RESTful API Conceptual Documentation Sample

This writing sample presents the introduction to API documentation for a (fictional) software company, "SoundDate." SoundDate is a dating app where users match with each other for blind dates based on the sound of each other's voice.

This introduction includes a general overview, common use cases, an architectural overview, an architectural overview, authentication and authorization, and tutorials with sample code.

SoundDate API Documentation

Overview

Welcome to the SoundDate API, which allows developers to integrate SoundDate's unique voice-based dating system into their own applications. SoundDate connects users through voice messages, letting them match for blind dates based solely on the sound of each other's voices. With our API, external developers can build creative tools to enhance the SoundDate experience, or use SoundDate voice messages and matchmaking in their own apps.

Use Cases

1. **Profile Management**: Allow users to create, manage, and update profiles across multiple dating apps.
2. **Matching & Messaging**: Enable audio-based matching and messaging between users on your app.
3. **Analytics & Recommendations**: Create personalized recommendations or insights for SoundDate users via user interaction data.
4. **Audio Content Moderation**: Integrate audio moderation capabilities for safer user experiences on your app.

Requirements

- SoundDate developer account
- HTTPS-compliant HTTP client

- JSON, RESTful API proficiency (interactions are JSON-based)

Key Concepts

- **User Profile**: SounDate's central data structure that holds a user ID, basic details, preferences, and audio files.
- **Match**: A connection between two users based on swiping history.
- **Audio Messages**: The voice messages users send to each other before deciding to meet in person.

Architecture

SoundDate utilizes RESTful principles to manage and maintain user data, audio files, and matchmaking algorithms.

Data Model & Workflow Diagrams

1. Data Model

2. Workflow

`User Authentication → Profile Creation → Audio Upload → Match Search → Match and Messaging → Ongoing Interaction`

Getting Started

To get started with the SoundDate API, follow the steps below.

1. **Registration**

   - Sign up for a SoundDate developer account on our Developer Portal (https://developer.sounddate.com).

   - Complete the registration process and verify your email to activate the account.

2. **Getting an App Key**

   - Log into the Developer Portal.

   - Navigate to the **App Management** section.

   - Create a new application and you will receive a unique **App Key**. This key will authenticate requests to the SoundDate API.

3. **Authorization**

   - All API requests require an authorization header with your App Key:

     ```
     Authorization: Bearer YOUR_APP_KEY
     ```

   - Use HTTPS for all requests to ensure secure data transmission.

Tutorials

Tutorial 1: Creating a User Profile

1. Authenticate with your App Key.

2. Use the **POST /user** endpoint to create a new user profile.

   - Required fields: username, email, preferences.

3. You’ll receive a unique User ID in the response.

Tutorial 2: Uploading an Audio File

1. With your User ID, authenticate and send an audio file to the **POST/user/{userId}/audio** endpoint.

2. Ensure your audio file meets specifications (MP3, WAV, max size 5MB).

3. The response will include the Audio ID and the link to your uploaded file.

Tutorial 3: Initiating a Post-Match Connection

1. Use the **GET /matches?userId={yourUserId}** endpoint to retrieve previous matches.

2. Send a **POST** request to **/matches/{matchId}/connect** to initiate a connection.

3. The system will notify the matched user, and you can start messaging if they accept.

Sample Code

Below is a sample Python script that demonstrates creating a user profile.

```python
import requests

API_BASE = "https://api.sounddate.com/v1"
APP_KEY = "YOUR_APP_KEY"

headers = {
    "Authorization": f"Bearer {APP_KEY}",
    "Content-Type": "application/json"
}

# 1. Create a User Profile
def create_user(username, email, preferences):
    url = f"{API_BASE}/user"
    data = {
        "username": username,
        "email": email,
        "preferences": preferences
    }
    response = requests.post(url, headers=headers, json=data)
    user_data = response.json()
    return user_data.get("userId")
```