

Case Study

Automating Event Intelligence & Scheduling with n8n



n8n



01 Project Overview



Project Overview

This project created a fully automated system using **n8n**, **ChatGPT**, and **Google Calendar** to scrape, enrich, color-code, and sync event data across platforms, transforming the client's manual workflows into a seamless, zero-research event planning process.

02 Problems

Problems

01

Continuously gather and organize event data from multiple sites.

02

Avoid manual work and searching for relevant interviews or summaries.

03

Sync all enriched data into both Google & Notion Calendar.

04

Maintain a fast, reliable, and scalable setup for their business

03 Solutions

Web Scraping via API

01

Each event was scraped using precise CSS selectors for each of the four websites. This approach eliminated the need for manual copying of event data and centralized event metadata from multiple platforms into one structured format.

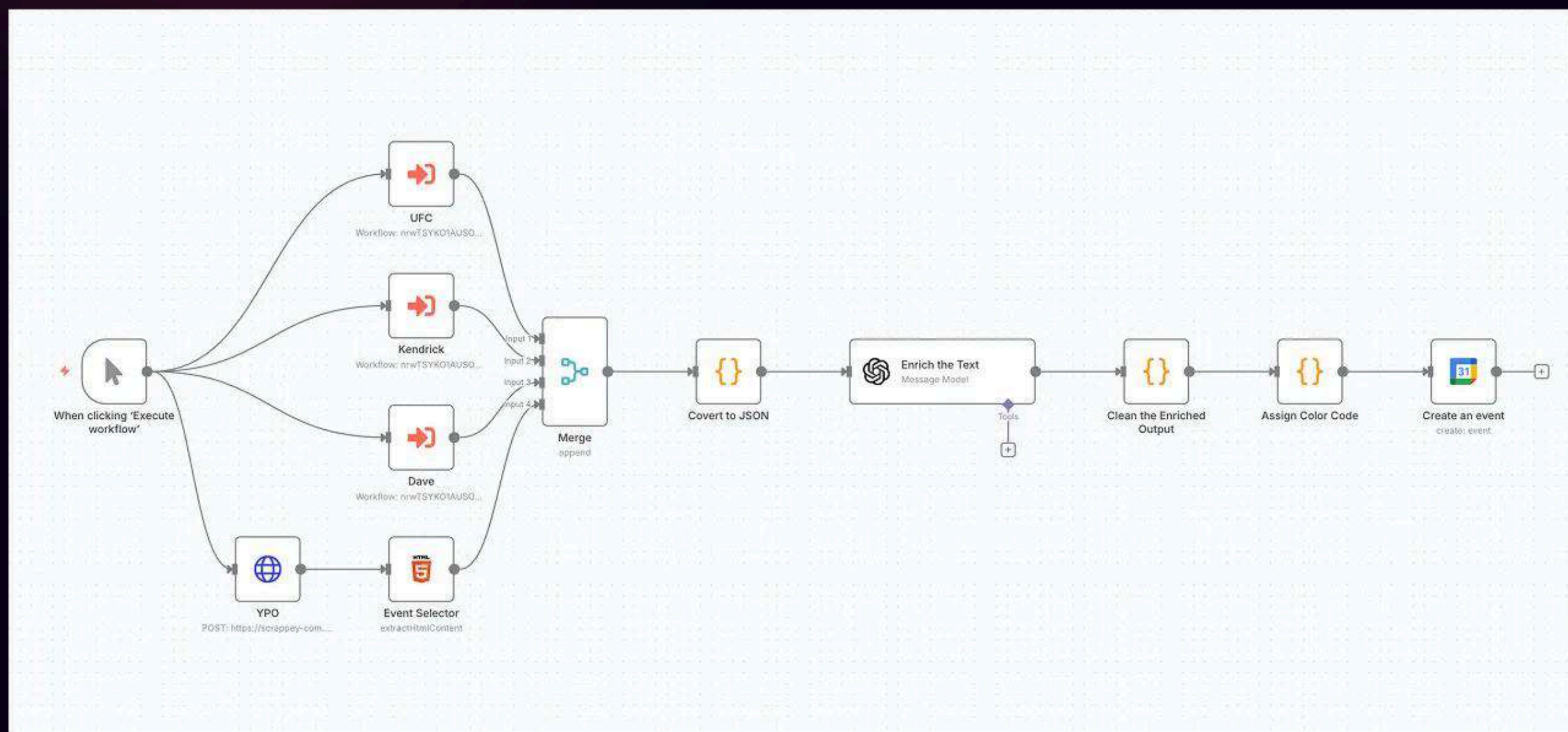
The screenshot displays a workflow automation interface for a step named 'UFC'. The 'INPUT' section shows a trigger 'When clicking "Execute workflow"'. The 'UFC' step configuration includes the following parameters:

- Source: Database
- Workflow: From list (Get Events Array)
- Workflow Inputs:
 - titleSelector: .c-card-event--result__headline a
 - dateSelector: .c-card-event--result__date
 - locationSelector: .c-card-event--result__location .field--name-location .address
 - venueSelector: (empty)
 - url: https://www.ufc.com/events?fa=tuf.home
- Attempt To Convert Types: (toggle off)

The 'OUTPUT' section shows a table with 15 items, 1 sub-execution. The table has columns for 'title' and 'date'.

title	date
0 : Taira vs Park	0 : Sat, Aug 2 / 9:00 PM EDT
1 : Dolidze vs Hernandez	Card [/event/ufc-fight-night-02-2025]
2 : Du Plessis vs Chimaev	
3 : Walker vs Zhang	1 : Sat, Aug 9 / 7:00 PM EDT
4 : Imavov vs Borralho	Card [/event/ufc-fight-night-09-2025]
5 : Lopes vs Silva	
6 : Ulberg vs Reyes	2 : Sat, Aug 16 / 10:00 PM EDT
7 : Ankalaev vs Pereira 2	Card [/event/ufc-319]
8 : Whittaker vs de Ridder	
9 : Holloway vs Poirier 3	3 : Sat, Aug 23 / 7:00 AM EDT
10 : Lewis vs Teixeira	Card [/event/ufc-fight-night-september-06-2025]
11 : Topuria vs Oliveira	
12 : Hill vs Rountree Jr.	5 : Sat, Sep 13 / 6:00 PM EDT
13 : Usman vs Buckley	Card [/event/ufc-fight-night-september-13-2025]
14 : Dvalishvili vs O'Malley 2	
15 : TBD vs TBD	6 : Sat, Sep 27 / 10:00 PM EDT
	Card [/event/ufc-fight-night-september-27-2025]
	7 : Sat, Oct 4 / 10:00 PM EDT
	Card [/event/ufc-320]
	8 : Sat, Jul 26 / 3:00 PM EDT
	Card [/event/ufc-fight-night-2025]
	9 : Sat, Jul 19 / 10:00 PM EDT
	Card [/event/ufc-318]
	10 : Sat, Jul 12 / 9:00 PM EDT
	Card [/event/ufc-fight-night-2025]
	11 : Sat, Jun 28 / 10:00 PM EDT
	Card [/event/ufc-317]

Workflow Orchestration



02

n8n runs a unified workflow that merges outputs from all sources, processes them in parallel, and feeds them into a single pipeline. This enabled centralized processing of otherwise siloed data sources and made the solution scalable for the future.

JSON & Data Structuring

The screenshot shows a data transformation tool interface. On the left, a 'Merge' node outputs a table with columns 'title' and 'date'. The 'title' column contains 16 entries, including 'Taira vs Park', 'Dolidze vs Hernandez', 'Du Plessis vs Chimaev', 'Walker vs Zhang', 'Imavov vs Borralho', 'Lopes vs Silva', 'Ulberg vs Reyes', 'Ankalaev vs Pereira 2', 'Whittaker vs de Ridder', 'Holloway vs Poirier 3', 'Lewis vs Teixeira', 'Topuria vs Oliveira', 'Hill vs Rountree Jr.', 'Usman vs Buckley', 'Dvalishvili vs O'Malley 2', and 'TBD vs TBD'. The 'date' column is empty. The main area shows a 'Convert to JSON' node with the following JavaScript code:

```
1 // Run Once for All Items
2 const allEvents = [];
3
4 // Loop over each merged-from-scraper source item
5 items.forEach(item => {
6   // pull out each array, defaulting to []
7   const titles = item.json.title || [];
8   const dates = item.json.date || [];
9   const locations = item.json.location || [];
10  const venues = item.json.venue || [];
11
12  // build one event per title
13  for (let i = 0; i < titles.length; i++) {
14    allEvents.push({
15      title: (titles[i] || '').trim(),
16      date: (dates[i] || '').trim(),
17      location: (locations[i] || '').trim(),
18      venue: (venues[i] || '').trim()
19    });
20  }
21 }
```

The output on the right shows a single item with a 'batch' array containing 4 objects:

```
batch
0
  title : Taira vs Park
  date  : Sat, Aug 2 / 9:00 PM EDT / Main Card
        [/event/ufc-fight-night-august-02-2025]
  location : Las Vegas, NV\nUnited States
  venue  : [empty]
1
  title : Dolidze vs Hernandez
  date  : Sat, Aug 9 / 7:00 PM EDT / Main Card
        [/event/ufc-fight-night-august-09-2025]
  location : Las Vegas, NV\nUnited States
  venue  : [empty]
2
  title : Du Plessis vs Chimaev
  date  : Sat, Aug 16 / 10:00 PM EDT / Main Card
        [/event/ufc-319]
  location : Chicago, IL\nUnited States
  venue  : [empty]
3
  title : Walker vs Zhang
  date  : Sat, Aug 23 / 7:00 AM EDT / Main Card
```

03

A JavaScript block parses all scraped text data into a clean JSON structure for downstream processing. This creates machine-readable output ready for AI enrichment and ensures consistent formatting regardless of the data source.

AI Enrichment via ChatGPT

04

A GPT-4o model is used to summarize each event, extract agency, country, venue, and interview details, and add contextual background and relevance. This removed the need for manual research and improved content quality with rich, AI-generated context.

The screenshot displays a workflow editor with three main panels: INPUT, ENRICH THE TEXT, and OUTPUT.

- INPUT:** A JSON array of event objects. Each object contains fields like "title", "date", "location", and "venue".
- ENRICH THE TEXT:** A configuration panel for the "Enrich the Text" step. It includes:
 - Parameters: "Credential to connect with" (OpenAi account), "Resource" (Text), "Operation" (Message a Model), "Model" (GPT-4O).
 - Messages: A prompt instructing the AI to act as a researcher and return a JSON object with keys: "summary", "background", "pr_agencies", "trending", and "agents".
 - Role: "User".
 - Buttons: "Add Message", "Execute step", "Simplify Output", "Tools".
- OUTPUT:** The resulting JSON array, where each event object now includes the AI-generated fields: "summary", "background", "pr_agencies", "trending", and "agents".

Cleaning & Parsing Model Output

The screenshot displays a workflow tool interface with three main sections:

- Left Panel:** A dropdown menu labeled "Enrich the Text" with "1 item" below it. Below the menu is a JSON object representing an enriched text item. The "content" field contains a string with escaped backslashes and newlines, representing a JSON array of event objects.
- Middle Panel:** A configuration area with tabs for "Parameters" and "Settings". The "Mode" is set to "Run Once for All Items" and the "Language" is set to "JavaScript". A code editor shows the following JavaScript code:

```
6 const clean = raw
7   .replace(/```json/g, '')
8   .replace(/```/g, '')
9   .trim();
10
11 // 3) Parse the JSON array
12 let enrichedArray;
13 try {
14   enrichedArray = JSON.parse(clean);
15 } catch (err) {
16   throw new Error(`Failed to JSON.parse enriched content:\n${clean}`);
17 }
18
19 // 4) Return one n8n item per enriched event
20 return enrichedArray.map(evt => ({
21   json: evt
22 }));
```

A tooltip at the bottom of the code editor reads: "Type \$ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console."
- Right Panel:** A vertical sidebar showing "36 items" and a partial view of a JSON array.

05

Since GPT returns rich text, another JavaScript function parses and sanitizes this output into a usable JSON object. This ensures the enriched data can be processed cleanly without formatting bugs.

Color by Geography

Assign Color Code Execute step

Parameters Settings Docs

Mode: Run Once for All Items

Language: JavaScript

```
JavaScript
20 }
21
22 const colorMap = {
23   Flamingo: "4",
24   Peacock: "1",
25   Blueberry: "9",
26 };
27
28 return {
29   json: {
30     ...item.json,
31     colorName,
32     colorId: colorMap[colorName],
33   }
34 };
35 });
36
```

Type \$ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console.

INPUT Schema Table JSON 36 items

```
{
  "summary": "Taira vs Park",
  "background": "An upcoming UFC fight, part of a series of events leading to larger championships. Known for its main card bouts, featuring notable fighters Taira and Park.",
  "pr_agencies": [
    "WME-IMG",
    "Endeavor"
  ],
  "trending": false,
  "agents": [
    "Ali Abdelaziz",
    "Malki Kawa"
  ],
  "interviews": [
    "MMA Junkie",
    "The MMA Hour"
  ],
  "Country": "United States",
  "Start Date": "2025-08-02T21:00:00.000-04:00",
  "End Date": "2025-08-02T21:00:00.000-04:00",
  "Venue": "Las Vegas, NV"
},
```

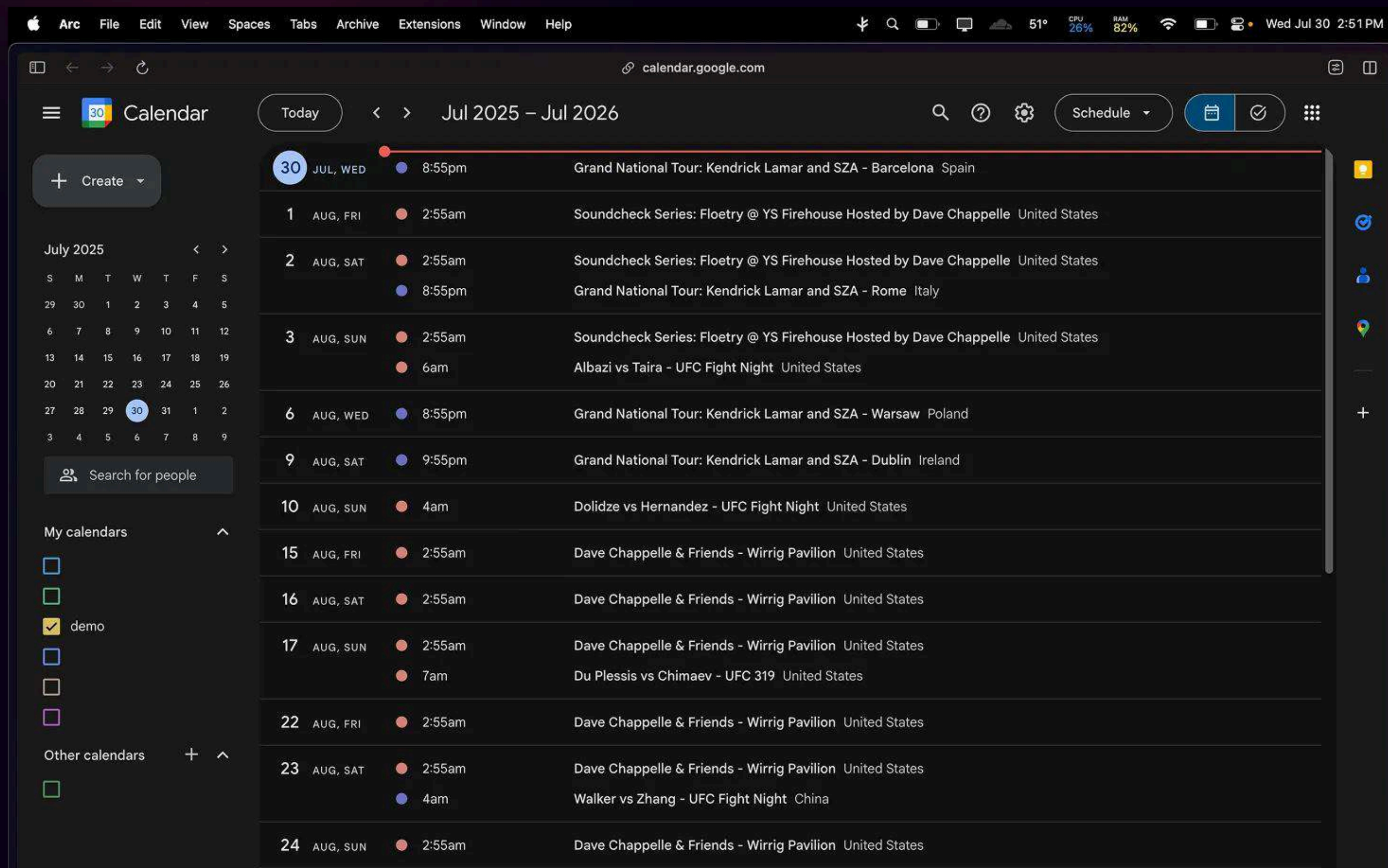
OUTPUT Schema Table JSON 36 items

```
{
  "summary": "Taira vs Park",
  "background": "An upcoming UFC fight, part of a series of events leading to larger championships. Known for its main card bouts, featuring notable fighters Taira and Park.",
  "pr_agencies": [
    "WME-IMG",
    "Endeavor"
  ],
  "trending": false,
  "agents": [
    "Ali Abdelaziz",
    "Malki Kawa"
  ],
  "interviews": [
    "MMA Junkie",
    "The MMA Hour"
  ],
  "Country": "United States",
  "Start Date": "2025-08-02T21:00:00.000-
```

06

Each event was scraped using precise CSS selectors for each of the four websites. This approach eliminated the need for manual copying of event data and centralized event metadata from multiple platforms into one structured format.

Google Calendar Integration



07

Each event was scraped using precise CSS selectors for each of the four websites. This approach eliminated the need for manual copying of event data and centralized event metadata from multiple platforms into one structured format.

04 Conclusion



Conclusion

This project transformed a scattered, time-intensive workflow into a fully intelligent automation pipeline using n8n, OpenAI, and Google services. With complete automation, hours of manual research were eliminated, event scheduling became accurate, detailed, and visual, events were instantly categorized by geography, and the system was designed to be reusable and expandable across any domain.

Thank You