Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

First Prototype Testing Plan

LungDetect:
Acoustic Imaging Device for Diagnosing Pneumonia

by

Team 21

Team Members:

Thinh Nguyen tnguy19@bu.edu
Matthew Pipko mattpi23@bu.edu
Hilario Gonzalez hilario@bu.edu
Astrid Mihalopoulos astridm@bu.edu
Shadin Almaiman shadin@bu.edu

Required Materials

Hardware:

1. Foam Enclosure
2. Two piezo contact microphones
3. Computer with USB-C slot
4. 3.5mm TRRS headphone male-to-male cable
5. Two female 3.5mm jack to male usb-c dongles
6. USB-C splitter hub
7. Voicemeeter for mixing both microphones into single input
8. Audacity for recording audio files using the microphones

Software:

1. Front-end:
    - React app:
        - Automatically starts up app running at network address localhost:3000 on the device
        - Receives .wav data file input through upload (or from detect)
        - User Interface built and customized with javaScript, HTML, and CSS
        - Backend link using Axios
2. Back-end:
    - Express server:
        - Listens on port 5000 with ‘app.post('/compute' ...’
        - Uses Multer to receive a .wav input from user upload (or in-app recording) and stores it in the application directory at ‘./public/data/uploads/’
        - Spawns a process running the data analysis script and passes the file name
        - Sends formatted output from data analysis script back to front end for visualization
    - Python data analysis script:
        - Interprets .wav file into a matrix. Rows = number of channels, Cols = number of samples
        - Stores all instances of audio spikes
        - Identifies spike families based on temporal proximity across channels
        - Filters out families that don’t have the appropriate number of channels represented
        - For every remaining family it selects only the channel instance with the highest amplitude
        - Now every crackle family is defined by the single channel instance with the highest amplitude within a window of time.
        - The channel with the earliest peak is selected as the mother, and every other channel is a daughter. Every channel is cross correlated with the mother in order to find how much the sound lagged behind on the daughter channel, and to calculate a transmission coefficient.
        - All of these results are returned in an array

Set Up and Pre-Testing Setup Procedure

Server/Software Side:

1. Ensure all dependencies for the React front-end web app and Express backend are installed and correctly configured.
2. Start server hosting React front-end app on local machine with 'npm start' command on the command line, by default hosting at address: localhost:3000.
3. Start server hosting Express backend-end on local machine with 'node server' command on the command line, by default hosting at address: localhost:5000. Ensure the command is run at the appropriate folder/directory on the project's folder on the local machine.
4. Ensure the front-end app correctly boots up as an automatic web page that appears if started up correctly. Backend app should return the output 'Server listening on port 5000' on the terminal of the IDE or command line if it boots up correctly.
5. Ensure there is enough space for uploaded files to be stored on 'backend/public/data/upload/' as this will be where uploaded sound files will be stored.

Hardware Side:

1. Make sure of proper connections with all the cables and dongles.
2. Open VoiceMeeter and mix both USB Audio microphone inputs into one stereo virtual output for Audacity recording (B1 output).
3. Open Audacity, set the recording device as the VoiceMeeter B1 virtual output, and open a stereo track for recording.

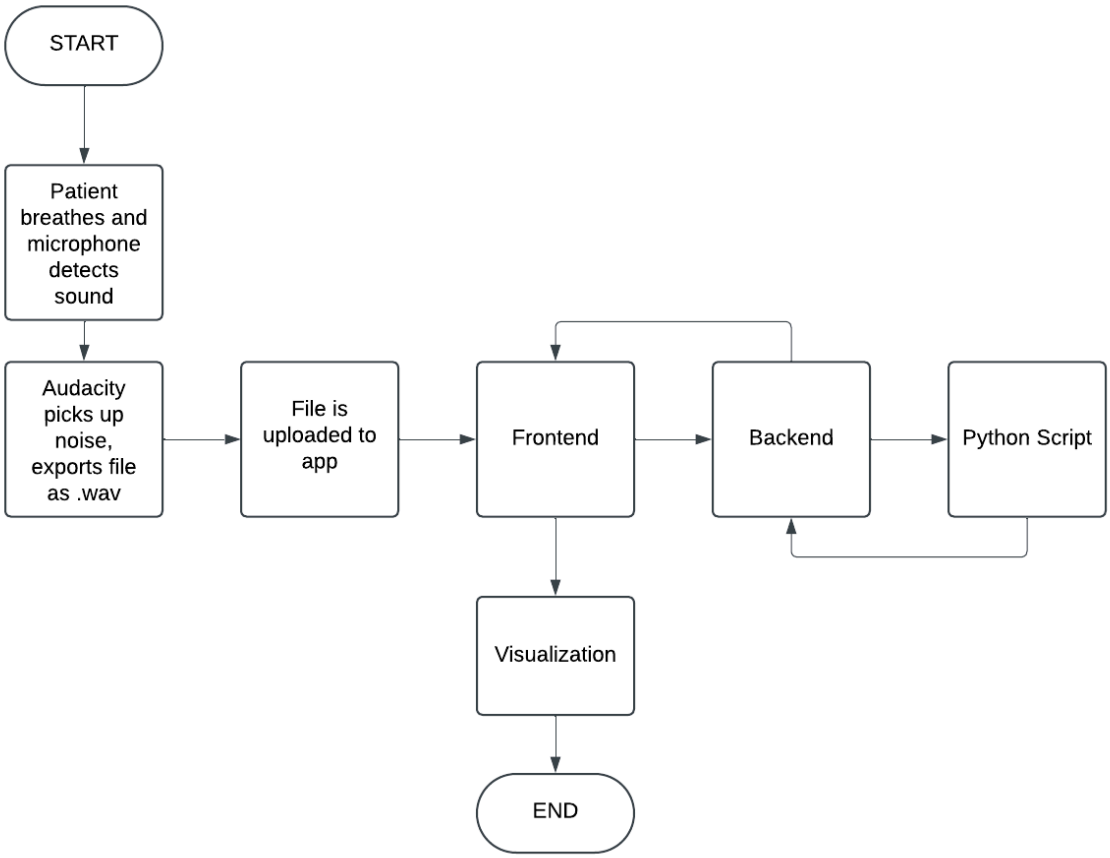

Figure 1: *Illustration of Flow*

Testing Procedure:

Hardware Side:

1. Place the device behind a testing subject's back, ensuring all microphones are in contact with the subject's back.
2. Briefly test the device's microphones by having the patient casually breathe in and out while recording.
3. Prepare the subject with a countdown before recording the data, ensuring they are ready.
4. Start recording on Audacity for a couple of minutes.
5. Stop the recording and export the recorded audio as a .wav file.

Software Side:

1. Allow users to upload an audio file and save it to the app directory.
2. Invoke the data analysis script with the uploaded file name.
3. Identify crackle instances and calculate the delay on every channel.
4. Server sends results back to the front end.
5. Front end produces visualization.

Measurable Criteria

Hardware Side:

1. The device must be able to record 2-channel sound.
2. Each microphone should record on a separate channel from the other in a stereo audio track.
3. Microphones should be able to detect sound while being embedded within the foam enclosure.

Software Side:

1. Software should be able to start both frontend and backend server running simultaneously
2. React App capable of allowing users to choose and upload a .wav file, then commence computation when the 'compute' button is pressed (no computation before or after this action should happen).
3. Software capable of receiving and storing .wav files in the project's directory on a local machine in the path 'backend/public/data/uploads/'.
4. The uploaded files should be able to be recognized and located by the backend Express app which will call the python script to run with the file. All of this will happen very quickly (almost in real-time), errors if any will be logged on the console terminal, successful runs will print the data received on the terminal.
5. Computation and result visualization should happen near instantaneously, more specifically no more than 30 seconds maximum. Also the process should happen automatically with no further manual input from the user after .wav file have been submitted
6. Correctly identify which side of the lungs that the sound detected is likely located in, labeled as either 'Sound' or 'No Sound'

Score Sheet

Microphone Contact (Left/Right)	**Visualized Location (Left/Right)**	**Correct? (Y/N)**