{API} insights.io BETA

2025-02-19 09:11:53

AssemblyAI API V1.3.4 API Results

AssemblyAI API

https://api.assemblyai.com

17 endpoints

Overall API Score

AI Ready
2 Issues
D
60

Design
5 Issues
D
66

Performance
3 Issues
F
52

Security
5 Issues
D
60

AI Ready

Fail **Response Descriptions**

Checks if each response has descriptions

Fail **Schema Types or Descriptions**

Checks if each schema has descriptions or types

Pass **Parameter Descriptions**

Checks if each parameter has descriptions

Pass **Operation ID**

Checks if each operation has an id

Design

Fail **JSON Support**

When designing APIs that send and receive JSON data, you should always declare the content-type header as application/json. This test checks to make sure the content-type header is application/json.

Fail **Unused Models**

Checks for the presence of data models defined but not used.

Fail **Rate Limiting**

Rate limiting is crucial for API security as it helps prevent abusive or malicious behavior by limiting the number of requests a client can make within a specified time. It mitigates the risk of unauthorized access, DDoS attacks, and resource exhaustion, ensuring fair and controlled usage of API resources while maintaining system stability and security. This test checks for the x-ratelimit-limit or x-rate-limit headers in the response payload to see if you use rate limiting.

Fail **Consistent Pluralization**

The base API endpoint provides the location of a group of resources (nouns). Therefore, your base endpoints should be described as plural nouns. For example: /users vs. /user

Fail **Examples Exist**

Checks for presence of value or externalValue for examples.

Pass **Operation Descriptions or Summaries**

All operations must contain either a description or summary

Pass **Multiple HTTP Methods**

A robust API should be designed methodically, allowing users to perform multiple functions. This test checks if your schema describes more than just GET response methods.

Pass **Consistent Noun Usage**

API functionality is defined by the HTTP verb (method) and convey actions or a state of a resource (noun). Therefore, your endpoints should be described as nouns instead of verbs. For example: POST method for endpoint /user/{id} vs. /postUser/{id}

Pass **Versioning**

Good API design provides a way manage changes that is transparent to API consumers and other API producers. We recommend using url-based versioning, either v or V followed by a number (/v2) or numeric based (/1).

Pass **Information Description**

Check to ensure all operations contain either a description or summary.

Pass **Contact Information**

Checks for presence of contact name, email, or URL.

Pass **Robust Responses**

A robust API will return response codes that help users understand what happened, even when the response is successful. This test checks to ensure your schema describes more than just (200) HTTP_OK responses.

Performance

Fail **Cache Support**

The cache-control header is used to specify browser caching policies, including how a resource is cached, where it is cached, and it's expiration.

Fail **Compression Support**

Compression speeds up the transmission of data and reduces packet loss. This test checks to see if compression content-encoding header exists.

Fail **CDN Usage**

Checks the base URL of the first entry in the servers section for responses of common CDN suppliers.

Skipped **Response size**

Checks if size of response is less than 100KB.

Pass **Load Time**

Load time is the time it takes for an API to process a request and return a response. While several factors affect load time, users expect a response in 500ms or less. This test checks for load times of API responses.

Pass **HTTP2 Usage**

Checks the base URL of the first entry in the servers section for version of the HTTP server.

Security

Fail **Content Security Policy**

The Content-Security-Policy header allows you to restrict which resources can be loaded and what URLs they can be loaded from. This test checks to see if the Content-Security-Policy header exists.

Fail **iFrame Embedding**

The X-Frame-Options header indicates whether a browser can render a page in a <frame> or <iframe>. If not set to DENY, sites can manipulate user's activity via clickjacking attacks. This test check if the X-Frame-Options header exists and its value is set to DENY.

Fail **Strict Transport Security**

The Strict-Transport-Security (HSTS) header is crucial for enforcing secure communication over HTTPS. When a server includes the HSTS header in its response, it instructs the browser to always connect to the API using HTTPS, even if the user enters an HTTP URL. This prevents potential downgrade attacks and ensures that all communication remains encrypted. This test checks for the Strict-Transport-Security header in your API responses.

Fail **Operation Enforces Security Scheme**

Checks to ensure operation security field is defined. If not anyone can access the API without any authentication.

Fail **Insecure Direct Object Reference(IDOR) Risks**

Checks for Insecure Direct Object Reference (IDOR) risks which can allow malicious users to access or modify objects by manipulating identifiers used in the API parameters. This test checks each path parameter's definition: they must be of type string and format UUID or type string with a valid UUID/ULID/Mongo ObjectID example present.

Pass **Authorization**

API Authorization is crucial for ensuring secure access and protecting sensitive data. This tests checks to ensure the securitySchemes object is present and is not empty.

Pass **Security Field Contains a Scheme**

Check to ensure there is no empty object in the security field.

Pass **Secure URLs (HTTPs)**

Good API design ensures that all data is transferred via an encrypted protocol. This test checks whether requests require HTTPS.

Pass **Global Security Field is Defined**

Checks if security field is defined

Pass **Content Type Options**

The X-Content-Type-Options header plays a vital role in protecting the API from MIME sniffing attacks. By setting the value of this header to nosniff, it instructs the browser to strictly adhere to the declared Content-Type and prevents it from attempting to sniff or interpret the response data based on its content. This test checks to see if the X-Content-Type-Options header exists and if its value is set to nosniff.