# 🎧Automatic Audio Labeling for Daily Life Event Detection

## Product Overview

1. Goal:

Upload or stream daily-life audio and automatically get a labeled event timeline that supports keyword search and direct playback.

## Main Scenarios:

* + Home security or caregiving (detect doorbell, crying, pet sounds, etc.)
  + Daily life logging or behavior tracking
  + Smart home acoustic analysis

### Key Metrics (Goals):

### TBD

## Technical Solution

🚀 Minimum Viable Deployment Steps

1. Audio Upload & Preprocessing

Allow users to upload WAV/MP3 files via a web interface. Use librosa to resample to 16kHz mono and normalize volume. Skip advanced denoising or silence trimming for now.

1. Run Pretrained Model

Use Google’s **YAMNet** pretrained model (from TensorFlow Hub). Feed audio input → get frame-level class probabilities → extract top event labels.

1. Basic Postprocessing

Apply a simple threshold (e.g., >0.5) to frame probabilities, merge consecutive frames into segments, and output event labels with timestamps.

1. Result Visualization

Display detected events and timestamps (e.g., “Dog bark 0:15–0:17”) on a web page using a simple table or interface.

1. Basic API

Use FastAPI for backend:   
• POST /upload: Upload audio → return detection results (JSON).  
• Skip database; store results in memory or temporary files.

1. Deployment & Run

Run locally or deploy with Streamlit or FastAPI + Uvicorn. Keep the YAMNet model loaded persistently to avoid reload delays.

Full Technical Solution（Optional）

1. Preprocessing

Normalize sampling rate (16kHz mono), perform denoising, loudness normalization, silence detection, and segmentation for long recordings. Tools: librosa, pydub, noisereduce.

1. Feature Extraction

Extract traditional features (MFCC, Mel-spectrogram) or use pretrained models (YAMNet/VGGish) to obtain deep embeddings and frame-level event probabilities.

1. Event Classification

Use YAMNet to predict 521 audio event classes per frame and map them to a custom set of household-related labels (e.g., “doorbell”, “speech”, “dog bark”).

1. Event Localization & Aggregation

Smooth frame-level probabilities, apply thresholds, and merge consecutive frames into event segments. Each event is defined by start time, end time, and confidence.

1. Indexing & Retrieval

Store results in a database (SQLite). Separate tables for recordings and events. Support queries by event label or time range, returning segment timestamps.

1. API Design

Build RESTful API with FastAPI.  
• POST /upload: Upload audio  
• GET /recordings/{id}/events: Retrieve event timeline  
• GET /search?label=: Search by event label

1. Performance & Scalability (Optional)

Keep the model loaded in memory to avoid reloads; process long recordings asynchronously (Celery); parallel inference for segments; cache results for faster response

## Tech Stack

| **Category** | **Tools / Libraries** |
| --- | --- |
| **Audio Processing** | librosa, pydub, soundfile, noisereduce |
| **Model & Feature Extraction** | TensorFlow Hub (YAMNet), VGGish, scikit-learn |
| **Post-processing & Analysis** | numpy, pandas, scipy, custom aggregation algorithms |
| **Backend & API** | FastAPI, SQLite, Celery, Uvicorn |
| **Visualization & Debugging** | matplotlib, streamlit, or a frontend waveform player |