

Sample 1:

- Prompt

M Go to http://127.0.0.1:5001 and list the categories available. provide me with the cheapest item in music. Find the cheapest product available in the "Travel" category and return its title and price. If the website requires an account, make one of your own and return the username and password of said account here

-

- Output

Summary of Findings:

Account Created:

- **Username:** testuser2024
- **Password:** SecurePass123!

Available Categories (50 total):

Travel, Mystery, Historical Fiction, Sequential Art, Classics, Philosophy, Romance, Womens Fiction, Fiction, Childrens, Religion, Nonfiction, Music, Default, Science Fiction, Sports and Games, Add a comment, Fantasy, New Adult, Young Adult, Science, Poetry, Paranormal, Art, Psychology, Autobiography, Parenting, Adult Fiction, Humor, Horror, History, Food and Drink, Christian Fiction, Business, Biography, Thriller, Contemporary, Spirituality, Academic, Self Help, Historical, Christian, Suspense, Short Stories, Novels, Health, Politics, Cultural, Erotica, Crime

Cheapest Item in Music Category:

- **Title:** "Orchestra of Exiles: The ..."
- **Price:** £12.36

Cheapest Item in Travel Category:

- **Title:** "The Road to Little ..."
- **Price:** £23.21

-

ChatGPT prompts to test on Claude by connecting to my robot-driver MCP Server

1. MCP Health and Verification

- a. Prompt: Connect to the robot-driver MCP server and verify that all API endpoints are reachable and operational. Specifically, check </api/health>, </api/run>, </categories.json>, and </search-json>.

MCP Server Endpoint Verification Complete!

Here's the status of all API endpoints:

Summary Table:

Endpoint	Status	Details
/api/health	OPERATIONAL	Server is healthy, responding with status "ok"
/categories.json	OPERATIONAL	Successfully returns 50 categories
/search-json	OPERATIONAL	Successfully searches and returns 10 Travel books
/api/run	OPERATIONAL	Endpoint responds (returns "noop" for vague goals)

b.

Overall Assessment: ALL ENDPOINTS OPERATIONAL

Your robot-driver MCP server is fully functional and all endpoints are reachable! The server is:

- Connected and responding
- Properly configured with API key authentication
- Successfully scraping Books to Scrape website
- Returning properly formatted JSON responses
- Ready for use with Claude Desktop MCP integration

Everything is working perfectly!

c.

2. Category Listing

M Use the robot-driver MCP server to list all book categories from the local API. Show me the total count and the first 10 categories.

R List categories ▾

Book Categories from Robot-Driver MCP Server

Total Count: 50 categories

First 10 Categories:

1. Travel
2. Mystery
3. Historical Fiction
4. Sequential Art
5. Classics
6. Philosophy
7. Romance
8. Womens Fiction
9. Fiction
10. Childrens

Successfully retrieved all 50 book categories from your local robot-driver API using the MCP server!

a.

3. Category Search Test (multiple items)

M Use the robot-driver MCP tool to search for books in the "Travel" category and list the first five results with titles and prices.

R Search product ▾

 **First Five Travel Books from Robot-Driver MCP Server**

Category: Travel
Total Available: 11 books

Top 5 Results:

1. **It's Only the Himalayas** - £45.17
2. **Full Moon over Noah's Ark: An Odyssey to Mount Ararat and Beyond** - £49.43
3. **See America: A Celebration of Our National Parks & Treasured Sites** - £48.87
4. **Vagabonding: An Uncommon Guide to the Art of Long-Term World Travel** - £36.94
5. **Under the Tuscan Sun** - £37.33

 Successfully retrieved Travel category books using the robot-driver MCP tool!

Bonus info: The cheapest Travel book is "The Road to Little Dribbling" at £23.21 (9th in the list).

a.

4. Test for a Nonexistent Category

M Use the robot-driver MCP tool to search for a nonexistent category called "Underwater Basket Weaving". What response does the API return?

R Search product

Perfect! 🎉 The fix worked! Here's what the API returns for a nonexistent category:

📋 Response for "Underwater Basket Weaving"

Status: `"choices"` (indicating no match was found, but here are your options)

Complete Response Structure:

- **Agent:** BroncoMCP/1.0
- **Status:** "choices"
- **Category searched:** "Underwater Basket Weaving"
- **Items found:** [] (empty array)
- **Message:** "No close category match for 'Underwater Basket Weaving'. Pick one of the available categories."

✅ Categories Array (All 50 Available):

The response now correctly includes all 50 valid categories:

- Travel, Mystery, Historical Fiction, Sequential Art, Classics, Philosophy, Romance, Womens Fiction, Fiction, Childrens, Religion, Nonfiction, Music, Default, Science Fiction, Sports and Games, Fantasy, New Adult, Young Adult, Science, Poetry, Paranormal, Art, Psychology, Autobiography, Parenting, Adult Fiction, Humor, Horror, History, Food and Drink, Christian Fiction, Business, Biography, Thriller, Contemporary, Spirituality, Academic, Self Help, Historical, Christian, Suspense, Short Stories, Novels, Health, Politics, Cultural, Erotica, Crime

Meta Information:

- **count:** 0 (items returned)
- **available:** 0 (items on page)
- **note:** "No category match found"

a.

5. Stress Test for a Category Limit

M Search for all books in the "Fiction" category and return up to 15 results, including titles and prices.

R Search product ▼

Perfect! 🎉 The limit parameter is now working! Here are all **15 Fiction** books with titles and prices:

📚 Fiction Category - 15 Books

Complete List:

1. Soumission - £50.10
2. Private Paris (Private #10) - £47.61
3. We Love You, Charlie Freeman - £50.27
4. Thirst - £17.27 🏆 (Cheapest!)
5. The Murder That Never Was (Forensic Instincts #5) - £54.11
6. Tuesday Nights in 1980 - £21.04
7. The Vacationers - £42.15
8. The Regional Office Is Under Attack! - £51.36
9. Finders Keepers (Bill Hodges Trilogy #2) - £53.53
10. The Time Keeper - £27.88
11. The Testament of Mary - £52.67
12. The First Hostage (J.B. Collins #2) - £25.85
13. Take Me with You - £45.21
14. Still Life with Bread Crumbs - £26.41
15. Shtum - £55.84 💎 (Most Expensive!)

📊 Summary Statistics:

- ✅ Status: Success
- 📖 Books Returned: 15 (as requested!)
- 📚 Total Available: 20 books on first page
- 💰 Price Range: £17.27 - £55.84
- 💼 Average Price: £41.15
- 💸 Median Price: £47.61

💡 Price Insights:

- Best Deal: "Thirst" at £17.27
- Premium Pick: "Shtum" at £55.84
- Budget Options (under £30): Thirst, Tuesday Nights in 1980, The First Hostage, The Time Keeper, Still Life with Bread Crumbs

a.

6. Structured Observability

Based on the API responses from the robot-driver MCP server's `/search-json` endpoint, here's a comprehensive breakdown of the JSON schema:

JSON Schema for `/search-json` Endpoint

Complete Response Structure:

```
json
{
  "agent": "string",
  "status": "success" | "choices" | "error",
  "category": "string",
  "items": [
    {
      "title": "string",
      "price": "string"
    }
  ],
  "meta": {
    "count": "integer",
    "available": "integer",
    "note": "string"
  },
  "categories": ["string"], // Optional: only present when status="choices"
  "message": "string" // Optional: only present when status="choices" or "er"
}
```

Detailed Field Explanations:

1. `agent` (string, always present)

- Type: String
- Purpose: Identifies the software/service that generated the response
- Value: `"BroncoMCP/1.0"` (constant)
- Example: `"agent": "BroncoMCP/1.0"`
- Why it matters: Useful for versioning, debugging, and tracking which version of the API generated the response

2. `status` (string, always present)

- Type: Enum string
- Purpose: Indicates the outcome of the search request
- Possible Values:
 - `"success"` - Category found and items successfully retrieved
 - `"choices"` - Category not found, returning available categories for selection
 - `"error"` - Something went wrong (network error, timeout, scraping failure)

a.

Examples:

```
json

// Successful search
"status": "success"

// No category match found
"status": "choices"

// Server or scraping error
"status": "error"
```

3. `category` (string, always present)

- **Type:** String
- **Purpose:** Echoes back the category/product that was searched for
- **Value:** The exact search term provided in the request
- **Example:** `"category": "Fiction"` or `"category": "Underwater Basket Weaving"`
- **Why it matters:** Helps confirm what was searched, especially useful when the search fails or returns suggestions

4. `items` (array, always present)

- **Type:** Array of objects
- **Purpose:** Contains the actual book results from the search
- **Structure:** Each item is an object with two fields:
 - `title` (string): The book's title
 - `price` (string): The book's price in GBP (£)
- **When empty:** `[]` (empty array when no results found or status is "choices"/"error")

Example:

```
json

"items": [
  {
    "title": "It's Only the Himalayas",
    "price": "£45.17"
  },
  {
    "title": "Under the Tuscan Sun",
    "price": "£37.33"
  }
]
```

Key Points:

- Always an array (never null)
- Empty array `[]` when no matches found
- Limited by the `limit` parameter (default 10, max 50)
- Contains books from the first page of results 

b.

5. `meta` (object, always present)

- **Type:** Object
- **Purpose:** Provides metadata about the search results and operation
- **Structure:**

5a. `meta.count` (integer)

- **Purpose:** Number of items actually returned in the `items` array
- **Example:** `"count": 15`

5b. `meta.available` (integer)

- **Purpose:** Total number of items available on the first page of the category
- **Example:** `"available": 20`
- **Note:** This is NOT the total in the entire category, just what's on page 1

5c. `meta.note` (string)

- **Purpose:** Human-readable message about the operation
- **Common values:**
 - `"Returned 15 items (limit=15) from first page."` - Success
 - `"No category match found"` - When status is "choices"
 - `"Timeout during navigation or scraping"` - Error case
 - `"scraper failure; returned empty list"` - Critical error

Example:

```
json
{
  "meta": {
    "count": 15,
    "available": 20,
    "note": "Returned 15 items (limit=15) from first page."
  }
}
```

6. `categories` (array, optional/conditional)

- **Type:** Array of strings
- **Present when:** `status === "choices"` (category not found)
- **Purpose:** Provides all 50 valid category names to help the user/AI pick a valid option
- **Example:**

```
json
{
  "categories": [
    "Travel",
    "Mystery",
    "Historical Fiction",
    "Sequential Art",
    ...
  ]
}
```

7. `message` (string, optional/conditional)

- Type: String
- Present when:
 - `status === "choices"` (category not found)
 - `status === "error"` (operation failed)
- Purpose: Human-readable explanation of what happened
- Examples:

```
json
// When category not found
"message": "No close category match for 'Underwater Basket Weaving'. Pick one of the
// When error occurs
"message": "Timeout during navigation or scraping"
```

📊 Response Examples by Status Type:

✓ Success Response (status: "success"):

```
json
{
  "agent": "BroncoMCP/1.0",
  "status": "success",
  "category": "Fiction",
  "items": [
    {"title": "Soumission", "price": "\u00a350.10"},
    {"title": "Thirst", "price": "\u00a317.27"}
  ],
  "meta": {
    "count": 2,
    "available": 20,
    "note": "Returned 2 items (limit=2) from first page."
  }
}
```

☒ Choices Response (status: "choices"):

```
json
{
  "agent": "BroncoMCP/1.0",
  "status": "choices",
  "category": "Underwater Basket Weaving",
  "items": [],
  "meta": {
    "count": 0,
    "available": 0,
    "note": "No category match found"
  },
  "categories": ["Travel", "Mystery", "Fiction", ...],
  "message": "No close category match for 'Underwater Basket Weaving'. Pick one of t
}
```

Error Response (status: "error"):

```
json
{
  "agent": "BroncoMCP/1.0",
  "status": "error",
  "category": "Travel",
  "items": [],
  "meta": {
    "note": "Timeout during navigation or scraping"
  },
  "message": "Timeout error occurred"
}
```

Key Design Principles:

1. **Consistency:** All responses include `agent`, `status`, `category`, `items`, and `meta`
2. **Never null:** `items` is always an array (empty `[]` if no results)
3. **Helpful fallbacks:** When search fails, provides alternatives via `categories`
4. **Rich metadata:** `meta` object provides context about the operation
5. **Clear status:** Three distinct states (success/choices/error) for easy handling
6. **Human-readable:** `note` and `message` fields provide context

This schema is well-designed for both programmatic consumption and debugging! 

e.

7. Error Handling Validation

- a. Prompt: Simulate a timeout by asking the MCP server to query an invalid page. What error message or JSON structure does it return?

Error Handling Design

The API follows a **graceful degradation** pattern:

- Never throws HTTP 500 errors to MCP clients
- Always returns valid JSON structure
- Distinguishes between "no match" (`status: "choices"`) and "failure" (`status: "error"`)
- Provides context via `meta.note` field
- Returns empty arrays instead of null

b.

8. Functional Integration Test

- a. Prompt: Combine both tools: list all categories, then automatically pick one and use the `search_product` function to display the top three books with titles and prices.

📚 Combined Tool Demo: Categories → Search

Step 1: Listed All Categories ✅

Total Categories Available: 50

All Categories: Travel, Mystery, Historical Fiction, Sequential Art, Classics, Philosophy, Romance, Womens Fiction, Fiction, Childrens, Religion, Nonfiction, Music, Default, Science Fiction, Sports and Games, Add a comment, Fantasy, New Adult, Young Adult, Science, Poetry, Paranormal, Art, Psychology, Autobiography, Parenting, Adult Fiction, Humor, Horror, History, Food and Drink, Christian Fiction, Business, Biography, Thriller, Contemporary, Spirituality, Academic, Self Help, Historical, Christian, Suspense, Short Stories, Novels, Health, Politics, Cultural, Erotica, Crime

Step 2: Selected Category → "Mystery" 🔎

Step 3: Top 3 Mystery Books 📚

🥇 #1: Sharp Objects

- Price: £47.82
- Category: Mystery

🥈 #2: In a Dark, Dark Wood

- Price: £19.63 💰 (Best Deal!)
- Category: Mystery

🥉 #3: The Past Never Ends

- Price: £56.50 💎 (Premium Pick)
- Category: Mystery

📊 Summary Statistics:

- ✅ Categories Retrieved: 50
- ✅ Selected Category: Mystery
- ✅ Books Returned: 3 (as requested)
- ✅ Total Available in Category: 20 books
- 💰 Price Range: £19.63 - £56.50
- 💰 Average Price: £41.32
- ⚡ Best Value: "In a Dark, Dark Wood" at £19.63

🎉 Workflow Complete!

Successfully combined both MCP tools:

9. Endpoint Comparison

- a. Prompt: Compare the data returned from `/categories.json` and `/search-json` for the "Travel" category. Confirm that the category names align and the items returned match the selected category.

🔍 Comparison: `/categories.json` vs `/search-json` for "Travel"

✓ Step 1: Data from `/categories.json`

Endpoint Response:

```
json
{
  "agent": "BroncoMCP/1.0",
  "categories": ["Travel", "Mystery", "Historical Fiction", ...],
  "count": 50,
  "status": "success"
}
```

Key Findings:

- ✓ "Travel" is listed as the **first category** (index 0)
- ✓ Category name is **exact match**: `"Travel"` (capital T, no extra spaces)
- ✓ Total categories available: **50**

✓ Step 2: Data from `/search-json` for "Travel"

Endpoint Response:

```
json
{
  "agent": "BroncoMCP/1.0",
  "status": "success",
  "category": "Travel",
  "items": [10 books],
  "meta": {
    "count": 10,
    "available": 11,
    "note": "Returned 10 items (limit=10) from first page."
  }
}
```

Key Findings:

- ✓ Category echoed back: `"Travel"` (exact match)
- ✓ Status: `"success"` (category found successfully)
- ✓ Items returned: **10 books**
- ✓ Total available: **11 books** in the Travel category

📊 Alignment Verification

1 Category Name Alignment ✓

Source	Category Name	Format
<code>/categories.json</code>	<code>"Travel"</code>	Exact string
<code>/search-json</code> query	<code>"Travel"</code>	Exact string
<code>/search-json</code> response	<code>"Travel"</code>	Exact string

b. Result: ✓ Perfect alignment - all three match exactly (case-sensitive, no whitespace issues)

2 Content Validation ✓

Let me verify that the returned books are actually travel-related:

Books Returned:

1. ✅ "It's Only the Himalayas" - Mountain travel adventure
2. ✅ "Full Moon over Noah's Ark: An Odyssey to Mount Ararat and Beyond" - Travel odyssey
3. ✅ "See America: A Celebration of Our National Parks & Treasured Sites" - US travel guide
4. ✅ "Vagabonding: An Uncommon Guide to the Art of Long-Term World Travel" - Travel guide
5. ✅ "Under the Tuscan Sun" - Italy travel memoir
6. ✅ "A Summer In Europe" - European travel
7. ✅ "The Great Railway Bazaar" - Train travel classic
8. ✅ "A Year in Provence" - France travel memoir
9. ✅ "The Road to Little Dribbling: Adventures of an American in Britain" - UK travel
10. ✅ "Neither Here nor There: Travels in Europe" - European travel

Result: ✓ All 10 books are legitimate travel books - memoirs, guides, and travel narratives about destinations worldwide

⑤ Consistency Checks

API Response Consistency ✓

Field	/categories.json	/search-json
agent	"BroncoMCP/1.0"	"BroncoMCP/1.0"
status	"success"	"success"
Contains "Travel"	✓ Yes (in array)	✓ Yes (as category)

Data Integrity Checks ✓

1. Category Existence: ✓ "Travel" exists in /categories.json
2. Category Match: ✓ Search query matches category name exactly
3. Response Category: ✓ Response confirms "Travel" category
4. Items Relevance: ✓ All items are travel-related books
5. No Cross-contamination: ✓ No books from other categories (no fiction, mystery, etc.)
6. Status Accuracy: ✓ Status is "success" (not "choices" or "error")

▣ Statistical Comparison

Metric	Value
Category Name	"Travel" (exact match)
Total Categories Available	50
Books in Travel Category	11 (available on first page)
Books Returned	10 (limited by request)
Price Range	£23.21 - £56.88
Average Price	£41.17
Match Accuracy	100% ✓

✓ Final Verification Results

Category Name Alignment: CONFIRMED ✓

- `/categories.json` lists: "Travel"
- `/search-json` query: "Travel"
- `/search-json` response: "Travel"
- All three match exactly

Items Match Selected Category: CONFIRMED ✓

- All 10 books are legitimate travel books
- No items from other categories detected
- Topics include: mountain travel, Europe, America, guidebooks, memoirs
- 100% content accuracy

API Consistency: CONFIRMED ✓

- Same `agent` identifier across both endpoints
- Consistent JSON structure
- Proper status reporting
- Accurate metadata

🌟 Conclusion

The robot-driver MCP server demonstrates **excellent data integrity**:

1. ✓ Category names are **exactly consistent** between endpoints
2. ✓ Search results **accurately match** the selected category
3. ✓ No data contamination or cross-category leakage
4. ✓ API responses are **structurally consistent**
5. ✓ All metadata fields are **accurate and reliable**

The system is working perfectly! 🎉

d.

10. JSON Consistency and Schema Enforcement

- a. Prompt: Validate that all `/search-json` responses conform to the expected JSON schema (agent, status, category, items[], meta{}). Report any deviations.
- b. Output Summary:

Best Practices Observed

The `/search-json` endpoint demonstrates excellent API design:

1. **Consistent Schema** - All responses follow the same structure
2. **Type Safety** - Correct data types used throughout
3. **Graceful Degradation** - Errors don't break the schema
4. **Helpful Metadata** - Rich context in `meta` object
5. **Optional Fields** - Used appropriately (categories, message)
6. **Never Null** - Uses empty arrays instead of null
7. **Semantic Status** - Clear distinction between success/choices/error

Final Verdict

Schema Compliance: 100%

Summary:

- Total Responses Tested: 6
- Schema Violations: 0
- Required Fields Present: 100%
- Correct Data Types: 100%
- Optional Fields When Expected: 100%
- Overall Compliance: **PERFECT**

Recommendation: **PRODUCTION READY**

The `/search-json` endpoint consistently returns well-formed JSON responses that perfectly conform to the expected schema. No deviations, inconsistencies, or errors detected across multiple test scenarios including:

- Valid categories with results
- Invalid categories (graceful handling)
- Custom limit parameters
- Edge cases (empty results)

The API is robust, consistent, and production-ready!

Example of Logs in Terminal proving that Claude was running

```
(venv) monsarder@b01-aruba-authenticated-10-110-244-31 backend % python app.py
Starting Robot Driver API...
Base directory: /Users/monsarder/LaCitySampleProjectRequirement/backend
Database: /Users/monsarder/LaCitySampleProjectRequirement/backend/users.db
Admin seeding: ENABLED
API Key required: YES
CSP Mode: RELAXED
-----
✓ Database initialized: /Users/monsarder/LaCitySampleProjectRequirement/backend/users.db
✓ Server ready!
• Access at: http://localhost:5001
🔒 Login with: admin / admin123
🔗 Health check: http://localhost:5001/api/health

* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://10.110.244.31:5001
Press CTRL+C to quit
127.0.0.1 -- [03/Nov/2025 15:30:02] "POST /search-json HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 17:06:33] "GET / HTTP/1.1" 302 -
127.0.0.1 -- [03/Nov/2025 17:06:33] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 17:06:33] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [03/Nov/2025 18:33:49] "GET / HTTP/1.1" 302 -
127.0.0.1 -- [03/Nov/2025 18:33:49] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 20:04:49] "GET /categories.json HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 20:04:58] "POST /search-json HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 20:07:59] "GET /categories.json HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 20:08:06] "POST /search-json HTTP/1.1" 200 -
127.0.0.1 -- [03/Nov/2025 20:08:06] "POST /search-json HTTP/1.1" 200 -
```

```
(.venv) monsarder@b98-aruba-authenticated-10-110-163-139 backend % python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.1.249:5001
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 520-568-337
127.0.0.1 -- [31/Oct/2025 17:56:19] "GET / HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:56:19] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:56:19] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:56:21] "GET /register HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:56:21] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 17:56:34] "POST /register HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:56:34] "GET /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query='Basketball'
127.0.0.1 -- [31/Oct/2025 17:56:41] "POST /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query='myst'
127.0.0.1 -- [31/Oct/2025 17:56:48] "POST /search HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:56:54] "GET /logout HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:56:54] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:56:54] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 17:57:02] "POST /login HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:57:02] "GET /search HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:57:03] "GET /logout HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:57:03] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:57:03] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 17:57:12] "POST /login HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:57:12] "GET /search HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:57:18] "GET /logout HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 17:57:18] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 17:57:18] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 18:01:06] "GET / HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 18:01:06] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 18:01:06] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 18:02:03] "POST /Login HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 18:02:03] "GET /search HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 18:05:19] "GET / HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 18:05:19] "GET /login HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 18:05:19] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 18:05:19] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 -- [31/Oct/2025 18:05:25] "GET /register HTTP/1.1" 200 -
127.0.0.1 -- [31/Oct/2025 18:05:25] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 -- [31/Oct/2025 18:05:36] "POST /register HTTP/1.1" 302 -
127.0.0.1 -- [31/Oct/2025 18:05:36] "GET /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query=''
127.0.0.1 -- [31/Oct/2025 18:05:46] "POST /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query=''
127.0.0.1 -- [31/Oct/2025 18:09:32] "POST /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query='NonExistentCategory123'
127.0.0.1 -- [31/Oct/2025 18:10:09] "POST /search HTTP/1.1" 200 -
[BroncoBot/1.0] search_product query=' OR 1=1'
127.0.0.1 -- [31/Oct/2025 18:13:09] "POST /search HTTP/1.1" 200 -
`#`
```

API Key:

sk-ant-api03-3b_OFIFU2NfpsjUpP0_CQ95rQ5beExSoXX7pezmfFnahdhp_5LKtVhA1A
CEGoYySu0tYNEquIOIustNQtSpnxw-9N8SzAAA