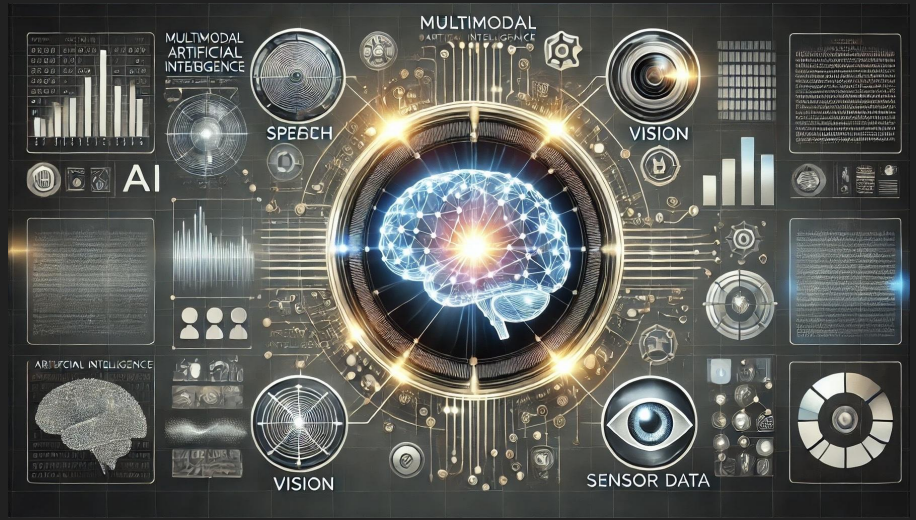
LLM-Enhanced Multimodal AI:

Revolutionizing Audio Interaction Technologies



Conf42.com Large Language Models (LLMs) 2025

PRESENTER:

WASEEM SYED

The Challenges of Audio Navigation

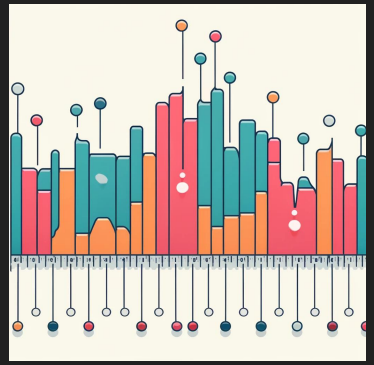
Content Overload: Increased volume of podcasts, audiobooks, and courses

Navigation Difficulties: Inefficient search within audio

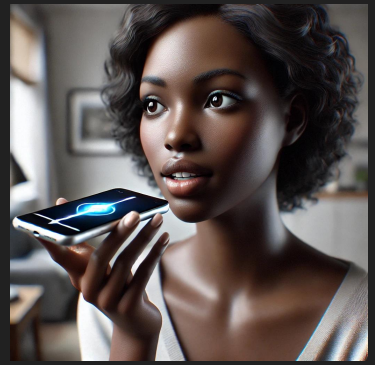
User Needs: Desire to access relevant segments quickly

AI-Powered Solution Overview

Speaker Diarization
Who spoke and when



Topic Segmentation
Classification of topics



Multimodal Search
Text, Voice, and visual inputs for interactive navigation

Speaker Diarization

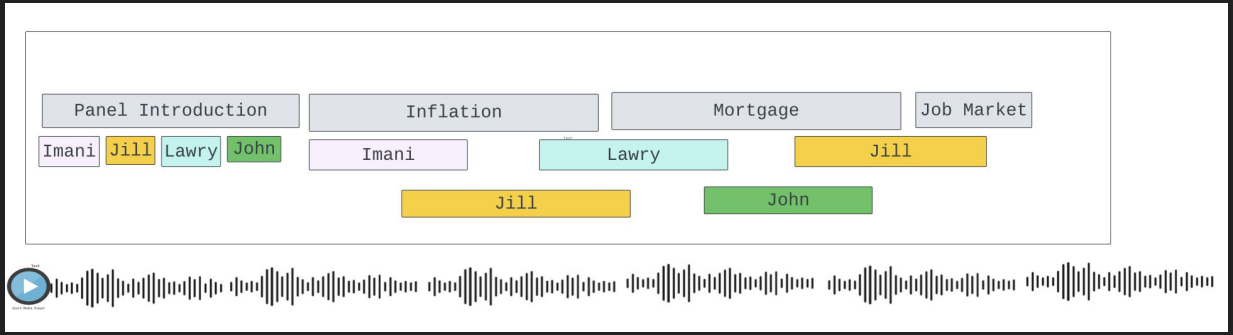
Identifying “**who** spoke **when**”

- Example: “*Watch segments where Jill spoke* ”

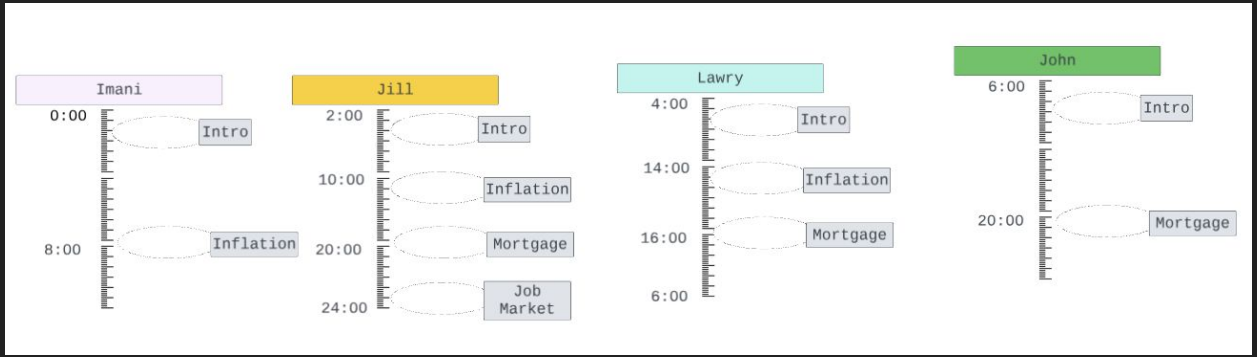
Reducing Diarization Error Rate (DER)

Dynamic Speaker Profiles and Metrics

Audio Stream



Speaker Classification



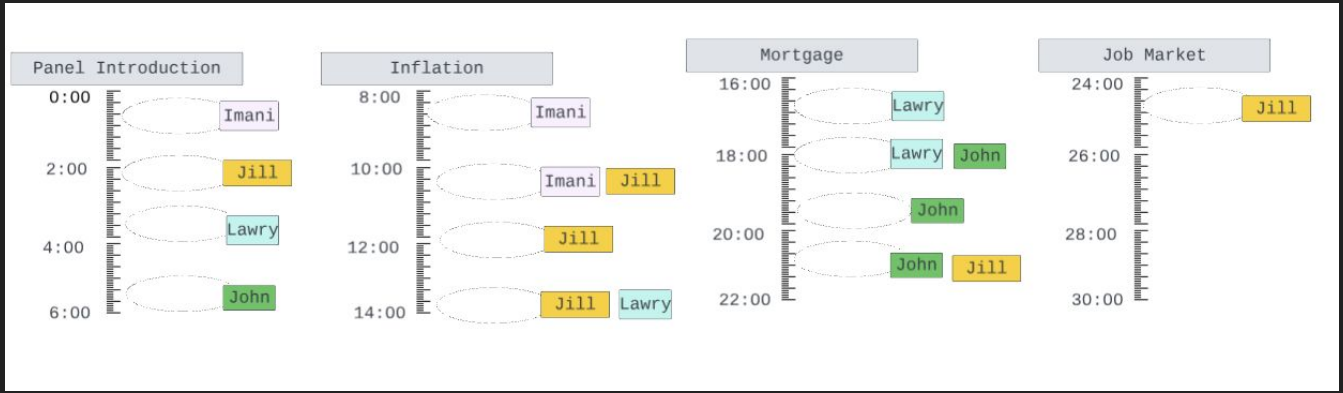
Topic Segmentation

Breaking down audio into meaningful **segments**

Using NLP techniques (Cosine Similarity, TF-IDF)

Interactive topic-based indexing

- Example: “*Find all discussion on Inflation*”



MultiModal Search Interface

Voice-based querying

Digital Inclusion through **Accessibility**

AI-generated contextual responses and indexing

Example: Asking 🗣️
“*What did Lawry say about Inflation?*”

AI-Driven Advanced Content Indexing

Dynamic annotations

Related topics links

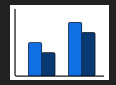
Integrated note-taking and bookmarking

Structured retrieval of audio segments

Example: *Indexing educational lectures for quick navigation*

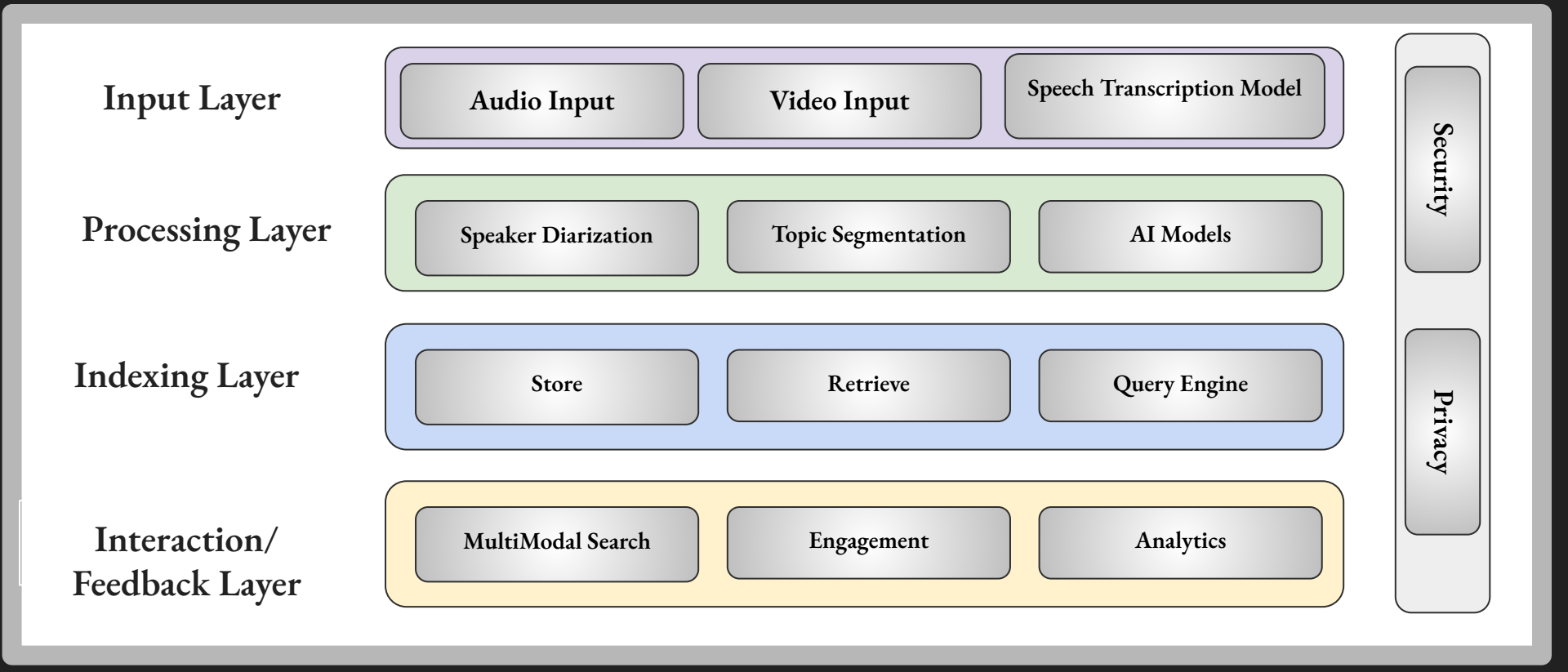
User Engagement and Feedback

Ratings and comments on topics 👍 👎 

Real-time analytics for content creators 

User-behavior driven content refining

System Architecture - Overview



Input Layer - Audio Processing

Goal: Convert raw audio into text with timestamps

Technology Used: Open AI Whisper

Functionality:

- Speech-to-text conversion
- Word-level timestamp generation
- Language Detection

Example Query:

“Convert this podcast episode to text and identify timestamps for each word”

Example Code(Pseudo Code)

```
def transcribe():
    # Get the audio file path from the request
    data = request.get_json()
    audio_file_path = data.get('audio_file_path')

    # Open the audio file
    try:
        with open(audio_file_path, 'rb') as audio_file:
            # Call OpenAI API to create the transcription
            transcript = client.audio.transcriptions.create(
                file=audio_file,
                model="whisper-1",
                response_format="verbose_json",
                timestamp_granularities=["word"]
            )
    except FileNotFoundError:
        return {"error": "File not found."}, 404
    except Exception as e:
        return {"error": str(e)}, 500

    return transcript
```

Sample Response


```
{
  "duration": 300.5,
  "text": "Welcome to our podcast. Today we discuss have on the panel...",
  "language": "english",
  "words": [
    {"start": 0.5, "end": 1.2, "word": "Welcome"},
    {"start": 1.3, "end": 2.0, "word": "to"},
    {"start": 2.1, "end": 2.5, "word": "our"},
    {"start": 2.6, "end": 3.5, "word": "podcast"}
  ]
}
```

Processing Layer - Speaker Diarization

Goal: Identify who spoke when using LLM-based diarization

Technology Used: Open AI Whisper + GPT4 or similar

Functionality:

- Uses timestamps and words from Input Layer
- Detect speaker changes based on pauses and content
- Uses LLMs to infer and assign speaker labels

Example Query:

“Who spoke in each segment of this podcast?”

Example Code(Pseudo Code)

```python
import openai

# Step 1: Get words & timestamps from Whisper API
whisper_transcription = get_whisper_transcription(audio_file)

# Step 2: Chunk transcript based on pauses (>2 sec = speaker change)
segments = chunk_transcript(whisper_transcription["words"])

# Step 3: Format transcript for LLM speaker identification
llm_prompt = format_for_llm(segments)

# Step 4: Call LLM (GPT-4 Turbo) to infer speakers
def infer_speakers(prompt):
    response = openai.ChatCompletion.create(
        model="gpt-4-turbo",
        messages=[
            {"role": "system", "content": "You are an expert in speaker diarization."},
            {"role": "user", "content": prompt}
        ]
    )
    return response["choices"][0]["message"]["content"]

speaker_diarization_result = infer_speakers(llm_prompt)

# Step 5: Parse and return structured output
diarized_output = parse_llm_response(speaker_diarization_result)
```

Sample Response


```
{
    [
        {
            "speaker": "Speaker 1",
            "segments": [
                {
                    "start": 0.5,
                    "end": 3.0,
                    "text": "Hello everyone Welcome to our Podcast"
                },
                {
                    "start": 10.0,
                    "end": 15.0,
                    "text": "Today we have on our panel..."
                }
            ]
        },
        {
            "speaker": "Speaker 2",
            "segments": [
                {
                    "start": 12.0,
                    "end": 19.5,
                    "text": "Thank you for having me...."
                }
            ]
        }
    ]
}
```

Processing Layer - Topic Segmentation

Goal: Identify topic boundaries and segment content into meaningful sections

Technology Used: NLP Techniques + LLM

Functionality:

- Segments transcript using timestamps from Whisper API
- Detects Topic Shifts in conversation
- Categorizes segments based on themes
- Uses LLMs to assign topic labels

Example Query:

“Find all sections discussing AI ethics in this podcast”

Example Algorithm(Pseudo Code)

Step1: Chunk Transcript into Time Intervals

Step2: Compute Text Similarity

Step3: Detect Topic Shifts

Step4: Assign Topic Labels Using LLM

Step5: Output Structured JSON

Sample Response

```
[
  {"topic": "Introduction to AI", "start": 0, "end": 50},
  {"topic": "AI in Healthcare", "start": 51, "end": 120},
  {"topic": "Ethical Challenges", "start": 121, "end": 180}
]
```

Indexing Layer - Store/Retrieve

Goal: Store and Retrieve Structured Audio Segments efficiently

Technology Used: Database indexing, full-text search, caching

Functionality:

- Stores speaker-labeled and topic-segmented data
- Fast Retrieval via topic, speaker and timestamps
- Optimized for real-time search and navigation

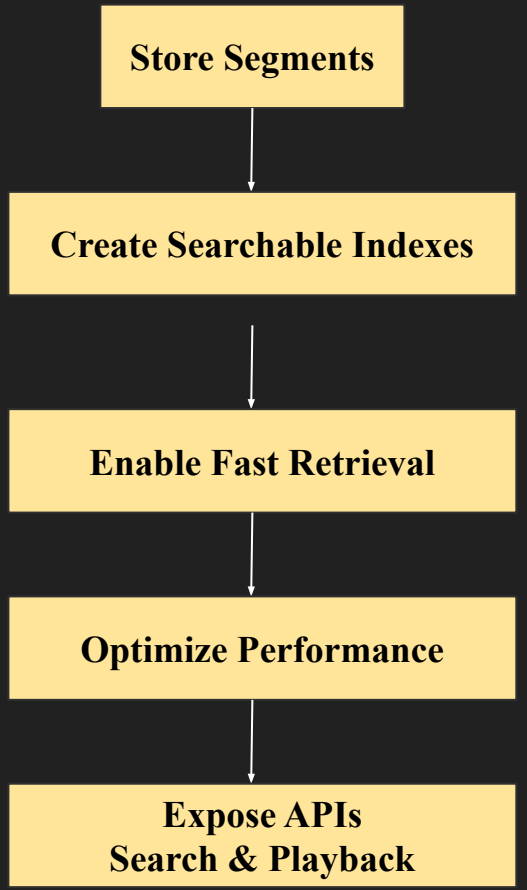
Example Query:

“Find all sections discussing AI ethics on last week’s podcast”

Indexing Layer - Store/Retrieve

Input:
Data(topics, speakers, start, end etc) from Processing Layer

Output:
Indexed Storage enabling faster search/indexing



Interaction/Feedback Layer

Goal: Enable Intuitive Search, Playback and Feedback collection

Technology Used: Speech Transcription Model, Real-time Analytics, AI

Functionality:

- Multimodal interaction(voice, text, UI-based search)
- Real-time feedback for improved AI accuracy
- Personalized Recommendations

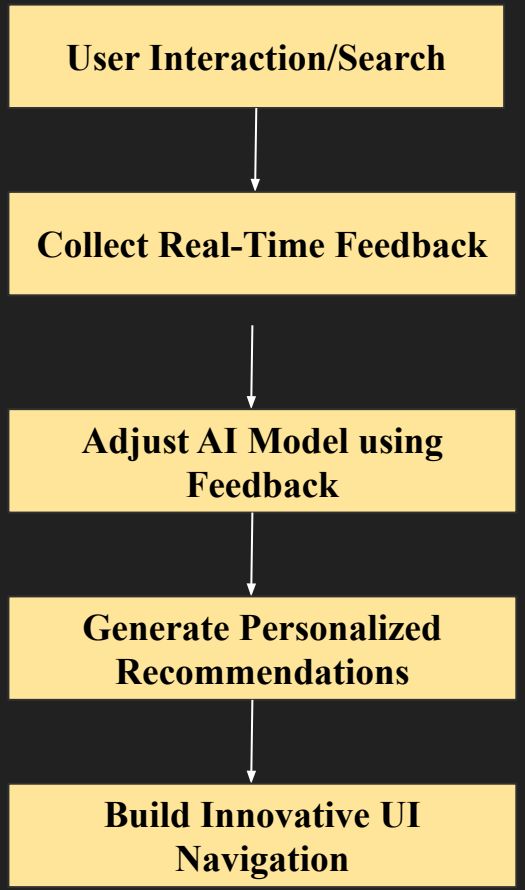
Example Query:

“What did John say about security in AI”

Interaction/Feedback Layer

Input:
User queries, playback interactions, feedback ratings

Output:
Personalized search, improved AI accuracy, and enhanced recommendations



Summary

Passive Listening -> Interactive Experience 🗣️ 🎤

Efficient Topic and Speaker Detection 🔎

Real-time Feedback and Continuous Improvement 📈📉📊

Scalable Framework ✅

Improved Accessibility 👂

Conclusion

More than **45% audio platforms** have an opportunity to benefit from the AI-powered navigation 🚀

Audio Source	Time Spent Listening (%)
Streaming Music (Spotify, Pandora, Apple Music, Amazon Music etc.)	18
Podcasts	10
YouTube Music/Music Videos	14
Audiobooks	3
AM/FM	36
Sirius XM	8
TV Music Channels	3
Owned Music (CDs, DVDs, music files etc.)	7
Other	1

Courtesy: Edison Research: Share of Ear(US Population 13+)

THANK YOU