Voice Authentication API Documentation

Overview

This system provides voice-based user authentication using FastAPI. The API allows:

- **User Registration**: Upload a voice sample and register a user.
- **User Authentication**: Upload a voice sample and verify if it matches a registered user.
- **User History Dashboard**: Retrieve the history of authentication attempts for a specific user.

It includes **audio liveness detection**, **feature extraction**, and **SQLite-based storage**, and is **cloud deployable** and **testable via Postman**.

---

1. API Endpoints

Base URL

cpp
CopyEdit
`http://<your-ec2-public-ip>:8000`

1.1 Register User

Endpoint: `POST /register`

Description: Registers a user with a `.npy` voice file.

Form Data:

- `username`: string
- `file`: `.npy` audio file

Response:

json
CopyEdit

```
{
  "status": "success",
  "message": "Voice registered for user '<username>'."
}
```

Error (e.g., Liveness failure):

json
CopyEdit

```
{
  "detail": "Liveness check failed."
}
```

1.2 Authenticate User

Endpoint: `POST /authenticate`

Description: Authenticates a user by comparing submitted audio with stored embedding.

Form Data:

- `username`: string
- `file`: `.npy` audio file

Response:

json
CopyEdit

```
{
  "username": "john",
  "similarity_score": 0.8123,
  "authenticated": true
}
```

Liveness or Unknown User Error:

json
CopyEdit

```
{
```

```
  "authenticated": false,
  "reason": "Liveness check failed"
}
```

1.3 Get Voice History

Endpoint: `GET /history/{username}`

Description: Returns a user's authentication attempt history.

Response:

json
CopyEdit

```
{
  "user": "john",
  "history": [
    {
      "score": 0.8123,
      "success": true,
      "message": "Authenticated",
      "timestamp": "2025-04-18T12:34:56"
    },
    ...
  ]
}
```

---

2. Cloud Deployment Guide (AWS EC2)

2.1 Setup EC2 Instance

- Launch an Ubuntu EC2 instance
- Open ports: **22 (SSH), 8000 (FastAPI)** in the Security Group
- SSH into the instance

bash
CopyEdit

```
ssh -i your-key.pem ubuntu@<public-ip>
```

2.2 Install Dependencies

bash
CopyEdit

```
sudo apt update
sudo apt install python3-pip ffmpeg -y
pip3 install fastapi uvicorn numpy sounddevice librosa scipy
matplotlib
```

2.3 Transfer Files

Upload project files via `scp`:

bash
CopyEdit

```
scp -i your-key.pem *.py ubuntu@<public-ip>:~/voice-auth/
```

2.4 Run API

bash
CopyEdit

```
cd voice-auth
uvicorn main:app --host 0.0.0.0 --port 8000
```

Your API is now live at:

cpp
CopyEdit

```
http://<your-ec2-public-ip>:8000
```

Test it via **Postman** using the above endpoints.

---

3. Benefits of the System

Liveness Detection

Prevents spoofing using recorded or silent audio.

Voice Embedding

MFCC-based feature extraction for speaker comparison.

Denoising Support

Removes background noise before feature computation.

SQLite Database

Stores voice embeddings and authentication logs.

Historical Logs

All attempts are timestamped and can be retrieved.

Modular Design

Voice utilities are separated for easy maintenance (`voice_utils.py`).

Postman-Testable

Endpoints accept `multipart/form-data` (username + `.npy` file).

---

4. Bonus Features Implemented

Feature	Description
User Registration Endpoint	Register with a `.npy` voice file
User Authentication Endpoint	Match against stored embedding with similarity score
User History Dashboard	Track all attempts with success/failure, timestamps, and reasons
Liveness & Denoising	Ensures only real-time speech and clean audio is processed
Cloud Deployable	Works on AWS EC2 or GCP Compute with minimal setup
Postman Compatibility	Easily test endpoints using Postman with file upload & form data

---

5. Testing via Postman

1. **Set Method**: `POST`
2. **URL**: `http://<public-ip>:8000/register` or `/authenticate`
3. **Body Type**: `form-data`
4. **Keys**:
    - `username`: your desired username (type: text)
    - `file`: upload `.npy` file (type: file)

---

6. Sample `.npy` File Creation (Local)

python
CopyEdit

```
import sounddevice as sd
import numpy as np

duration = 4  # seconds
fs = 16000
audio = sd.rec(int(duration * fs), samplerate=fs, channels=1,
dtype='float32')
sd.wait()
np.save("myvoice.npy", audio.flatten())
```