# Use the Citation Checker in EduGenAI

Register this tool as an EduGenAI Extension so the assistant can verify a paper's references against OpenAlex — inside your own EduGenAI chat, with no separate LLM key.

## How it works

An EduGenAI Extension is a function-calling tool: you give EduGenAI an HTTP endpoint and a function definition, and its assistant calls that endpoint as a JSON request whenever it needs to. The work splits across two sides:

- **EduGenAI's assistant** reads the paper and extracts the reference list plus the sentences that cite each source.
- **This extension endpoint** looks each reference up in OpenAlex and checks title, authors, year, journal, DOI, volume, issue and pages — deterministically, with no LLM — and returns the abstract.
- **EduGenAI's assistant** compares the citing sentence to that abstract and flags misquotes.

Because the reasoning stays on EduGenAI's side, the extension needs no LLM API key. It only wraps OpenAlex, which is free. An optional OpenAlex Premium key can be stored securely in the Headers / Azure Key Vault section (Step 3).

## Before you start

Deploy this app at a public HTTPS URL — EduGenAI calls it server-to-server. Everywhere below, replace `https://YOUR-DEPLOYMENT-HOST` with your deployment's address.

## Step 1 — Open the Extension builder

In EduGenAI, create a new Extension (Name, Short description, Headers, Functions).

## Step 2 — Name and describe it

**Name**

```
OpenAlex Citation Verifier
```

**Short description**

```
Verifies a paper's references against OpenAlex: checks that each
work exists and that the printed title, authors, year, journal,
DOI and pages match, and returns the abstract.
```

**Detail description (paste verbatim — this is the assistant's instruction)**

```
Use this extension to fact-check the reference list of an uploaded paper.
(Emoji are written as words below; copy the exact version with emoji from the
/edugenai page.)

STEP 1 - EXTRACT. Read the document. For every entry in the reference list, pull
out: title, the full list of author names in order, whether it ends in "et al.",
year, DOI (only if printed), journal/venue, volume, issue, and page range. Also
find the sentence(s) in the body where each source is cited, with one sentence of
context on each side. Keep those citation sentences in your own working notes for
Step 3 - do NOT send them to the function.

STEP 2 - VERIFY. Call the "verify_references" function once, passing every
reference. For each it returns: "badge" and "severity" (0-100), "status" (found /
fuzzy / not_found / lookup_failed), "mismatched_fields" and "minor_fields", a
"field_check" comparing each printed detail to OpenAlex (reference_value vs
openalex_value; each field's status is "match", "close" = a minor naming
variation such as an abbreviated journal name, or "mismatch"), the matched
"work" (authors, year, venue, doi, url) and its "abstract", and - for FUZZY
```

matches only, where "work" is null and "field_check" is empty - a "candidates" array of the closest works (each with title, authors, year, venue, doi, url).

STEP 3 - JUDGE MISQUOTES. For each reference whose status is "found", compare the citation sentence(s) you kept in Step 1 with the returned "abstract". If the student uses the source as though it were about a different topic than the abstract shows, that is a MISQUOTE; otherwise it is consistent. If there is no abstract, or the reference is never cited, the misquote check is "unclear". Never invent a verdict.

STEP 4 - PRESENT (follow this format exactly). The user must NEVER see the raw JSON from the function - it is a machine interface. Convert it into ONE readable Markdown table, one row per reference, sorted by DESCENDING severity. Use the "severity" value; if you judge a reference to be a misquote MISMATCH, raise its severity to AT LEAST 80 (keep the function's number if it is already higher). Columns:

  | # | Flag | Reference (as printed) | Status | Metadata issues | Misquote | What the paper is about |

- Flag: blue if status is lookup_failed (could not be checked); otherwise red if severity >= 70, amber if 45-69, green otherwise.
- Status: the "badge" value with a coloured dot - red Potential hallucination (not_found), amber Fuzzy match (fuzzy), blue Lookup failed (no result to compare), green Verified (found).
- Metadata issues: if "mismatched_fields" is empty, write "-". Otherwise, for each mismatched field take reference_value and openalex_value from "field_check" and write:  field: "printed" -> "OpenAlex".  Bold **authors** when it is one of them. (A fuzzy row has no field_check yet - write "-".) Fields with status "close" ("minor_fields") are NOT errors - leave them out of this column, or mention them only as "minor naming variant" without a flag.
- Misquote: Mismatch (red) / Likely mismatch (amber) / Consistent (green) / Unclear (dash), plus a 6-10 word reason. (Only "found" rows have a misquote.)
- What the paper is about: one short clause from the "work" abstract, or "-".

After the table, add a "Details" section for the red and amber rows only.
- FOUND row: the matched title as a link to "work.url", the field-by-field comparison, and one sentence on the misquote if any.
- FUZZY row: "work" is null - instead list "candidates" (each title as a link to its "url", with authors and year) and ask the user to confirm which, if any, is the intended source. Do NOT present a fuzzy row as verified.

End with one summary line:
"N references - X verified, Y fuzzy, Z potential hallucinations, plus flags."

Never invent a DOI, author, year, or verdict the function did not return. If "status" is "lookup_failed", say the reference could not be checked - do NOT call it a hallucination.

## Step 3 — Add the Header

In the Headers section add: **Key** `Content-Type` **Value** `application/json`.

Optional — OpenAlex Premium: add a second header `X-OpenAlex-Key` with your key as the value, using the Secure header values option so it is stored in Azure Key Vault.

## Step 4 — Add the function

**Method & URL**

```
POST  https://YOUR-DEPLOYMENT-HOST/api/verify_batch
```

**Function definition**

```
{
  "name": "verify_references",
  "description": "Verify a list of bibliographic references against OpenAlex.
    For each reference, returns whether the work exists (found / fuzzy /
    not_found), a field-by-field comparison of the printed metadata (title,
    authors, year, journal, DOI, volume, issue, pages) against the real record,
    and the abstract of the matched work.",
  "parameters": {
    "type": "object",
    "properties": {
      "references": {
        "type": "array",
        "description": "Every reference in the paper's bibliography.",
        "items": {
          "type": "object",
          "properties": {
            "title":   { "type": "string" },
            "authors": { "type": "array", "items": { "type": "string" },
                         "description": "Every author name printed, in order." },
            "et_al":   { "type": "boolean" },
            "year":    { "type": "integer" },
            "doi":     { "type": "string" },
            "journal": { "type": "string" },
            "volume":  { "type": "string" },
            "issue":   { "type": "string" },
            "pages":   { "type": "string" }
          },
          "required": ["title"]
        }
      }
    },
    "required": ["references"]
  }
}
```

## Step 5 — Submit and test

Click Submit. In a chat, upload a paper and ask: “Check the citations in this paper against OpenAlex.”

## What the endpoint returns

**Request EduGenAI sends to /api/verify_batch:**


```
{
  "references": [
    {
      "title": "Highly accurate protein structure prediction with AlphaFold",
      "authors": ["Smith, J.", "Jones, B."],
      "et_al": false,
      "year": 2019,
      "journal": "Science",
      "pages": "100-110"
    }
  ]
}
```


**Response (abbreviated) — wrong authors, year and journal are caught, and the abstract is returned for the misquote step:**


```
{
  "count": 1,
  "results": [{
    "index": 1,
    "status": "found",
    "badge": "Verified",
    "severity": 85,
    "priority": "Review",
    "mismatched_fields": ["authors", "year", "journal"],
    "minor_fields": [],
    "work": {
      "title": "Highly accurate protein structure prediction with AlphaFold",
      "authors": ["John Jumper", "Richard Evans", "..."],
      "year": 2021, "venue": "Nature",
      "doi": "10.1038/s41586-021-03819-2",
      "abstract": "Proteins are essential to life ... (used for the misquote check)"
    },
    "field_mismatch_count": 3,
    "field_check": [
      { "field": "title",   "status": "match" },
      { "field": "authors", "status": "mismatch",
        "reference_value": "Smith, J., Jones, B.",
        "openalex_value": "John Jumper, Richard Evans, ..." },
      { "field": "year",    "status": "mismatch",
        "reference_value": 2019, "openalex_value": 2021 },
      { "field": "journal", "status": "mismatch",
        "reference_value": "Science", "openalex_value": "Nature" }
    ]
  }]
}
```


A single-reference endpoint is also available at `POST /api/verify` (same fields, no outer “references” array). Prefer the batch endpoint for a whole bibliography — one round trip, kinder to OpenAlex rate limits.

## From JSON to a readable table — what you'll see in chat

You never look at that JSON — it is the machine interface between EduGenAI and the endpoint. The Detail description from Step 2 instructs the assistant to convert it (STEP 4) into a severity-sorted table with flags, scores, and the mismatches spelled out, mirroring how the website presents results. A typical chat answer looks like:

```
| # | Flag | Reference (as printed)      | Status              | Metadata issues                | Misquote
|---|------|-----------------------------|---------------------|--------------------------------|----------
| 3 | RED  | Zorblatt, Q. (2021). Quantum blockchain effects on medieval cheese trading... | Potential halluc
| 2 | RED  | Smith, J. & Jones, B. (2019). Highly accurate protein structure prediction... | Verified | autho
| 1 | GREEN | Vaswani, A. et al. (2017). Attention is all you need. NeurIPS. | Verified | - | Consistent | At
```

...followed by a Details section for the flagged rows and a one-line summary count. If you still see raw JSON in the chat, the Detail description probably wasn't saved with the extension (re-check Step 2) or was truncated — re-paste it, or as a per-chat override end your request with: “Present the results as a

severity-sorted table with flags and the metadata mismatches spelled out; do not show raw JSON." The endpoint's JSON ships ready-made display fields (badge, severity, priority, mismatched_fields) precisely so the assistant only has to lay them out, not compute them.

## What data passes through your server?

Because the extension calls your deployment, here is exactly what travels there — and what does not:

- **The paper's full text, the student's name, and the citing sentences stay inside EduGenAI.** The assistant keeps the citing sentences for its own misquote reasoning; they are never sent to your endpoint.
- Only reference **metadata** — title, authors, year, DOI, journal, volume, issue, pages — reaches your endpoint, and (titles/DOIs) go on to OpenAlex for the lookup.
- No LLM key is involved server-side; the reasoning runs on EduGenAI's model.

So: only bibliographic metadata goes through your server, and only in transit. The function call is the only thing that reaches it. This app is stateless — no database, no file writes — and never logs request bodies or keys. Whatever hosting stack or reverse proxy you deploy behind may keep its own access log (method + path + status, not bodies); that part is under your control.

Caveat about the OpenAlex Premium key: it is used per-request, never stored, and scrubbed from any error message this app returns. But when your endpoint calls OpenAlex the key rides as a query parameter on the outbound URL, so an egress proxy that logs full outbound URLs could capture it. If that matters, omit the Premium key (the tool works without one) or use a proxy that redacts query strings.

## Notes & limits

- No LLM key is stored or used by the extension; OpenAlex is free and needs no key.
- OpenAlex's canonical year can differ from a printed year (online-first vs issue year) — treat a lone year mismatch as a prompt to double-check, not a verdict.
- The misquote judgement uses the abstract only, so a claim supported by the full text but not the abstract may read as uncertain. Treat results as leads for a human reviewer.
- Batch requests are capped at 200 references.