📊 Topic Researcher Agent

AI-Powered Podcast Topic Discovery

AIToday Live Podcast

Intelligent Content Discovery System

What is the Topic Researcher Agent?

Purpose:

Discover 6-8 interesting AI topics for weekly podcast episodes

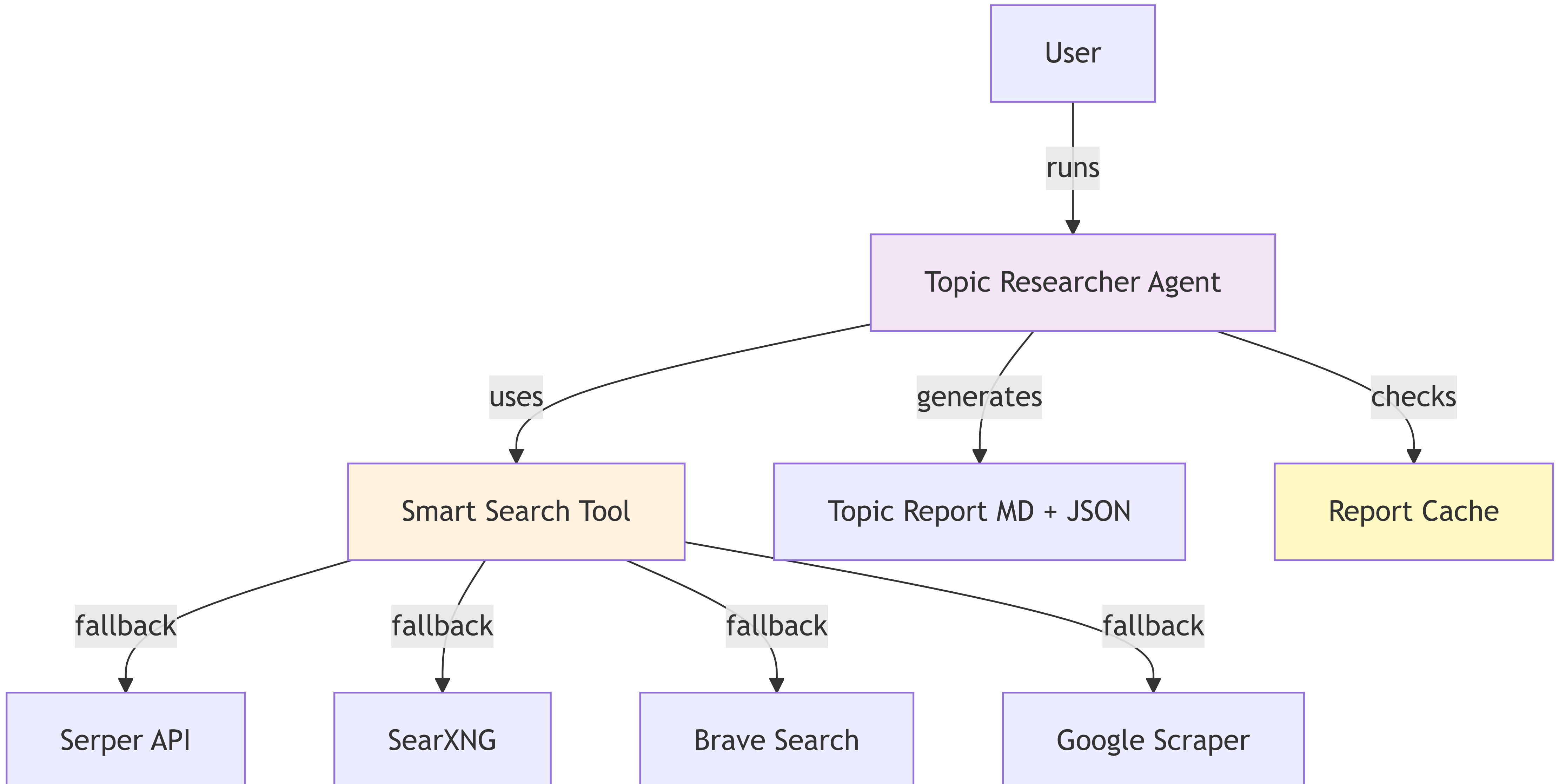
Powered by:

- Claude Sonnet 4 with extended thinking
- Multi-provider web search
- Smart caching system

Output:

- Diverse topic mix across 6 categories
- Actionable discussion angles
- Keywords for guest search integration

System Architecture



Target Persona: Anne de Vries

Who is Anne?

👤 IT Product Owner

- Works at mid-size Dutch company
- Manages product development team
- Tech-savvy but not AI specialist

Characteristics:

- 🚀 Early adopter
- 📚 Knows basic AI concepts
- 💼 Wants practical applications
- 🤔 Critical thinker
- ⚠️ Wants to know what works AND what doesn't

Six Topic Categories

Ensuring Diverse Content:

1. 🔬 **Wetenschappelijk**
   - Research breakthroughs
   - Academic papers
   - Practical implications
2. 💼 **Praktijkvoorbeeld**
   - Dutch organizations using AI
   - Real-world implementations
   - Success stories & failures
3. 📚 **Informatief**
   - AI concept explanations
   - Technology deep-dives
   - How things work

Topic Search Criteria

What Makes a Good Topic:

Time Range:

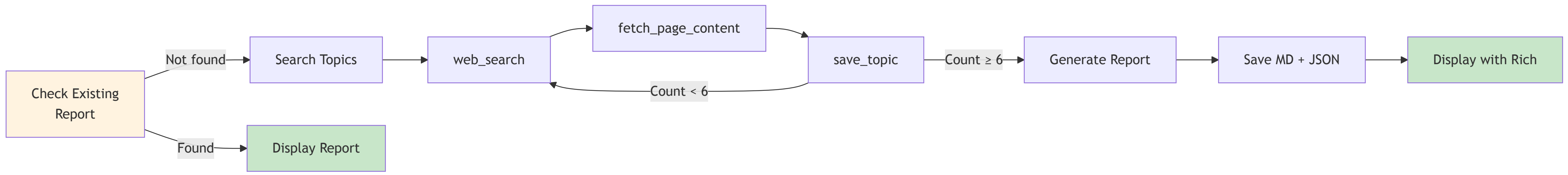
- ✅ Last **month** (not just 2 weeks)
- ✅ Recent enough to be relevant
- ✅ Not too old to be outdated

Quality Criteria:

- Must be interesting for Anne (practical focus)
- Should enable 20-30 minute discussion
- Needs Dutch angle when possible
- Multiple source verification (min 2)

Diversity:

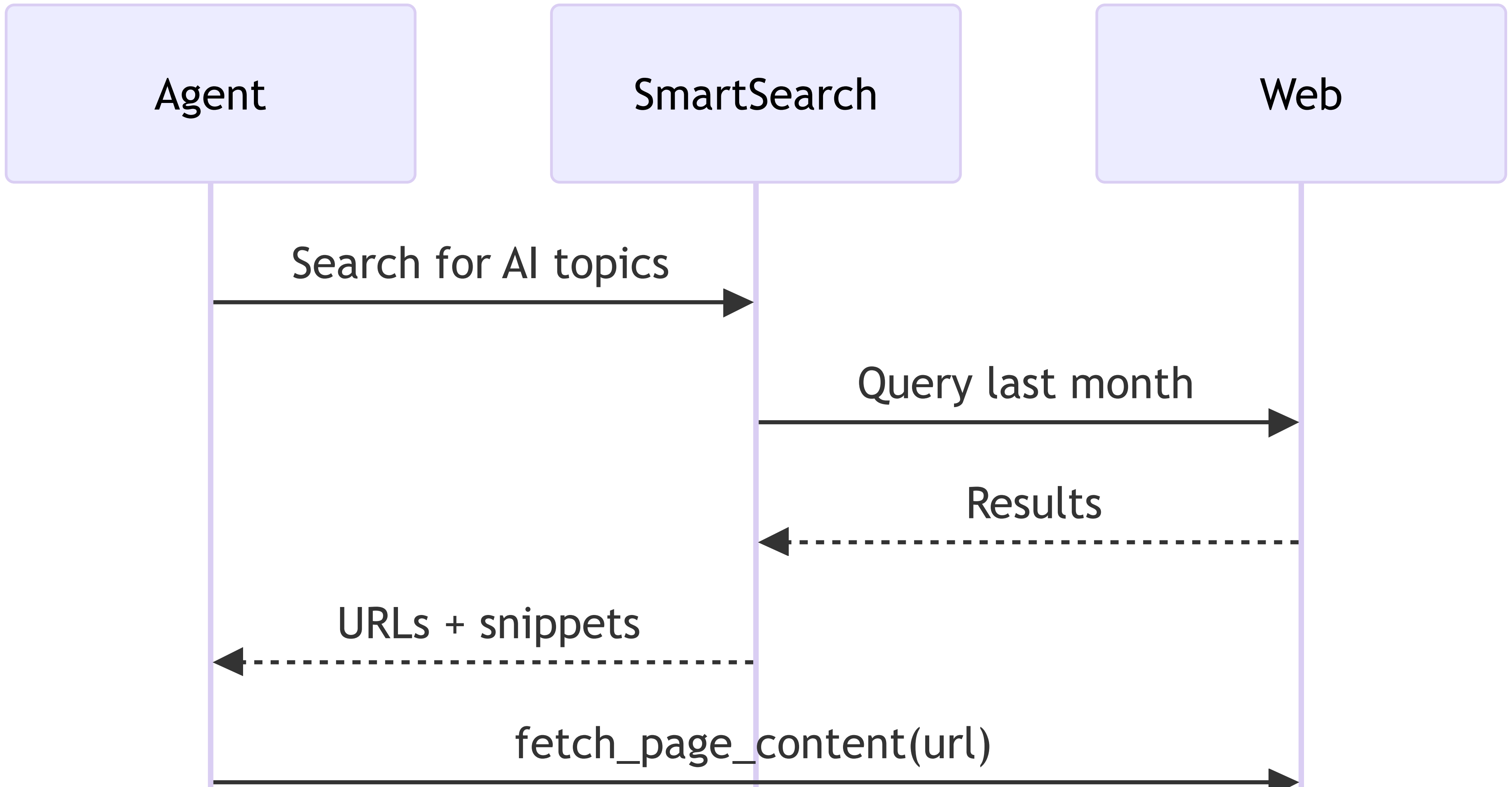
How It Works: Search Flow



Smart Caching: Checks for today's report before searching

Search Process

Step-by-Step:



Report Content Structure

Basic Info:

- Title (max 60 chars)
- Category (1 of 6)
- Description (2-3 sentences)

For Anne:

- Why relevant for her
- Practical angle

For Production:

- **Keywords** → Guest Finder
- Discussion angles (3-4)
- Sources (min 2, prefer NL)

Removed:

- ~~Ideal guest profile~~

Report Example

```markdown
## RAG Implementatie in de Praktijk
**Categorie:** 💼 Praktijkvoorbeeld

**Waarom relevant:** Product owners kunnen deze techniek
direct toepassen voor betere zoekfunctionaliteit.

**Beschrijving:** RAG combineert zoeken met AI.
Nederlandse bedrijven zien 30% verbetering.

**Keywords:** RAG, retrieval augmented generation,
Nederlandse implementaties, vector databases

**Discussie:**
- Wanneer RAG gebruiken?
- Typische valkuilen?
- Kosten vs traditioneel zoeken?

**Bronnen:** [RAG in Productie](https://...) (5 okt)
[Vector DB](https://...) (12 okt)
```

Duplicate Prevention

Smart Caching System:

```
output/topic_reports/
├── week_41_20251012.md      # Today's report
├── week_41_20251012.json    # Structured data
├── week_40_20251005.md      # Last week
└── week_40_20251005.json
```

Behavior:

- ✅ Checks for today's report before searching
- ✅ Shows existing report if found (0 API cost!)
- ✅ Option to force new search (overwrites)
- ✅ One report per day maximum

Benefits: Saves API costs & prevents duplicate work

Report Display

Rich Terminal Rendering:

Features:

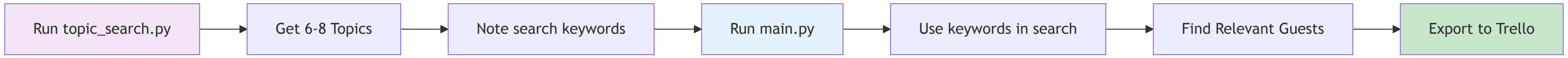
- 📝 Beautiful markdown rendering
- 🎨 Syntax highlighting
- 📋 Formatted headers
- 🔗 Clickable URLs
- ➖ Section dividers
- 📊 Category icons

User Experience:

- Easy to scan
- Professional output
- No external viewer needed
- Terminal-native

Integration with Guest Finder

Workflow: Topics → Guests



Example Integration:

1. Topic: "RAG implementation in healthcare"
2. Keywords: "RAG, retrieval augmented generation, Dutch healthcare"
3. Guest Search: Finds experts implementing RAG
4. Result: Topic-aligned guests

Configuration

Environment Variables:

```
# Required
ANTHROPIC_API_KEY=key

# Recommended (≥1)
SERPER_API_KEY=key
BRAVE_API_KEY=key
```

Settings:

- Time range: Last month
- Target: 6-8 topics
- Sources: Min 2/topic
- Categories: All 6 types

Note: No Trello needed

Usage

Run Topic Search:

```
python topic_search.py
```

What Happens:

1. ✅ Check for today's report
2. ✅ If exists: Show summary + option to view
3. ✅ If not: Run search (6-8 topics)
4. ✅ Generate report (MD + JSON)
5. ✅ Display with Rich markdown
6. ✅ Save to `output/topic_reports/`

Daily Workflow:

Output Files

Directory Structure:

```
output/
└── topic_reports/
    ├── week_41_20251012.md      # Markdown report
    ├── week_41_20251012.json    # Structured data
    ├── week_40_20251005.md
    └── week_40_20251005.json
```

File Naming:

- Format: `week_{week}_{date}.{ext}`
- Example: `week_41_20251012.md`
- Week number from ISO calendar
- Automatically dated

Encoding: UTF-8, git-friendly

JSON Output Structure

Structured Data for Automation:

```json
{
  "week": 41,
  "year": 2025,
  "date_generated": "2025-10-12T10:30:00",
  "topics": [
    {
      "title": "RAG Implementatie in de Praktijk",
      "category": "Praktijkvoorbeeld",
      "description": "...",
      "why_relevant_for_anne": "...",
      "search_keywords": "RAG, retrieval...",
      "discussion_angles": ["...", "..."],
      "sources": [
        {
          "url": "https://...",
          "title": "...",
          "date": "2025-10-05"
        }
      ]
    }
  ]
```

Testing Coverage

15 Tests - All Passing ✅

Agent:

- Initialization
- Tool definitions
- Tool handling
- Report generation
- Error handling

Validation:

- Category validation
- Topic structure
- Source requirements

Performance & Costs

API Usage per Run:

Token Breakdown:

- Search: ~15,000 tokens
  - 6-8 topic searches
  - Page content analysis
- Report: ~3,000 tokens
  - Markdown generation
  - JSON structuring

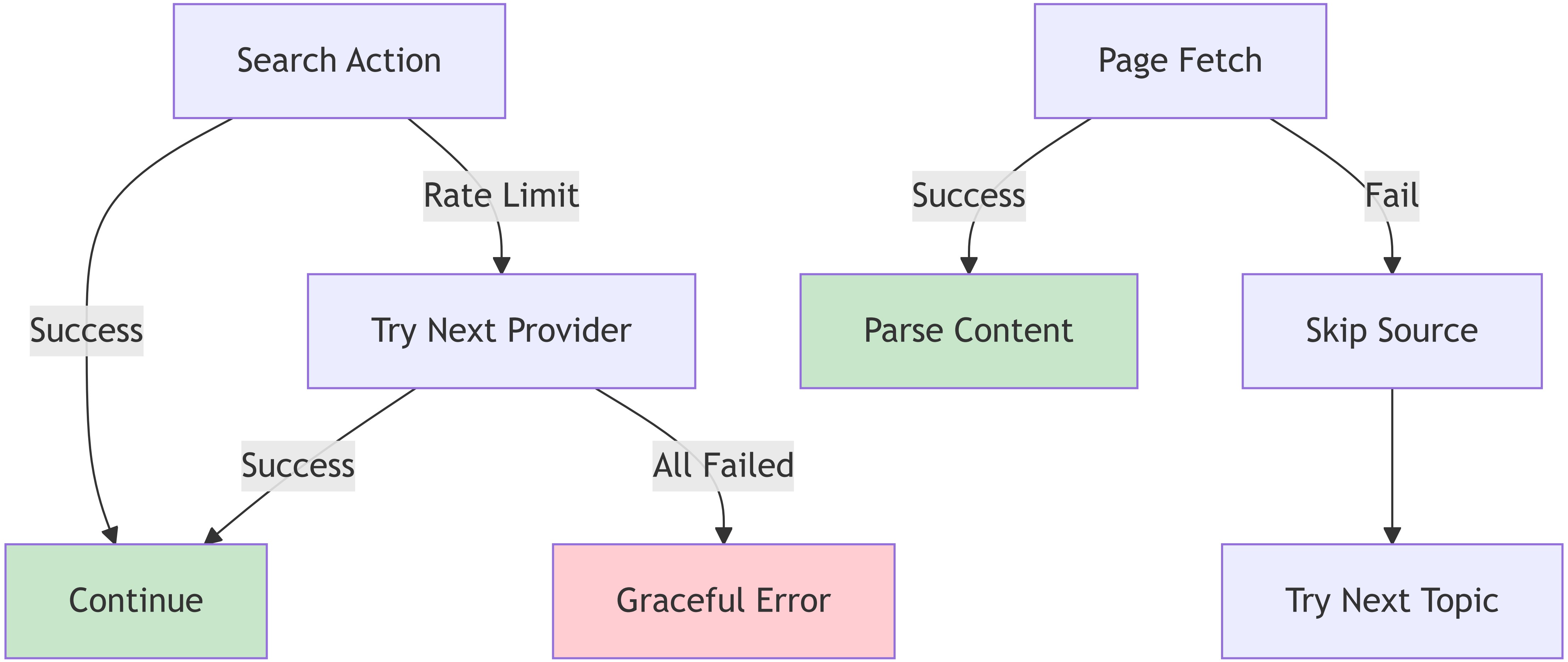
Total: ~18,000 tokens/run

Estimated Costs:

- Claude Sonnet 4: ~$0.30/run
- Search APIs: Free tier sufficient

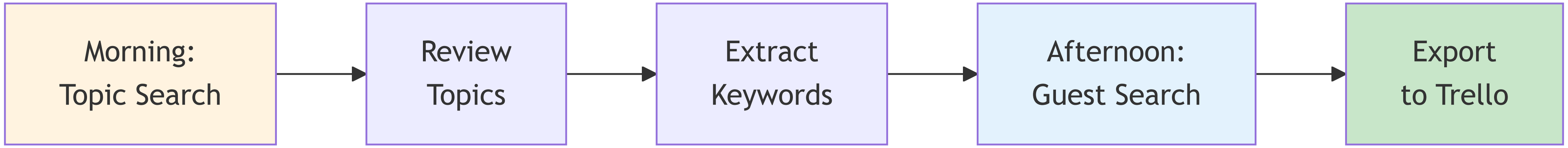
Error Handling

Robust Fallback System:



Daily Workflow

Recommended Process:



Steps:

1. **09:00** - Run `python topic_search.py`
2. **09:15** - Review 6-8 topics, extract keywords
3. **14:00** - Run `python main.py` with keywords
4. **14:30** - Select guests, export to Trello
5. **Weekly** - Review and clean up Trello board

Best Practices

Content Guidelines:

Do:

- ✅ Focus on practical applications
- ✅ Include Dutch angle when possible
- ✅ Balance opportunities & warnings
- ✅ Verify multiple sources
- ✅ Extract clear keywords for guest search

Don't:

- ❌ Run search twice same day (use cache)
- ❌ Include topics older than 1 month
- ❌ Focus only on one category

Tech Stack

Core Technologies:

- Python 3.10+
- Anthropic Claude Sonnet 4
- Rich (Terminal UI)
- BeautifulSoup4 (HTML)
- Requests (HTTP)

Search Providers:

- Serper API
- SearXNG
- Brave Search API
- Google Scraper

Testing:

- Pytest (15 tests)
- Unittest.mock

Key Features Summary

What Makes It Special:

- ✅ **Persona-targeted** - Content for Anne specifically
- ✅ **Six diverse categories** - Balanced content mix
- ✅ **Last month timeframe** - Not just 2 weeks
- ✅ **Smart caching** - One report per day
- ✅ **Rich rendering** - Beautiful terminal output
- ✅ **JSON + MD output** - Machine & human readable
- ✅ **Guest integration** - Keywords for guest search
- ✅ **Cost-effective** - ~$0.30 per search, $0 cached

Future Enhancements

Potential Features:

Content:

- 🎯 Multiple personas (beyond Anne)
- 🌐 Multi-language support
- 📊 Topic trend analysis
- 🔔 Weekly auto-scheduling

Integration:

- 📱 Slack/Discord notifications
- 📧 Email digest format
- 🤖 Auto-suggest episode structure
- 📈 Analytics dashboard

Contributions welcome!

Repository: https://github.com/Joopsnijder/quest_search

Documentation

Available Resources:

- 📘 README.md - Quick start & overview
- 📗 USAGE.md - Complete usage guide
- 📙 RATE_LIMIT_HANDLING.md - Search details
- 📊 TEST_COVERAGE_SUMMARY.md - Test details
- 🏗️ architecture.md - Arc42 documentation

All docs: Maintained and up-to-date

Comparison: Before vs After

Before (Manual Process):

- 🕐 2-3 hours browsing news sites
- 🤷 Inconsistent topic diversity
- 😓 Repetitive content patterns
- 📋 Manual note-taking

After (Topic Researcher Agent):

- ⚡ 5 minutes automated search
- 🎯 Guaranteed category diversity
- 🔄 Fresh perspectives weekly
- 📊 Structured JSON output
- 💰 Cost: ~$0.30 per week

Demo Time! 🎬

Let's see the Topic Researcher in action:

1. Check for existing report
2. Run new topic search
3. Category diversity
4. Rich markdown display
5. JSON output structure
6. Keyword extraction for guest search

Questions? 💬

Repository: https://github.com/Joopsnijder/guest_search

License: MIT

Built with: Claude Code by Anthropic

Thank You! 🎉

Happy Topic Discovery! 📊 🎙️