	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Capstone	**Aim: Implementation - Continuous progress review**	
Submission - 4	**Date:**	**Enrollment No: 92200133044**

Implementation – Continuous Progress Review

1. Introduction

The implementation phase of the capstone project focuses on developing a fully functional prototype of the **Unified Voice Application**. This system integrates **voice data management, search, matching, and direct comparison functionalities** to address challenges in voice-based authentication and organization.

During this phase, emphasis was placed on **high-quality coding standards, modular architecture, robust functionality, and seamless integration** of all components. The system was developed using **Python** for backend processing, **Streamlit** for the user interface, and **SQLite** for database management. Audio processing leverages **Librosa**, **NumPy**, and **SciPy** to ensure accurate voice comparisons and reliable similarity measurements.

2. Problem Statement

Existing solutions for voice management and verification face the following issues:

- Scattered and unorganized storage of audio files, making retrieval difficult.
- Inaccurate or unreliable voice comparison tools.
- Limited support for multiple audio formats and batch processing.
- Complexity in usage for non-technical users.

The implementation addresses these problems by providing a **user-friendly, efficient, and accurate system** capable of managing, searching, and comparing voice data in multiple formats while ensuring data integrity.

3. Objectives

The main objectives achieved during implementation are:

1. **Voice Data Management:** Upload, organize, and store audio files efficiently.
2. **Voice Search and Matching:** Search by name or compare audio files using MFCC, DTW, and cosine similarity algorithms.
3. **Direct Voice Comparison:** Accurately determine if two audio files belong to the same speaker.
4. **System Integration:** Ensure smooth interaction between front-end, backend, and database components.
5. **Code Quality and Documentation:** Maintain clean, modular, and well-commented code for maintainability.

4. Relevance to ICT Domain

This project demonstrates key ICT concepts, including:

- **Multimedia Processing:** MFCC extraction and dynamic time warping for voice analysis.
- **Database Management:** Efficient storage and retrieval of audio data using SQLite.
- **Web-based Application Development:** Streamlit provides a responsive, interactive UI.
- **Software Engineering Principles:** Modular design, error handling, input validation, and code documentation.

5. Feasibility Analysis

Technical Feasibility

- Python, Streamlit, and SQLite were used due to their simplicity and efficiency.
- Librosa library ensures accurate audio processing and feature extraction.
- The modular design allows independent testing of components, reducing errors.

Economic Feasibility

- Open-source tools and libraries reduce project costs.
- Minimal hardware requirements make the system cost-effective.

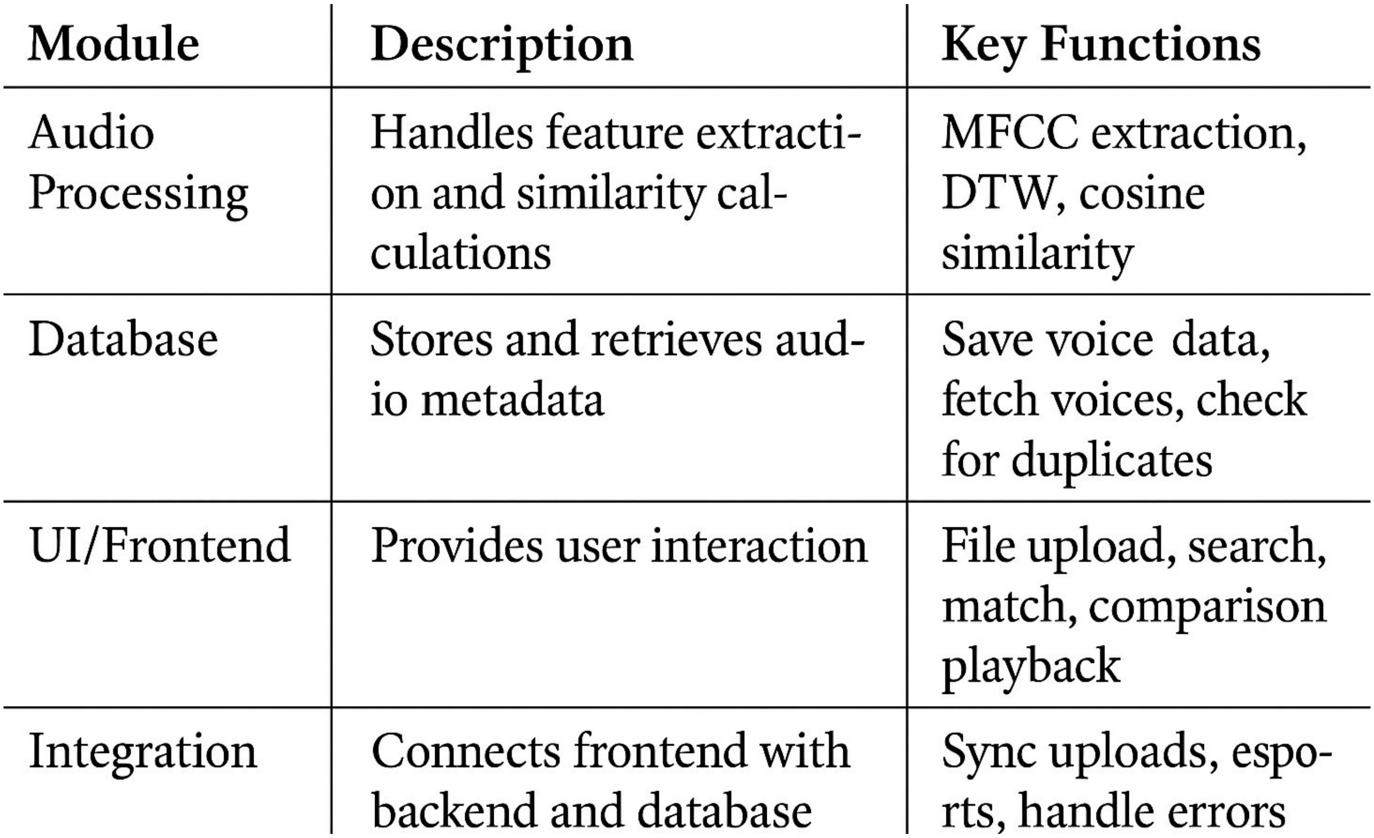
Operational Feasibility

- Simple and intuitive user interface ensures easy adoption.
- Batch upload, file syncing, and automated database management enhance operational efficiency.

6. Market/User Needs Analysis

Voice-based systems are widely used in:

- **Banking:** Voice authentication for secure transactions.
- **Healthcare:** Managing patient audio records for identification and monitoring.
- **Law Enforcement:** Detecting and verifying suspects or witnesses.
- **Education:** Storing lecture recordings for easy retrieval.



Module	Description	Key Functions
Audio Processing	Handles feature extraction and similarity calculations	MFCC extraction, DTW, cosine similarity
Database	Stores and retrieves audio metadata	Save voice data, fetch voices, check for duplicates
UI/Frontend	Provides user interaction	File upload, search, match, comparison playback
Integration	Connects frontend with backend and database	Sync uploads, esports, handle errors

User requirements addressed by the implementation:

- Fast and accurate voice matching.
- Multi-format audio support and batch upload.
- Clear and actionable comparison results.
- Simple interface for non-technical users.

7. Literature Review

Existing research highlights:

- **MFCC Feature Extraction:** A standard for capturing voice characteristics.
- **Dynamic Time Warping (DTW):** Aligns two time-series signals for comparison.
- **Cosine Similarity:** Useful for comparing voice embeddings and verifying speaker identity.
- **Database Solutions:** SQLite is sufficient for small to medium-scale voice management systems.

8. Implementation Details

❖ Code Structure and Organization

- **Backend:** Python modules for audio processing, database handling, and voice comparison.
- **Frontend:** Streamlit interface for uploading, managing, and comparing voices.
- **Database:** SQLite for storing metadata and file paths.

❖ Functional Implementation

- **Voice Upload:** Supports individual audio files and ZIP archives.
- **Search & Match:** Users can search by name or upload a sample to find matches.
- **Direct Comparison:** Compares two audio files and returns verdict with similarity scores.
- **Error Handling:** Ensures robustness through input validation and exception management.

❖ Integration Across Components

- Frontend communicates with Python backend modules via Streamlit functions.
- Database interactions are encapsulated in separate modules for clarity.
- Voice comparison and search modules fetch data from the database or upload directory seamlessly.

Testing Procedures

- **Unit Testing:** Each module was tested individually for functionality.
- **Integration Testing:** End-to-end testing ensured all components worked together.
- **Results:** Screenshots and logs confirm successful uploads, searches, and accurate comparison results.

9. Conclusion

The implementation phase successfully delivered a fully functional, integrated voice management system. It meets all objectives, including voice upload, search, matching, and comparison.

Key outcomes:

- Clean, modular, and documented code for maintainability.
- Accurate and reliable voice comparison using MFCC, DTW, and cosine similarity.
- Seamless integration between frontend, backend, and database.
- User-friendly interface supporting multiple audio formats and batch processing.