TalkNShop

Media Service and MCP Integration

Design Specification

Version 0.1

Puneet Bajaj
(puneet.bajaj@sjsu.edu)

Version History

Version	Changes
0.1	Design details of Media service and current implementation status

Introduction

The Media Service is a core backend component of **TalkNShop**. It processes multimedia content (images and audio) using **AWS AI/ML services** together with **local vision–language models**. It enables multimodal input for the orchestrator by:

- extracting detailed product characteristics from images, and
- transcribing audio content to text.

The service is accessible **both** via **REST API** and via an **MCP (Model Context Protocol) server**, enabling:

- direct REST calls when media processing is explicitly required, and
- AI-driven tool selection in orchestration workflows via MCP tools.

References

AWS Transcribe documentation for speech-to-text conversion
AWS Rekognition documentation for image analysis
AWS S3 documentation for media file storage
Ollama documentation for local LLM inference
Model Context Protocol (MCP) specification for tool-based AI integration
TalkNShop System Architecture (AWS API Gateway, ALB, Orchestrator-Service integration)
FastAPI documentation for async web framework

Requirements

Functional Requirements

- Accept base64-encoded media files (images/audio) from orchestrator or client
- Process audio files using AWS Transcribe for speech-to-text conversion
- Analyze images using AWS Rekognition for labels, text, and object detection
- Extract enhanced product characteristics from images using vision-language model (Ollama LLaVA)

- Determine product type automatically from image analysis (shoe, clothing, bottle, electronics, etc.)
- Return structured analysis results with confidence scores and metadata
- Support batch processing for multiple media files
- Generate presigned S3 URLs for direct media uploads
- Extract characteristics from audio transcripts using LLM analysis
- Provide health check endpoint for service monitoring
- Return raw AWS results alongside enhanced characteristics for comparison

Non-Functional Requirements

- Response time P95 < 6s for image characteristic extraction (AWS Rekognition ~1-2s, LLM ~4-6s)
- Response time P95 < 15s for audio transcription (AWS Transcribe async processing)
- Graceful fallback when Ollama LLM unavailable (AWS-only extraction with logging)
- Secure credential management via environment variables/.env
- File size validation (max 50MB) and format validation (JPEG/PNG/WebP for images, MP3/WAV/M4A/FLAC for audio)
- Structured logging with request correlation IDs
- Future: Result caching, rate limiting, automated health checks

Functional Overview

MCP Integration

The Media MCP Server is an adapter between Orchestrator and Media Service.

Purpose

- Invoke media processing as **MCP tools**

Implementation

- MCP server exposes media analysis functions

Benefits

- Standardized orchestrator↔media contract
- Tool-based invocation model (extract_image_characteristics, transcribe_audio, etc.)
- Seamless multimodal processing in workflows

MCP Tools (Completed)

- extract_image_characteristics — Vision LLM-based characteristic extraction

- analyze_image — Basic Rekognition (labels/text/objects)
- transcribe_audio — AWS Transcribe
- extract_audio_characteristics — Extract traits from transcripts
- upload_media — Presigned S3 URLs=

Media Service API

FastAPI microservice (async) specialized for multimodal content:

Endpoints

- GET /health — health probe
- POST /api/v1/transcribe — base64 audio → transcript
- POST /api/v1/analyze-image — Rekognition analysis
- POST /api/v1/extract-characteristics — vision LLM–enhanced extraction
- POST /api/v1/extract-audio-characteristics — traits from transcript
- POST /api/v1/upload — generate presigned S3 URLs
- POST /api/v1/analyze-image/batch — batch images
- POST /api/v1/transcribe/batch — batch audio

Internal Flow

1. Request via REST or MCP
2. Validate media (format, size, encoding)

Image

- Decode base64 → bytes
- Rekognition: **labels / text / objects**
- Derive **product type** from labels
- If Ollama available: pass image to **vision LLM (LLaVA)** for enhanced traits
- Parse LLM response → structured characteristics (brand, color, material, style, …)
- Return **combined** (AWS + LLM) result

Audio

- Decode base64
- Upload to **S3**
- Start **Transcribe** job (language, speakers)
- Poll until complete
- Retrieve transcript + confidences
- If Ollama available: extract traits from transcript
- Return transcript + extracted characteristics

MCP Integration Flow

1. Orchestrator detects multimodal input
2. Selects appropriate MCP tool
3. MCP server receives invocation (media + params)
4. MCP server calls Media REST endpoint
5. Media service processes request
6. MCP server formats response to tool schema
7. Orchestrator consumes structured response in reasoning pipeline

Configuration/ External Interfaces

Environment Variables

```
AWS_REGION=us-west-1
AWS_ACCESS_KEY_ID=<AWS access key>
AWS_SECRET_ACCESS_KEY=<AWS secret key>
S3_BUCKET_NAME=talknshop-media-storage
OLLAMA_MODEL=llava:7b
OLLAMA_HOST=http://localhost:11434
MAX_FILE_SIZE=52428800
ALLOWED_AUDIO_FORMATS=mp3,wav,m4a,flac
ALLOWED_IMAGE_FORMATS=jpg,jpeg,png,webp
DEBUG=false
LOG_LEVEL=INFO
```

External APIs

- **AWS Transcribe**
  - StartTranscriptionJob, GetTranscriptionJob, retrieve results (S3)
- **AWS Rekognition**
  - DetectLabels, DetectText, **objects with bounding boxes**
- **AWS S3**
  - PutObject, GeneratePresignedUrl, GetObject, DeleteObject
- **Ollama (Local)**
  - POST /api/chat (vision inference, base64 image)
  - GET /api/tags (list models)

- **MCP Server (Media)**
    - Tools: extract_image_characteristics, analyze_image, transcribe_audio, extract_audio_characteristics, upload_media
    - Tool invocation protocol & standardized responses

Internal Interfaces

- **Orchestrator → Media (MCP):** tool invocations
- **Orchestrator → Media (REST):**
    - POST /api/v1/extract-characteristics (ImageAnalysisRequest)
    - POST /api/v1/transcribe (AudioTranscriptionRequest)
- **Media → Orchestrator/UI:**
    - ExtractedCharacteristics (item_type, primary_item, characteristics[])
    - AudioTranscriptionResponse (transcript, confidence, speakers)

MCP Server Interface

- Exposes media tools wrapping REST endpoints
- Orchestrator **selects** tools based on input type
- Standardized **response** for orchestration pipeline

Debug

Planned debug endpoints

- GET /api/v1/debug/rekognition — raw Rekognition payload
- GET /api/v1/debug/ollama — raw Ollama response
- GET /api/v1/debug/transcribe — raw Transcribe job details

Logging

- **INFO:** media received, AWS call start/end, extraction method, confidence scores
- **ERROR:** AWS failures (status, message), LLM parsing errors, file validation failures
- **DEBUG:** raw LLM responses, parsing steps, item-type detection logic (if DEBUG=true)
- **Planned:** request correlation IDs, structured JSON logs

Sample

```
INFO  Starting image analysis for item type: bottle
```

```
INFO  Rekognition: 15 labels, 2 text detections, 1 object
INFO  Ollama vision extraction: 10 characteristics
ERROR Ollama connection failed: Connection refused -- fallback to AWS-only
ERROR Rekognition failed: InvalidImageFormatException
```

Counters

- Images processed by analysis type (labels/text/objects)
- Audio transcription: started vs completed
- Characteristic extraction attempts (LLM vs AWS-only)
- Ollama availability status
- AWS service latency per provider (Rekognition, Transcribe)
- File validation failures (size, format)

Implementation

Design Notes

- **FastAPI** + **uvicorn** (async)
- **boto3** for Rekognition/Transcribe/S3
- **Ollama** Python client for local LLM
- CharacteristicExtractor for vision LLM + parsing
- AudioCharacteristicExtractor for transcript analysis
- **Unified models** (models.py)
    - ImageAnalysisRequest/Response
    - AudioTranscriptionRequest/Response
    - Characteristic (name, value, confidence, category)
    - ExtractedCharacteristics (item_type, primary_item, characteristics[])
    - ItemType enum (shoe, clothing, bottle, electronics, furniture, bag, watch, jewelry, book, toy, unknown)
- **AWS wrappers** (aws_services.py)
    - RekognitionService: detect_labels, detect_text, detect_objects, analyze_image
    - TranscribeService: start_transcription_job, wait_for_completion, get_transcription_results
    - S3Service: upload_file, generate_presigned_url, download_file, delete_file
- **Vision LLM integration**
    - Model: **LLaVA 7B**
    - Structured prompts; parse numbered lists → Characteristic

    - Fallback to AWS-only extraction if LLM unavailable
- Item-type detection via label keyword matching
- Robust error handling with AWSServiceError and graceful degradation
- **MCP server** wraps REST endpoints as tools

Subtasks

- Rekognition client (**Completed**)
- Transcribe client (**Completed**)
- S3 client (**Completed**)
- Characteristic extractor + Ollama (**Completed**)
- Item-type detection (**Completed**)
- Vision LLM response parsing (**Completed**)
- Image characteristic endpoint (**Completed**)
- Audio transcription endpoint (**Completed**)
- Audio characteristic endpoint (**Completed**)
- Batch endpoints (**Completed**)
- Health check (**Completed**)
- Presigned S3 URL generation (**Completed**)
- MCP server integration & tools (**Completed**)
- Debug endpoints (**Planned – Sprint 10**)
- Result caching (**Planned – Sprint 10**)
- Rate limiting & retries (**Planned – Sprint 10**)

Testing

General Approach

- Manual **curl/Postman** tests (image extraction, audio transcription)
- Schema compliance, error handling, multimodal processing
- E2E with orchestrator
- Controlled failures: invalid keys, timeouts, **Ollama** down, oversized files

Unit Tests (Planned)

- Item-type detection logic
- Vision LLM response parsing (structured vs free text)
- Characteristic extraction (LLM vs AWS-only)
- Error scenarios (no Ollama, invalid image, AWS failures)

Integration Tests (Planned)

- E2E image analysis (/api/v1/extract-characteristics)
- E2E audio transcription (/api/v1/transcribe)
- Ollama connection + fallback behavior
- AWS integration (Rekognition, Transcribe, S3)
- Batch processing
- MCP tool invocation from orchestrator (extract, transcribe)
- MCP schema validation and response formats

Functional Tests

Image: Owala water bottle (JPEG, base64)

- Expect item_type="bottle"; characteristics include brand/color/material
- extraction_method is "ollama_vision_enhanced" or "aws_only"
- Confidence scores in **0–1**; AWS results included

Image: Adidas shoes (JPEG, base64)

- Expect item_type="shoe"; brand "Adidas", style, condition; primary item detected

Audio: “I want a red running shoe under 100 dollars”

- Transcript matches text; characteristics extracted; confidence > **0.8**

Invalid file (AVI) → **400** with supported formats

>50 MB file → **400** size limit message

Appendix

Sample Response — Image Characteristic Extraction

```
{
  "analysis_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "completed",
  "item_type": "bottle",
  "primary_item": "Water Bottle",
  "characteristics": [
    {"name": "Brand/Manufacturer", "value": "Owala", "confidence": 0.95,
"category": "brand"},
    {"name": "Color", "value": "Blue, Orange", "confidence": 0.90,
"category": "color"},
    {"name": "Material", "value": "Plastic", "confidence": 0.85,
"category": "material"},
    {"name": "Style", "value": "Insulated, Travel", "confidence": 0.80,
"category": "style"},
    {"name": "Features", "value": "Leakproof lid, Insulation",
"confidence": 0.85, "category": "features"},
    {"name": "Use Case", "value": "Travel, Workout", "confidence": 0.80,
"category": "use_case"},
    {"name": "Target", "value": "Health-conscious consumers", "confidence":
0.75, "category": "target"},
    {"name": "Price Range", "value": "Mid-range", "confidence": 0.70,
"category": "price_range"}
  ],
  "extraction_method": "ollama_vision_enhanced",
  "confidence_score": 0.85,
  "aws_results": {
    "labels": [
      {"name": "Bottle", "confidence": 100.0, "parents": []},
      {"name": "Water Bottle", "confidence": 99.8775634765625, "parents":
["Bottle"]},
      {"name": "Shaker", "confidence": 97.27078247070312, "parents":
["Bottle"]}
    ],
    "text_detections": [
      {
        "text": "owala",
        "confidence": 90.63531494140625,
        "bounding_box": {"left": 0.4140625, "top": 0.2265625, "width":
0.0869140625, "height": 0.0263671875}
      }
    ],
    "objects": [
```

```
      {
        "name": "Shaker",
        "confidence": 97.27078247070312,
        "bounding_box": {"left": 0.37248289585113525, "top":
0.055694159120321274, "width": 0.26166456937789917, "height":
0.8676441311836243}
      }
    ]
  },
  "processing_time": 6.234
}
```

Sample Response — Audio Transcription

```
{
  "transcription_id": "550e8400-e29b-41d4-a716-446655440001",
  "status": "completed",
  "transcript": "I want a red running shoe under 100 dollars",
  "confidence": 0.95,
  "speakers": [
    {
      "speaker": "spk_0",
      "start_time": 0.0,
      "end_time": 3.2,
      "text": "I want a red running shoe under 100 dollars"
    }
  ],
  "processing_time": 12.5
}
```

API Request/Response Schemas

ImageAnalysisRequest

```
{
  "image_file": "<base64_encoded_image>",
  "analysis_types": ["labels", "text", "objects"],
  "max_labels": 20,
```

```
  "min_confidence": 0.5
}
```

AudioTranscriptionRequest

```
{
  "audio_file": "<base64_encoded_audio>",
  "language_code": "en-US",
  "speaker_count": 2,
  "vocabulary_name": null
}
```

Characteristic Data Structure & Categories

Characteristic

```
{ "name": "Brand/Manufacturer", "value": "Owala", "confidence": 0.95,
"category": "brand" }
```

Categories

- **brand** — Brand/Manufacturer
- **color** — Primary/secondary colors
- **material** — Plastic, leather, fabric, etc.
- **size** — Dimensions/size indicators
- **style** — Casual, formal, sporty, etc.
- **condition** — New, used, damaged
- **features** — Key product features
- **use_case** — Primary use case(s)
- **target** — Target audience
- **price_range** — Budget / mid-range / premium

MCP Tool Schema Examples

extract_image_characteristics

```
{
```

```
  "name": "extract_image_characteristics",
  "description": "Extract detailed product characteristics from images
using vision-language model.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "image_file": { "type": "string", "description": "Base64 encoded
image file" },
      "analysis_types": { "type": "array", "items": {"type": "string"},
"default": ["labels", "text", "objects"] },
      "max_labels": { "type": "integer", "default": 20 },
      "min_confidence": { "type": "number", "default": 0.5 }
    },
    "required": ["image_file"]
  }
}
```

transcribe_audio

```
{
  "name": "transcribe_audio",
  "description": "Transcribe audio file to text using AWS Transcribe.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "audio_file": { "type": "string", "description": "Base64 encoded
audio file" },
      "language_code": { "type": "string", "default": "en-US" },
      "speaker_count": { "type": "integer" },
      "vocabulary_name": { "type": "string" }
    },
    "required": ["audio_file"]
  }
}
```

analyze_image

```
{
  "name": "analyze_image",
  "description": "Basic image analysis using AWS Rekognition.",
```

```
  "inputSchema": {
    "type": "object",
    "properties": {
      "image_file": { "type": "string", "description": "Base64 encoded
image file" },
      "analysis_types": { "type": "array", "items": {"type": "string"},
"default": ["labels", "text", "objects"] },
      "max_labels": { "type": "integer", "default": 10 },
      "min_confidence": { "type": "number", "default": 0.7 }
    },
    "required": ["image_file"]
  }
}
```

extract_audio_characteristics

```
{
  "name": "extract_audio_characteristics",
  "description": "Extract product characteristics from audio transcript
using LLM analysis.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "audio_file": { "type": "string", "description": "Base64 encoded
audio file" },
      "language_code": { "type": "string", "default": "en-US" },
      "speaker_count": { "type": "integer" }
    },
    "required": ["audio_file"]
  }
}
```