ACS Endpoints

The ACS (Azure Communication Services) integration acts as middleware that allows legacy IVR systems (built on VXML) to interact with modern conversational AI platforms. For this milestone, ACS is implemented as a mock service to simulate request–response flows between the IVR and ACS.

The implementation follows a layered architecture with three main components :

Routes, **Controller**, and **Service**.

acsRoutes.js

- Defines the API endpoints that external systems, such as IVR, can call.
- Directs incoming requests (e.g., /acs/response) to the appropriate controller.
- Contains no logic other than path definitions.
- Analogy: Functions like a reception desk that directs requests to the correct department.

acsController.js

- Serves as the decision-making layer between routes and services.
- Validates incoming data and ensures it is correctly structured.
- Forwards valid requests to the service layer.
- Ensures a consistent JSON response format.
- Analogy: Works like a manager who checks documents, forwards them to the right team, and communicates the result.

acsService.js

- Simulates ACS behavior and generates mock responses for milestone testing.
- Processes the request using the provided sessionId and userInput.
- Returns a structured response including a platform identifier (platform: "ACS") to distinguish ACS replies from other systems.
- Analogy: Operates like a processing machine that takes in input and delivers well-formatted output.

Example Flow

1. IVR sends a request:

   { "sessionId": "12345", "userInput": "press 1 for balance" }

2. acsRoutes.js directs the request to acsController.js.

3. acsController.js validates inputs and passes them to acsService.js.
4. acsService.js returns a mock ACS response:

```
{
  "platform": "ACS",
  "sessionId": "12345",
  "received": "press 1 for balance",
  "message": "ACS processed: press 1 for balance"
}
```

5.The final structured JSON response is delivered back to the IVR with success: true

Error Handling

To ensure clarity and maintainability, the `/acs/response` API returns well-defined error payloads for every major error scenario. Each error response uses a consistent JSON structure with clear status codes.

Standard Error Status Codes

- **400 Bad Request:** Invalid input provided.
- **401 Unauthorized:** Authentication failure.
- **404 Not Found:** Session or resources missing.
- **429 Too Many Requests:** Rate limit exceeded.
- **500 Internal Server Error:** Unexpected middleware or ACS failure.

JSON Error Structure

Each error response includes:

- `code`: An error code string.
- `message`: A human-readable description.
- `details`: Additional context or advice.

Sample Error JSONs:

```json
{
"errorCode": "INVALID_INPUT",
"message": "The digit provided is not supported.",
"docsLink": "https://api.docs/errors#INVALID_INPUT"
}
```