**VoiceScan: Parkinson's Disease Voice Screening App**

**VoiceScan** is a web-based proof-of-concept application designed for the non-invasive, preliminary analysis of vocal features associated with Parkinson's Disease (PD). It utilizes a specialized Machine Learning model, trained on acoustic characteristics, to classify a voice sample as either **'Healthy'** or indicating **'Parkinson's Likelihood.'**

**Critical Medical and Ethical Disclaimer**

**This is a research and educational project only and is NOT a medical diagnostic tool.**

* The results provided by this application are computational classifications and **DO NOT constitute a medical diagnosis.**
* The output must **NEVER** be used for self-diagnosis or to replace consultation with a qualified healthcare professional (e.g., a neurologist or speech-language pathologist).

**The Problem and Solution**

**Problem Statement**

Early diagnosis of Parkinson's Disease (PD) is challenging but critical for effective treatment. Since over 90% of PD patients experience voice disorders (hypokinetic dysarthria), vocal instability (e.g., increased jitter and shimmer) often serves as a key early biomarker. The challenge is creating an accessible, objective screening method to quantify these subtle vocal changes.

**Solution**

VoiceScan addresses this by:

1. **Standardizing Input:** Requiring users to read a specific, standard passage for a consistent speech sample.
2. **Objective Analysis:** Extracting acoustic features (MFCC/MFE) from the recorded audio on the backend.
3. **Machine Learning:** Running the features through a pre-trained classification model (deployed via Edge Impulse WebAssembly) to provide a preliminary likelihood assessment.

**User Workflow & Protocol**

To ensure consistency in the voice samples, the application adheres to a strict 5-step user protocol.

1. **Display Prompt:** The frontend displays the standard reading passage.
2. **Start Recording:** The user clicks "Start Recording" and reads the passage aloud.
3. **Capture Sample:** The system captures **10–15 seconds** of the reading.
4. **Backend Processing:** The recorded audio is sent to the backend where acoustic features (MFCC/MFE) are extracted and fed into the ML model.
5. **Display Result:** The frontend receives and displays the result (e.g., "Result: Healthy" or "Result: Parkinson's Likelihood").

**Standard Reading Passage**

The user must read the following passage to provide the required speech sample:

**“The North Wind and the Sun were disputing which was the stronger, when a traveler came along wrapped in a warm cloak. They agreed that the one who first succeeded in making the traveler take his cloak off should be considered stronger than the other. Then the North Wind blew as hard as he could, but the more he blew the more closely did the traveler fold his cloak around him; and at last the North Wind gave up the attempt. Then the Sun shone out warmly, and immediately the traveler took off his cloak. And so the North Wind was obliged to confess that the Sun was the stronger of the two.”**

**Technical Stack**

**Frontend**

* **Technologies:** HTML, CSS, JavaScript (using Web Audio API for recording).
* **Role:** Handles user interface, audio recording, and sending the audio blob to the backend API.

**Backend & Inference Engine**

* **Framework:** (e.g., Node.js with Express/Flask/FastAPI - *specify your framework here*)
* **Machine Learning Model:** Deployed via **Edge Impulse WebAssembly (WASM)**.
* **Feature Extraction:** Utilizes an audio processing library (e.g., librosa if using Python, or a Node.js equivalent) to convert the raw audio into the feature vector (MFCC/MFE array) required by the ML model.

**ML Model Deployment**

The core classification logic is housed in the node/ directory, which contains the Edge Impulse deployment package:

| File | Description |
| --- | --- |
| node/edge-impulse-standalone.wasm | The compiled WebAssembly file containing the neural network weights and signal processing logic. |
| node/edge-impulse-standalone.js | The JavaScript wrapper for loading and interacting with the WASM module. |
| node/run-impulse.js | An example script demonstrating how to run inference on a feature vector. |

Export to Sheets

**Local Setup Guide**

**1. Prerequisites**

You will need to have **Node.js** (and optionally **Python/Flask/Express** if using a traditional backend server) installed.

**2. Frontend & Backend Integration**

*(Instructions here should detail how to set up the web server and link the front-end files.)*

**3. ML Inference Engine Test**

To confirm the Edge Impulse model is operational in the Node.js environment:

1. Navigate to the node/ directory.
2. Run the included test script with a placeholder feature vector (the server logic will replace this with real features):

Bash

# This example runs the model using a placeholder feature set.

# Replace the list of numbers with a comma-separated feature vector (MFCC/MFE)

$ node run-impulse.js "0.1, 0.5, 0.9, -0.3, 1.2, -0.8, ..."

1. The script will output the prediction (e.g., {"prediction": "Healthy", "confidence": 0.95}).

**4. Running the Full Application**

*(Add the final command to start your main server here, e.g.:)*

Bash

# Example for a Node.js/Express server

$ node server.js

# Example for a Python/Flask server

$ flask run

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| File | Description |  |  |  |  |  |
| node/edge-impulse-standalone.wasm | The compiled WebAssembly file containing the neural network weights and signal processing logic. |  |  |  |  |  |
| node/edge-impulse-standalone.js | The JavaScript wrapper for loading and interacting with the WASM module. |  |  |  |  |  |
| node/run-impulse.js | An example script demonstrating how to run inference on a feature vector. |  |  |  |  |  |

**References :**

<https://it.wikipedia.org/wiki/La_tramontana_e_il_sole>

<https://fr.wikipedia.org/wiki/Alphabet_phon%C3%A9tique_international_cursif>

<https://www.kaggle.com/datasets/nutansingh/mdvr-kcl-dataset>