TSLHostedAI

Self-hosted · Local AI · OpenAI-compatible

API Integration Guide

v1.0 · Generated: 7 November 2026

Overview

TSLHostedAI is a self-hosted AI gateway that routes requests to locally-running Ollama models. It provides an OpenAI-compatible REST API with API-key authentication, per-user token quotas, monthly usage tracking, an admin dashboard, and PWA support.

Base URL

```
https://tslhostedai.thehackitect.com.ng/api/v1
```

Authentication

Every request must include a valid Bearer token in the Authorization header. Generate API keys from your user dashboard. Each key can be revoked independently. Revoked keys can be deleted after revocation.

```
Authorization: Bearer <YOUR_API_KEY>
```

Available Endpoints

Method	Path	Auth	Description
GET	/api/v1/models	Yes	List all enabled models
POST	/api/v1/chat/completions	Yes	Chat completion (OpenAI-compatible)
GET	/api/openapi	No	OpenAPI 3.1 specification (JSON)
GET	/api/docs/pdf	No	Download this documentation PDF

HTTP Error Codes

400	Bad Request	malformed body or exceeded token limit
401	Unauthorized	missing or invalid API key
403	Forbidden	key revoked or insufficient role
429	Too Many Requests	monthly quota exceeded
500	Internal Server Error	model error or server fault
503	Service Unavailable	Ollama not running

GET /api/v1/models List Models

Returns the list of enabled AI models available on this platform.

Response (200 OK):

```
{
  "object": "list",
  "data": [
    { "id": "deepseek-r1:1.5b", "object": "model", "created": 1700000000 }
  ]
}
```

POST /api/v1/chat/completions Chat

OpenAI-compatible chat endpoint. Forwards to the selected Ollama model. Tracks usage and enforces quotas.

Request body fields:

Field	Type	Required	Description
`model`	string	No	Ollama model name. Falls back to platform default model.
`messages`	array	Yes	Array of {role, content}. Roles: system | user | assistant.
`stream`	boolean	No	Set true to receive a server-sent events (SSE) token stream.

cURL example:

```
curl -X POST https://tslhostedai.thehackitect.com.ng/api/v1/chat/completions \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "deepseek-r1:1.5b",
    "messages": [{"role": "user", "content": "Explain recursion briefly."}]
  }'
```

Python example (openai pkg as HTTP client points to YOUR server):

```
# TSLHostedAI is self-hosted. No data is ever sent to OpenAI.
# The 'openai' package is used only as an HTTP client.
from openai import OpenAI
client = OpenAI(base_url="https://tslhostedai.thehackitect.com.ng/api/v1", api_key="YOUR_API_KEY")
response = client.chat.completions.create(
    model="deepseek-r1:1.5b",
    messages=[{"role": "user", "content": "Explain recursion briefly."}],
)
print(response.choices[0].message.content)
```

JavaScript / Node.js using openai package as HTTP client

TSLHostedAI is fully self-hosted. No data is sent to OpenAI.
The openai npm package is used only as a typed HTTP client pointed at your own server.

```
// No connection to OpenAI  talks to YOUR self-hosted TSLHostedAI server
import OpenAI from "openai"; // npm install openai

const client = new OpenAI({
  baseURL: "https://tslhostedai.thehackitect.com.ng/api/v1",  // your server
  apiKey: "YOUR_API_KEY",      // your TSLHostedAI key
});

const res = await client.chat.completions.create({
  model: "deepseek-r1:1.5b",
  messages: [{ role: "user", content: "Explain recursion briefly." }],
});
console.log(res.choices[0].message.content);
```

Plain fetch (no library required)

You can also call the API with any HTTP client no SDK needed.

```
const response = await fetch("https://tslhostedai.thehackitect.com.ng/api/v1/chat/completions", {
  method: "POST",
  headers: {
    "Authorization": "Bearer YOUR_API_KEY",
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    model: "deepseek-r1:1.5b",
    messages: [{ role: "user", content: "Explain recursion briefly." }],
  }),
});
const data = await response.json();
console.log(data.choices[0].message.content);
```

Notes & Limitations

" Set stream: true to receive an SSE token stream (OpenAI-compatible). Default is false (blocking JSON response).

" API keys are shown only once at creation. Store them securely they cannot be recovered.

" Monthly token quotas reset at the start of each calendar month (UTC).

" Admins can set platform-wide input token limits. Unlimited-access users bypass these limits.

" Models must be enabled by an admin before they appear in /api/v1/models responses.

HTTP Status Codes

`200`	Success JSON response (non-stream) or SSE stream.
`400`	Bad request missing or invalid fields in the request body.
`401`	Unauthorized missing or invalid Bearer API key.
`404`	Model not found the requested model is not enabled by an admin.
`429`	Rate limited monthly token quota exceeded for this user.
`500 / 502`	Internal error Ollama runtime error or unexpected server failure.
`503`	Service unavailable Ollama is not running or unreachable.