trained machine learning model (CNN + tabular features) to predict species, and associating each prediction with recording metadata such as location, date/time, and device. The system also supports adding new labeled recordings so the model can be fine-tuned to specific locations to improve accuracy and adapt to regional variation in bat calls. The primary users are bat researchers, field technicians, and students working with acoustic datasets.*

1.2 The Purpose of this Document

The purpose of this Software Requirements Specification (SRS) is to describe the functional and nonfunctional requirements for Release 1.0 of the BatLab Acoustic Classification System (BACS). This document provides a complete description of the external behavior of the system, the data it manages, and the quality attributes it must satisfy. It is intended to align project stakeholders, including the sponsor, senior design team, and course staff, by clearly defining the scope, functionality, constraints, and interfaces of the system. The SRS will guide implementation, testing, and future maintenance.

1.3 Document Conventions

Requirements ID format: *REQ-X.Y (e.g., REQ-5.1)*

Quality attributes: *Prefix with category (e.g., USE-1 for usability, PER-1 for performance).*

System name: *"BACS" refers to the BatLab Acoustic Classification System.*

Section 7 Requirement Types:

Section 7 (Data Requirements) includes specific sub-categories with consistent prefixes: ***DA-#*** *— Data Acquisition requirements,* ***DI-#*** *— Data Integrity requirements,* ***DR-#*** *— Data Retention requirements,* ***DD-#*** *— Data Disposal requirements)*

Bold text indicates defined terms that also appear in the project glossary.

1.4 References

- Project Glossary: URL
- Vision and Scope: URL
- Use Cases: URL
- Business Rules: URL
- Prototype: URL

2. Project Glossary

The project glossary contains all of the associated definitions and terminology employed throughout the various documents referenced in the Bat Lab project. The link to the glossary is available here: URL

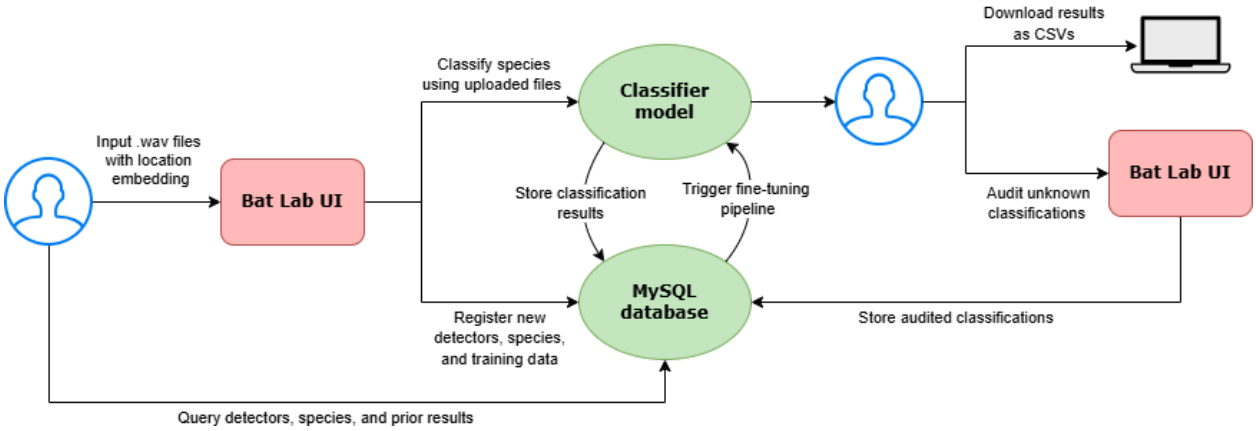
3. Vision and Scope

The vision and scope document outlines the limits of the Bat Lab project, including the business requirements, desired outcomes, and the features and capabilities of the system. Furthermore, it defines what the Bat Lab project serves and why its implementation exists now. The document is accessible here: URL

The problem of	*manually classifying and identifying a significant number of bat calls*
affects	*researchers with large amounts of unclassified bat call data*
the impact of which is	*a high cost in time required for identification*
a successful solution would be	*a Machine Learning Model that would dramatically improve classifying and accurately identifying local bat species*

4. Software Architecture

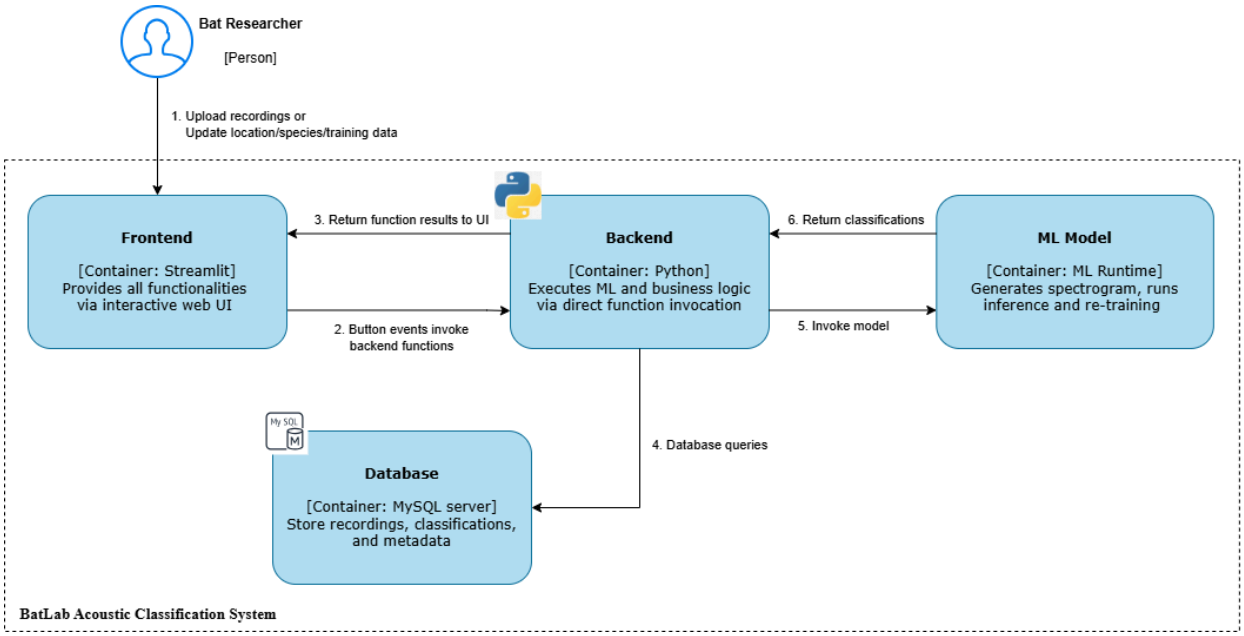
4.1 System Context Diagram



The Level 1: The system context diagram shows BACS as a central web application used by **Bat Researcher** and **Field Technician** actors. Users upload audio recordings from local storage or an external shared drive, view spectrograms and model predictions, and download results. BACS stores recordings, predictions, and inference metadata in a database, which are accessible to users. Optionally, authorized admins can trigger a model retraining process using the accumulated labeled data. External systems include:

- **External Storage / Research Drive** – source of `.wav` files for upload.

4.2 Container Diagram



The container diagram for BACS includes the following containers:

- **Web Client (Web UI)**
  - Runs in the user's browser.

    - Provides pages to: run inference, inspect species classifications and download results through sound recordings, add and view detectors, add and view bat species, manage training data, and run fine-tuning on selected subsets of detectors and species.
- **Backend Application**
    - Handling UI events and coordinating workflow.
    - Calling the ML Model component for inference.
    - Persisting and retrieving data from the database and file storage.
    - Returning results back to the UI for display.
- **ML Model Service**
    - Python-based service that generates spectrograms from .wav files and runs the trained CNN + tabular model to produce species predictions and confidence scores.
    - Fine-tunes the based model on species subset to provide more accurate prediction for localized data
    - Can be invoked synchronously (single file) or asynchronously for batches.
- **Database**
    - Stores recording metadata, file paths, detectors and species labels, predictions, confidence scores, and training data for fine-tuning purposes.
- **File Storage**
    - Stores uploaded `.wav` files and generated spectrogram images (e.g., in a file system folder or cloud storage bucket).

Here are the detailed steps:

1. Upload .wav files or new data via the BACS website: The Bat Researcher begins by accessing the system through their browser on localhost (via a preset batch script) and indicates their intended usage.
2. Invoke backend functions by registering button events: The Backend (built using Python) listens to button events from the Frontend (served on Streamlit) that users interact with to invoke the correct functions for desired functionalities.
3. Backend return function results to users: Invoked backend functions return results to users by displaying them on the frontend.
4. Backend makes queries to MySQL database: The Backend communicates with the Database by calling functions that make underlying database queries .
5. Backend invokes ML Model: The Backend interacts with the ML Model by calling functions to make predictions or finetune by providing the corresponding parameters within the call.
6. ML Model returns classification results: The ML Model generates spectrograms provided by the Backend to run inference or finetuning. In case of an inference request, classification results are returned to Backend and then Frontend as described in step 3. In case of a fine-tuning request, ML Model returns classification results: The ML Model generates spectrograms provided by the Backend to run inference or finetuning. In case of an inference request, classification results are returned to Backend and then Frontend as described in step 3. In case of a fine-tuning request, the updated model weights and parameters are persisted, and a new model version is logged for user's usage when classification.

This sequence of actions outlines how the BACS system components collaborate to facilitate seamless functionality for the bat researcher while ensuring efficient data management and communication.

4.3 Operating Environment

OE-1: BACS shall be deployed on a Python-based server environment, with the backend application running locally on each start up. All environment setup and application requirements are automatically installed on initial setup.
OE-2: End users shall access BACS via localhost on a modern web browser (Chrome, Firefox, Edge, Safari) on desktop or laptop computers.
OE-3: The database shall run on a relational database management system (e.g., PostgreSQL or MySQL) hosted on the same server within the same secure network.

OE-4: The system is intended primarily for use by researchers at Texas Christian University, with potential future extension to additional institutions.

4.4 Design and Implementation Constraints

CO-1: The system shall be implemented using technologies approved by the course (e.g., Python for the ML pipeline and backend, and a web framework such as Flask/Django).
CO-2: The CNN model shall be implemented using a standard deep learning library (e.g., TensorFlow or PyTorch).
CO-3: The system shall integrate with the existing GitHub repository and follow the team's branching and code review policies.
CO-4: The database design shall follow the team's existing ER diagram and must store enough metadata to support future retraining.
CO-5: The system must run within the resource limits of the assigned university infrastructure (e.g., no dedicated GPU in production).

4.5 Assumptions and Dependencies

AS-1: Users have already collected `.wav` *recordings that contain bat calls with sufficient signal quality.*
AS-2: At least a subset of uploaded recordings will be labeled with the correct species and location to serve as training data.
AS-3: Users have sufficient storage space for model and database storage.

DE-1: The system depends on external Python libraries (e.g., NumPy, SciPy, librosa, matplotlib, TensorFlow/PyTorch) being available in the runtime environment.
DE-2: The system depends on the university-provided server or cloud platform remaining available and properly maintained.

5. Functional Requirements

5.1 Use Cases

The use cases are available here: URL.

UC-5.1-1 – ***Submit Collected Sound Files***
Researchers upload acoustic recordings associated with known locations or species so that the system can later classify them.

UC-5.1-2 – ***Classify Species from Uploaded Sound Files***
Researchers request automatic species identification using the trained classification model.

UC-5.1-3 – ***Audit Unknown Classification Results***
Researchers manually review and label recordings the model could not classify confidently.

UC-5.1-4 – ***Generate a Report of Classification Results***
Researchers export a CSV file containing species predictions, behavior classifications, and extracted call parameters.

UC-5.1-5 – ***Generate a Report of the Calls Database***
Researchers (with Admin permissions) export a complete dataset of all stored recordings and metadata for broader analysis.

UC-5.1-6 – ***Update Database of Existing Bat Calls***
Administrators add additional validated calls to existing species to improve prediction ability for localized species.

UC-5.1-7 – ***Add New Bat Species to an Existing Location***
Administrators register new species from detectors in specific regions so that classification models reflect correct ecological distributions.

UC-5.1-8 – ***Add New Sound Detector Location to Database***
Administrators register new detector sites so recordings can be associated with valid locations for training and reporting.

UC-5.1-9 – ***Audio File Output***
The system produces spectrograms, processed audio features, and derived parameters for each recording.

UC-5.1-10 – ***Graphical Visualization (GUI)***
The system displays sonograms and detailed pulse-level information through an interactive GUI.

5.2 Non-Use Case Functional Requirements

REQ-5.2-1 – ***Single File Prediction***
When a researcher uploads a single `.wav` *file, the system shall generate a spectrogram and return a predicted species label and confidence score.*

REQ-5.2-2 – ***Batch Processing***
When a researcher uploads a batch of `.wav` *files (e.g., a zip folder), the system shall enqueue them for processing, generate spectrograms, and produce predictions for each file.*

REQ-5.2-3 – ***Metadata Storage***
The system shall store, for each recording, at minimum: file identifier, upload user, upload timestamp, species label (predicted and/or confirmed), location metadata (e.g., site name, region), and model version used.

REQ-5.2-4 – ***Label Editing***
When a user with "Researcher" permissions updates a species label for a recording, the system shall record the new label, the user who changed it, and the timestamp of the change.

REQ-5.2-5 – ***Confidence Threshold Flagging***
When the model's prediction confidence is below a configurable threshold (e.g., 0.8), the system shall flag the recording as "Unknown" in the UI to signify manual review.

REQ-5.2-6 – ***Search and Filter***
The system shall allow users to filter recordings by species, location, date range, prediction confidence, and review status.

REQ-5.2-7 – ***Export***
The system shall allow users to export a CSV file containing selected recordings with their metadata, predictions, and labels.

REQ-5.2-8 – ***Preprocessing Enforcement***
When a .wav file is uploaded, the system shall automatically run the file through preprocessing before classification.

REQ-5.2-9 – ***Invalid File Handling***
When the system detects an unsupported, corrupted, or empty audio file, it shall reject the file and notify the user.

REQ-5.2-10 – ***Unknown Output Rule***
When the model cannot confidently classify a species or behavior, the system shall return "Unknown" instead of generating a forced prediction.

REQ-5.2-11 – ***Model Version Tracking***
The system shall store the version of the model used for each prediction to ensure scientific reproducibility.

REQ-5.2-12 – ***Role-Based Access Control***
The system shall restrict editing of species labels, thresholds, and training rules to authorized users only.

6. Business Rules

The business rules are available here: URL.

BR-1: The system shall grant access at two permission levels:

1) Users
    a) Users may upload recordings, view predictions, audit unknown classifications, and export reports
2) Admin
    a) Admins may additionally manage species lists, detector locations, confidence thresholds, and training data, and may trigger model fine-tuning

BR-2: The model may only classify audio clips that meet required recording standards, including acceptable signal-to-noise ratio, file integrity, and presence of detectable call pulses. Files that do not meet these standards must be flagged as unusable and must not proceed to classification.

BR-3: If the model cannot confidently assign a species or behavior due to noise, overlap, or insufficient acoustic detail, the output must be returned as “Unknown.” No forced classification is permitted.

BR-4: The system may reject any file whose metadata or format is not compatible with the preprocessing pipeline. Only .WAV audio files will be accepted for processing.

BR-5: All automated classifications must be based solely on the parameters learned from the curated and verified call library provided by expert researchers.

BR-6: The model must not generate classifications for species or behaviors that do not exist in the trained regional dataset. Predictions must be constrained to the species and behaviors defined by the client.

BR-7: Visualization of spectrograms, call parameters, and prediction confidence must be available for every processed file, ensuring transparency of all model decisions.

BR-8: Model fine-tuning is only permitted when new, expert-validated call data is provided. No external or unverified datasets may be added to the training library.

BR-9: If a prediction is validated as incorrect by an expert reviewer, it must be immediately marked for correction and included in the training dataset for the next fine-tuning cycle.

BR-10: In cases where multiple bats call simultaneously within the same clip, the model may return only one dominant species. If no dominant call can be reliably isolated, the result must be returned as “Unknown.”

BR-11: The tool cannot override expert determinations. When expert review contradicts a model prediction, the expert decision is final and must be recorded as the confirmed label.

BR-12: Users may modify species-specific classification thresholds only through approved Admin configuration GUI. Direct modification of internal model files or parameters is prohibited. Threshold values must remain within system-defined valid ranges.

BR-13: The system shall export results, including call parameters and prediction outputs, only in approved formats (.CSV). No other export formats are permitted for Release 1.0.

BR-14: If the preprocessing software (Sonobat) fails or does not produce a valid output for a file, the system must not proceed to classification for that file. Preprocessing must successfully complete before the classification stage begins.

BR-15: Any updates to classification rules, species lists, behavior definitions, or confidence thresholds must be logged with timestamps and the identity of the user who made the change to maintain full version control.

BR-16: A prediction label may only be manually edited by a user with Researcher or Admin permissions. All label changes must be recorded with the new label, the user who changed it, and the timestamp of the change.

BR-17: Model fine-tuning may only be initiated by an Admin. Fine-tuning requires a minimum quantity of validates, label training examples as defined by the system configuration before a fine-tuning run may begin.

BR-18: Uploaded audio files and their associated classification results must be retained for a minimum of 5 years. Expert-validated training datasets must be retained indefinitely to support scientific reproducibility.

BR-19: Batch processing jobs must complete in their entirety before results are made available for export. Partial or incomplete batch results must not be exported.

7. Data Requirements

7.1 Business Domain Model

The Bat Lab domain focuses on the identification and classification of bat echolocation calls. The primary entities and relationships represent how audio files, extracted call parameters, species profiles, and classification results interact within the system.

Key Domain Entities

- **AudioFile**
  - Represents an uploaded .WAV file containing bat calls.
  - Contains metadata such as recording location, acoustic features, and device ID.
- **CallSegment**
  - Individual pulse or call extracted from an AudioFile.
  - Includes duration, slope, shape, minimum/maximum/characteristic frequencies, and waveform attributes.
- **Species**
  - Represents a bat species known to inhabit the Eastern Cape region.
  - Includes region-specific call parameter ranges and known behavioral patterns.
- **Detector**
  - Represents a recording device used for capturing echolocation in a specific area.
  - Contains information about its location and the recordings it has captured.
- **ClassificationResult**
  - The model's output for a CallSegment or AudioFile.
  - Contains predicted species, predicted behavior, confidence score, and optionally "Unknown."
- **TrainingExample**
  - An instance of **AudioFile** used for supervised fine-tuning, labeled with a specific **Species** and **Detector**.
  - Contains an AudioFile, CallSegments, expected species, and behavior labels.
- **SubsetModel**
  - An improved version of the base classification model that was fine-tuned on a chosen subset of TrainingExample
- **User**
  - Researcher or analyst interacting with the system.
  - May create or update classification rules and initiate fine-tuning.

Domain Relationships (Described)

- An **AudioFile** contains one or multiple **CallSegments**.
- Each **CallSegment** may be classified into a **Species**.
- **Users** can request to use a **SubsetModel** for species classification.
- Each classification produces a **ClassificationResult**.
- **TrainingExamples** contain **AudioFiles** and **CallSegments** labeled with **Species**, and are specific to one or multiple **Detectors**.
- A **Species** may map to multiple **Detectors**.
- **Users** may trigger exports, configure model thresholds, and update rules

7.2 Data Acquisition, Integrity, Retention, and Disposal

DA-1*: The system shall allow users to upload audio recordings in .WAV format produced by bat detectors.*

DA-2: *The system shall automatically extract call segments and compute call parameters, including minimum frequency, maximum frequency, characteristic frequency, duration, slope, and call shape.*

DA-3*: The system shall accept pre-cleaned audio files generated from external tools (e.g., SonoBat) and ingest their associated metadata.*

DA-4*: The system shall support importing expert-labeled training data containing known species and behavioral classifications.*

DA-5*: The system shall allow new labeled examples to be added incrementally for future fine-tuning or retraining.*

DI-1*: The system shall store uploaded recordings and their associated metadata unless manually deleted by a Database Admin.*

DI-2*: The system shall retain prediction results and label history for each recording for the lifetime of the project.*

DI-3*: When a recording is deleted by an Admin, the system shall permanently remove all associated data.*

DI-4*: The system shall validate that uploaded files are in an allowed audio format (e.g.,* `.wav`*) before processing.*

DR-1: *All uploaded audio files, extracted call parameters, and classification outputs shall be retained for* ***a minimum of 5 years****.*

DR-2: *Expert-validated training datasets used for model development and retraining shall be retained indefinitely to support reproducibility.*

DR-3: *System logs, including rule updates, retraining events, and user actions, shall be retained for at least* ***2 years****.*

DR-4: *Exported analytical products (e.g., .xlsx parameter files) shall be stored or versioned only when explicitly saved by the user.*

DR-5: *Intermediate temporary files generated during segmentation or visualization may be retained only for the duration of the active session.*

DD-1: *Temporary processing artifacts (e.g., cached spectrograms, intermediate segmentation output) shall be automatically deleted after workflow completion.*

DD-2: *When a user deletes an uploaded audio file, associated call segments, parameters, and cached visualizations shall also be removed from active storage.*

DD-3: *Deleted data shall not remain accessible through the GUI or any user interface component.*

8. External Interface Requirements

8.1 User Interfaces

UI-1: The system shall provide a ***Dashboard*** *section summarizing recent uploads, recent predictions, and key statistics (e.g., number of recordings by species and number of "Unknown" classifications).*

UI-2: The system shall provide an ***Upload*** *page where users can upload single or multiple .wav files, view upload progress, and see validation or error messages for unsupported or corrupted files.*

UI-3: The system shall provide a ***Recordings Table*** *page with sortable and filterable columns, including at minimum: file name, species, prediction confidence, location, date, and review status.*

UI-4: The system shall provide a ***Classification Detail*** *view for each recording that displays the spectrogram image, predicted species and/or behavior, confidence score, and full recording metadata, and allows authorized users to edit the ground-truth label.*

UI-5: The system shall provide a ***Reports / Export*** *page where users can configure filters (e.g., by species, date range, location, confidence threshold) and export a CSV file of classification results, including parameters and prediction outputs.*

UI-6: The system shall provide an ***Admin Configuration*** *page available only to Admin users to manage species lists, detector locations, and configuration parameters such as confidence thresholds and classification rules.*

UI-7: The UI shall be usable on a standard laptop display (e.g., 1366×768 or larger) without requiring horizontal scrolling for core workflows (upload, classification, review, export).

UI-8: All pages shall include consistent navigation, a clear indication of the current page or workflow step, and clearly worded error/validation messages (e.g., "File must be a .wav audio file") to guide users in correcting problems.

UI-9: The UI shall follow the team's approved wireframes and layouts defined in the Prototype deliverable for this project.

8.2 Software Interfaces

SI-1: Relational Database Interface

- ***SI-1.1:*** *The backend shall interact with a relational database management system (e.g., MySQL or PostgreSQL) to store and retrieve recordings, spectrogram references, metadata, predictions, user accounts, and configuration settings using SQL queries.*
- ***SI-1.2:*** *The database connection shall use authenticated credentials configured in the server environment; no direct database access shall be available from the client browser.*
- ***SI-1.3:*** *The database schema shall support tracking the model version used for each prediction so that results are reproducible in future analyses.*

SI-2: ML Model Service Interface

- ***SI-2.1:*** *The backend shall invoke the ML Model component through an internal Python API (function calls or module interface), passing either the file path to the stored* `.wav` *file or pre-extracted audio features and receiving back predicted species/behavior labels and confidence scores.*
- ***SI-2.2:*** *For batch classification, the backend shall support queued or iterative calls to the ML Model service and aggregate the results for each file before returning them to the UI.*
- ***SI-2.3:*** *The backend shall log the model version identifier returned by the ML Model service or configured in the environment alongside each prediction.*

SI-3: File Storage Interface

- ***SI-3.1:*** *The backend shall save uploaded audio files and generated spectrogram images to a designated file storage location (e.g., local file system directory or cloud storage bucket) and store corresponding paths/URLs in the database.*
- ***SI-3.2:*** *The backend shall read files from storage when needed for preprocessing, classification, visualization, or export workflows.*
- ***SI-3.3:*** *File storage operations (create, read, delete) shall be encapsulated behind a storage access module so that the underlying storage technology may be changed (e.g., from local disk to cloud) without affecting other components.*

SI-4: External Preprocessing and Call-Parameter Tools (e.g., SonoBat)

- ***SI-4.1:*** *The system shall be able to ingest pre-cleaned audio files and associated metadata produced by external tools (e.g., SonoBat), mapping their exported fields (such as frequency ranges and call parameters) into the BACS data model.*
- ***SI-4.2:*** *When integrated directly, the backend shall invoke approved preprocessing tools via command-line or library APIs, passing file paths to input recordings and reading resulting cleaned files or parameter files from a configured output directory.*
- ***SI-4.3:*** *If preprocessing (e.g., SonoBat) fails or does not produce valid output, the system shall not proceed to classification for the affected file and shall log the failure according to the business rule that preprocessing must complete before classification.*

SI-5: Web Framework / Frontend Interface

- ***SI-5.1:*** *The Web Client (Streamlit-based UI) shall communicate with the Backend Application via HTTP requests and responses within the same deployment, using framework-provided mechanisms for invoking backend functions and returning results for display.*

- ***SI-5.2:*** *The interface between the Web Client and Backend shall support passing file upload payloads, filter parameters, and user actions (e.g., "classify batch", "export CSV") and returning status messages, prediction results, and generated links to downloadable files.*

8.3 API Document

No API system was used for this project.

8.4 Hardware Interfaces

No dedicated hardware interfaces (e.g., sensors, custom devices) have been identified for Release 1.0. The system assumes that recordings have already been captured and stored as files by external recording equipment.

8.5 Communications Interfaces

CI-1: Client–Server Web Communication

- ***CI-1.1:*** *The system shall communicate with clients (user web browsers) using HTTP/HTTPS over standard ports (80/443). For production and staging deployments, all traffic shall be served over HTTPS to protect credentials and research data in transit.*
- ***CI-1.2:*** *The web interface shall be accessible via a stable URL (e.g., a Streamlit Cloud URL or university-hosted URL), and any HTTP requests to the site shall be redirected to HTTPS where supported.*

CI-2: Backend–Database Communication

- ***CI-2.1:*** *The Backend Application shall communicate with the relational database over TCP/IP using the database vendor's standard protocol, restricted to a private network or localhost interface.*
- ***CI-2.2:*** *Database connections shall use authenticated credentials and, where supported, encrypted connections (e.g., TLS) configured by the deployment environment.*

CI-3: Internal Service Communication

- ***CI-3.1:*** *Communication between the Backend Application and the ML Model Service shall occur within the same runtime environment through in-process API calls or, if deployed separately, over an internal network interface that is not exposed to the public Internet.*
- ***CI-3.2:*** *Any intermediate results (e.g., spectrogram images, temporary feature files) exchanged between components shall be referenced via file paths or identifiers stored in the database rather than transmitted directly over the network.*

CI-4: User Notifications

- ***CI-4.1:*** *For Release 1.0, user notifications (e.g., "classification completed", "export ready", "file rejected") shall be delivered via the web interface (inline messages, status banners, or dialogs). No outbound email or SMS communication is required.*
- ***CI-4.2:*** *If future releases add email notifications, they shall use an authenticated SMTP or mail service over TLS and restrict attachm*ents to small text-based outputs such as CSV reports, in accordance with university policies.

9. Quality Attributes

9.1 Usability

USE-1: A new researcher with basic computer skills shall be able to upload a recording and view a prediction within 10 minutes of first using the system, without formal training.

USE-2: All primary features (upload, view predictions, download report) shall be easily accessible.

USE-3: Error messages shall clearly describe what went wrong and what the user can do to fix it (e.g., "File must be a .wav audio file").

USE-4: The system shall display spectrograms alongside predictions to allow researchers to visually verify results.

9.2 Performance

PER-1: For single .WAV recordings under 5 MB (typical 3-4 second clips), 90% of prediction requests shall complete and display results within 10 seconds.

PER-2: The system shall support batch processing of at least 100 recordings without crashing.

PER-3: The system shall support at least 2 concurrent users performing uploads and queries without performance degradation.

9.3 Security

SEC-1: The system shall validate all uploaded files to accept only .wav format and reject invalid file types with clear error messages.

SEC-2: Admin routes (update database, add species, add location) shall include input validation.

SEC-3: The system shall log all database modifications (species additions, location updates, classification updates) with timestamps for audit purposes.

9.4 Safety

SAF-1: The system shall prevent accidental data loss by requiring explicit confirmation before deleting any recording or dataset.

SAF-2: When predictions have low confidence scores, the system shall clearly flag these for manual expert review.

9.5 Availability

AVL-1: The system shall target 95% availability during typical research use hours (8:00 AM – 6:00 PM local time), excluding scheduled maintenance.

9.6 Robustness

ROB-1: If a recording fails to process (e.g., corrupted file, incompatible format), the system shall mark it as "Processing Failed," log the error details, and allow the user to re-upload or delete the file.

ROB-2: If the database connection fails during upload or classification, the system shall display an error message and allow the user to retry.

ROB-3: The system shall handle invalid admin inputs by displaying validation errors rather than crashing or allowing corrupted data into the database.

9.7 Modifiability

MOD-1: The codebase shall be organized into separate modules (UI, ML pipeline, database access) to support future student teams extending the system.

10. Deployment

For this project, BACS will be deployed in a **single-server configuration** suitable for a prototype research tool:

- **Development environment:**
  - Backend and frontend is deployed locally on localhost
  - Components include:
    - Backend Python services
    - Streamlit frontend
    - PyTorch model file
    - Local MySQL or SQLite instance
  - Used for iterative testing and debugging.
- **Staging / Demo environment:**
  - Backend code is maintained on GitHub.
  - Frontend is deployed on Streamlit Cloud.
  - Mirrors production configuration on a smaller scale.
  - Used for demos, reviews, and user acceptance testing.

Deployment Procedure:

- Download the latest code from the GitHub repository to the client's device.
- Windows batch file installs dependencies using the appropriate package manager and launch application as a UI in the client's browser.

Update and Maintenance Process:

- Merge updates into the main branch.
- Pull updates from the main branch onto the client's device.
- Run migration scripts if needed (for database server and/or ML services).

Future Plans:

- Host the PyTorch model and database in a cloud environment to improve scalability and accessibility.
- Explore cloud-managed ML services and database hosting (e.g., AWS SageMaker, RDS) for production deployment.

11. Internationalization and Localization Requirements

For Release 1.0, BACS will support only English language labels and messages. There are no explicit internationalization or localization requirements. Future versions may consider support for additional languages if the user base expands.

12. Other Requirements

OR-1: The system shall include basic logging of key events (logins, uploads, prediction requests, errors) to support debugging and auditability.
OR-2: The team shall provide minimal user documentation (e.g., a "Getting Started" page) describing how to upload recordings, interpret predictions, and export data.
OR-3: The system shall comply with any applicable university policies for research data storage and access.

13. Appendix A