Azure Speech Gateway API

Text-to-Speech (primary) + Speech-to-Text (bonus)
Hosted on DigitalOcean · Powered by Azure AI Speech

1. Overview

The **Azure Speech Gateway API** is a simple REST service that sits between clients and **Azure AI Speech**.

- Clients call **this** API (hosted on a DigitalOcean droplet).
- The gateway then calls **Azure Text-to-Speech** and **Speech-to-Text** behind the scenes.
- Clients do **not** need an Azure account or knowledge of Azure's SDKs, tokens, or regions.

This project is part of **ITIS 6177 (System Integration)** and demonstrates:

- Integration with a third-party cloud API (Azure AI Speech)
- Clean REST API design and error handling
- Basic observability and deployment on a Linux server

2. Intended Audience

2.1 Business Stakeholders

Use this API if you want to:

- Turn product messages, alerts, or content into natural-sounding audio
- Prototype voice features (voice prompts, announcements, etc.) without managing Azure directly

Key benefits:

- No Azure account or portal configuration required for consumers
- Only the gateway owner controls Azure usage and billing
- A single, documented endpoint for text-to-speech and speech-to-text

2.2 Developers

You get:

- Simple REST endpoints:

    - `POST /api/v1/speech/synthesize` (Text-to-Speech)
    - `POST /api/v1/speech/transcribe` (Speech-to-Text)

- JSON in, **audio out** (TTS) or **JSON out** (STT)
- Standard HTTP methods and status codes
- Easy to test in Postman or integrate into any backend/frontend

2.3 Architects

Architecture highlights:

- Pattern: **API Gateway / Backend for Frontend** around Azure AI Speech
- The service is **stateless**, so it can be scaled horizontally on DigitalOcean
  Clear separation of concerns:

    - Gateway: auth (if enabled), validation, mapping to Azure
    - Clients: only business logic and UX

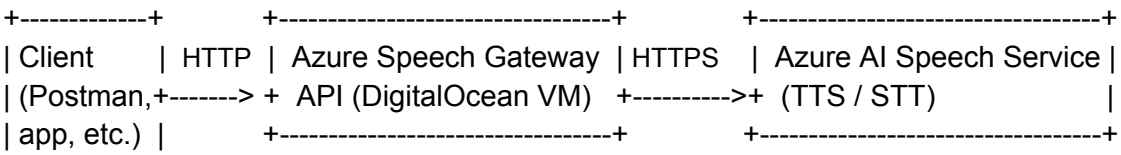
If needed in the future, the backend could swap Azure for another provider (e.g., Google Cloud TTS) with minimal changes for clients.

3. System Overview

High-level flow:

1. Client sends a request to the gateway API (TTS or STT).
2. Gateway validates input and builds the correct Azure request.
3. Gateway calls Azure AI Speech over HTTPS.
4. Azure returns audio (for TTS) or text (for STT).
5. Gateway returns the result to the client.

Suggested architecture diagram for the repo:



4. Live API & Base URL

Base URL (current deployment)

Fall-2025 ITIS-6177 System Integration **Shipra(801430071)**

http://159.89.184.113:3000/api/v1

Endpoints:

- `POST /speech/synthesize` – Text-to-Speech (primary)
- `POST /speech/transcribe` – Speech-to-Text (bonus)

Example full URLs:

- `http://159.89.184.113:3000/api/v1/speech/synthesize`
- `http://159.89.184.113:3000/api/v1/speech/transcribe`

5. Quickstart (Postman Demo)

5.1 Text-to-Speech in One Request

Request

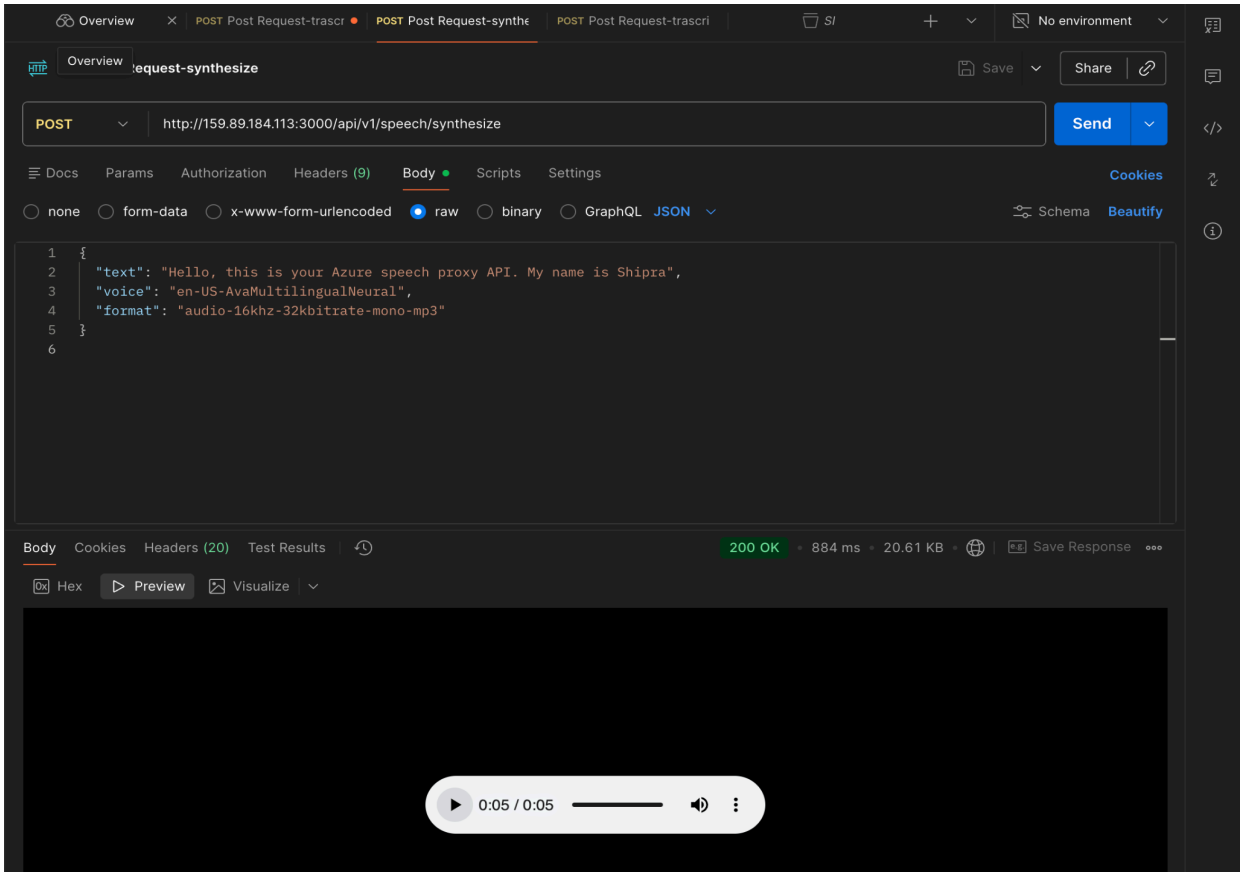
- Method: `POST`
- URL: `http://159.89.184.113:3000/api/v1/speech/synthesize`
- Headers:
    - `Content-Type: application/json`

Request (raw JSON)

```
{
  "text": "Hello, this is your Azure speech proxy API. My name is Shipra",
  "voice": "en-US-AvaMultilingualNeural",
  "format": "audio-16khz-32kbitrate-mono-mp3"
}
```

Response

- Status: `200 OK`
- Content-Type: `audio/mpeg` (or equivalent audio type)
- Body: binary audio stream (you can play or save it from Postman).



5.2 Speech-to-Text (Extra)

Request

- Method: `POST`
- URL: `http://159.89.184.113:3000/api/v1/speech/transcribe`
- Headers:
  - `Content-Type: multipart/form-data`

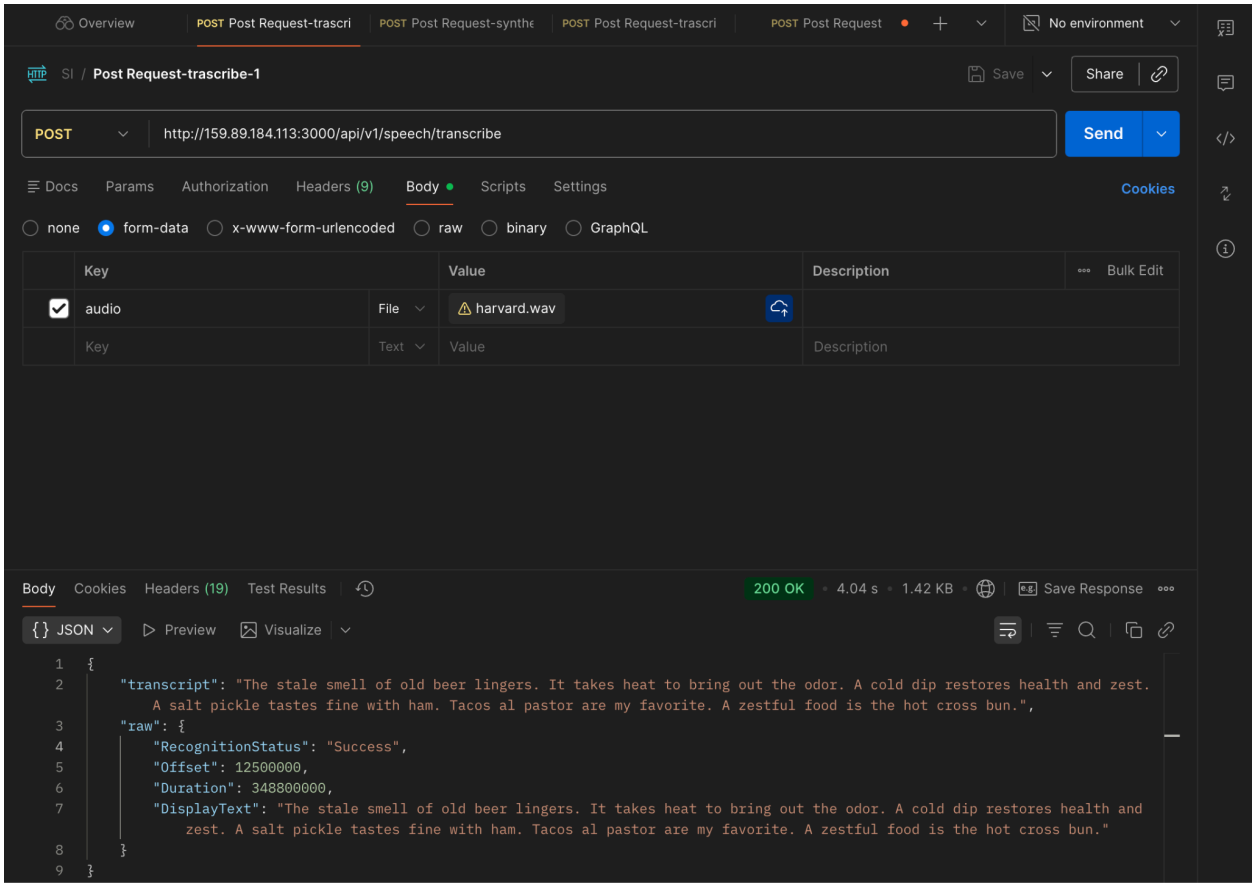
Body (form-data)

- Key: `file`
- Type: **File**
- Value: upload a short `.wav` or `.mp3` audio file

Example Response (JSON)

{
"transcript": "Add in soil to plant spring corn.",
"raw": {

```
  "RecognitionStatus": "Success",
  "Offset": 700000,
  "Duration": 57800000,
  "DisplayText": "Add in soil to plant spring corn."
 }
}
```

- `transcript` is the convenient, high-level text.
- `raw` contains the original Azure STT response for debugging or advanced uses.

6. Authentication & Security

For the class demo, the API is left **open** to simplify grading.

Optionally, it can be secured later using a simple API key:

- Header: `x-api-key: <PUBLIC_GATEWAY_KEY>`
- Requests without a valid key would receive `401 Unauthorized.`
- Azure secrets (e.g., `SPEECH_KEY, SPEECH_REGION`) are stored in environment variables on the server and never exposed to clients or committed to Git.

8. Data Formats & Limits

8.1 Text Input (TTS)

- Encoding: UTF-8
- Supports normal text, punctuation, and emojis
- Recommended max length: e.g., **2000 characters** to protect performance and Azure usage

If the text is too long, the API should reject the request with a **4xx** status and a clear error message.

8.2 Audio Output (TTS)

Common formats (depending on configuration):

- `audio-16khz-32kbitrate-mono-mp3` (MP3)
- Other Azure formats can be mapped as needed via the `format` field.

8.3 Audio Input (STT)

Recommended:

- File types: `.wav` or `.mp3`
- Reasonable duration (e.g., under 30–60 seconds) for faster response and lower cost

If a non-audio file is uploaded, the API returns a **4xx** error with a human-readable explanation.

9. Error Handling

The API uses standard HTTP status codes:

- `400 Bad Request` – invalid input (missing text, invalid file, etc.)
- `401 Unauthorized` – if API key auth is enabled and key is missing/invalid
- `413 Payload Too Large` – if text is too long or file is too large (if enforced)
- `429 Too Many Requests` – optional; rate limiting if configured
- `5xx` – unexpected errors or upstream Azure failures

Typical error response shape (JSON):

{

```
  "error": {
    "code": "STRING_CODE",
    "message": "Human-readable explanation.",
    "details": "Optional technical details for developers."
  }
}
```

Examples:

- `INVALID_INPUT` – missing `text` or `file`
- `INVALID_FILE_TYPE` – non-audio file uploaded
- `VENDOR_ERROR` – Azure returned an unexpected error

10. Deployment & Runtime

- Hosting: **DigitalOcean** Linux droplet
- Runtime: **Node.js** application
- Process management: `pm2` (keeps the API running and restarts on reboot)
- Configuration via environment variables, for example:
    - `SPEECH_KEY`
    - `SPEECH_REGION`
    - `API_KEY` (optional, for gateway-level auth)

The service is stateless, so multiple instances can be added behind a load balancer if needed.