IdeaFactory — User Manual

An MCP Server for Idea Development & Daily Review

Leo

July 2026

Contents

1 Introduction

IdeaFactory is an **idea development engine** that lives inside your development workflow as an MCP (Model Context Protocol) server. It helps you capture, nurture, and mature ideas until they are ready to become real projects.

1.1 Core Concepts

The Maturity Score (0-10) Every idea has a dynamic maturity score that represents how much external validation and momentum the idea has. The score rises when search results contain signal keywords (funding news, partnerships, launches, etc.) and decays slowly when an idea isn't reviewed. 10 means "ready to build," 0 means "not yet validated."

The Daily Review Once per day (via systemd timer), IdeaFactory scans all active ideas, runs web searches against their keywords, and adjusts their maturity scores. This is the heartbeat of the system — it keeps your idea pipeline fresh without manual effort.

The Promotion Pipeline When an idea's maturity score crosses the threshold (>=7), it becomes a *promotion candidate*. You can approve the promotion — which graduates the idea into a full project with a generated project brief — or request more research.

1.2 Dual-Mode Architecture

IdeaFactory runs in two modes:

Mode	Entry Point	Purpose
MCP Server	`idea-factory` (no args)	Supplies MCP tools to Claude Code for interactive use
Daily Review	`idea-factory --daily-review`	One-shot review scan, runs as a cron job

2 Installation & Setup

2.1 Prerequisites

- **Bun v1.x** runtime
- **Brave Search API key** (free tier: 2,000 queries/month)
- Linux with systemd (for daily cron)

2.2 Build from Source

```
# Clone and build
cd /opt/projects/ideafactory
bun install
bun build --compile src/main.ts --outfile idea-factory

# Install system-wide
sudo cp idea-factory /usr/local/bin/
```

2.3 Configuration

Configuration is stored in `~/.ideas/config.json`:

```
{
  "brave_api_key": "your-brave-api-key",
```

```
  "smtp_host": "your-smtp-host",
  "smtp_user": "your-smtp-user",
  "smtp_password": "your-password"
}
```

Field	Required	Purpose
`brave_api_key`	**Yes**	Brave Search API key
`smtp_host`	No	SMTP server for email error alerts
`smtp_user`	No	SMTP username
`smtp_password`	No	SMTP password

2.4 Directory Layout

```
~/.ideas/
  config.json          Configuration file
  ideas.db             SQLite database (WAL mode)
  projects/            Generated project briefs on promotion
    <idea-id>/
      project-brief.md
  research/            Generated research documents
    <idea-id>-<timestamp>.md
  backups/             Database backups from VACUUM INTO
```

2.5 MCP Configuration

For Claude Code to use IdeaFactory as an MCP server, add this to your project's `.claude/mcp.json` or `~/.claude/mcp.json`:

```
{
  "mcpServers": {
    "idea-factory": {
      "command": "/usr/local/bin/idea-factory",
      "args": []
    }
  }
}
```

2.6 Systemd Timer (Daily Review)

```
# Install units
sudo cp scripts/idea-factory-daily-review.service /etc/systemd/system/
sudo cp scripts/idea-factory-daily-review.timer /etc/systemd/system/
sudo systemctl daemon-reload

# Enable the timer to run once daily
sudo systemctl enable idea-factory-daily-review.timer
sudo systemctl start idea-factory-daily-review.timer
```

The timer fires daily at midnight with a randomized delay of up to 30 minutes. Uses `Persistent=true` so a missed run fires on the next boot.

3 How to Use the Tools

IdeaFactory tools are MCP (Model Context Protocol) tools. They are available inside **Claude Code** whenever you are in the IdeaFactory project directory or have the MCP server configured globally.

You do not type tool commands directly. Instead, you just ask in plain English. Claude Code decides which tool to call and fills in the parameters.

For example, to create an idea you would say to Claude Code:

> *"I have an idea for an AI resume parser. Keywords: resume parsing, HR tech. Tags: saas, b2b."*

Claude Code will ask for your permission, then call `idea.create` with the details. The result appears in the conversation.

To run a daily review:

> *"Check on my ideas — run the daily review."*

To see promotion candidates:

> *"Any of my ideas ready to become projects?"*

To promote one:

> *"Approve the resume parser for promotion."*

If you want to request more research on a specific idea:

> *"Tell the daily review to look into competitor pricing for the resume parser."*

This uses the `request-info` action internally — Claude Code handles the mapping.

To run a deep research pass:

> *"Do a deep research pass on my AI resume parser idea."*

The key point: **just talk naturally about your ideas.** Claude Code invokes the appropriate tool, shows you what it found, and asks for permission before making changes.

3.1 Permissions

Each time Claude Code wants to call a tool, it will show you the tool name and parameters and ask you to approve. You can:

- **Allow once** — runs the tool this time
- **Allow always** — runs the tool this time and remembers the decision for the session
- **Deny** — skips it

3.2 MCP Tool Reference

The full list of available tools:

3.3 idea.list

List all ideas, optionally filtered by status.

Parameters:

Name	Type	Required	Description
`status`	string	No	Filter: `idea`, `researching`, `promotion-candidate`, `project`, `archived`

Example:

List all active ideas (shows the count of recent errors if any):

Returns a JSON array of ideas sorted by creation date (newest first).

3.4 idea.create

Capture a new idea.

Parameters:

Name	Type	Required	Description
`title`	string	**Yes**	Idea title (max 200 chars)
`tags`	string[]	No	Categorization tags
`keywords`	string[]	No	Keywords for daily review web searches
`notes`	string	No	Free-form notes
`status`	string	No	`idea` (default) or `researching`

Example:

Creates the idea, generates a URL-safe slug as the ID, initializes its maturity score at 0, and sets `last_reviewed` to today.

3.5 idea.update

Modify an existing idea's metadata.

Parameters:

Name	Type	Required	Description
`id`	string	**Yes**	Idea ID (the slug)
`title`	string	No	New title
`status`	string	No	Any valid status
`tags`	string[]	No	Replace tags
`keywords`	string[]	No	Replace keywords
`notes`	string	No	Replace notes
`maturity_score`	number	No	Override score (0-10)

Name	Type	Required	Description
`review_directive`	string, null	No	Set or clear the review directive

3.6 idea.delete

Permanently remove an idea from the database.

Parameters:

Name	Type	Required	Description
`id`	string	**Yes**	Idea ID

[**!**] **Irreversible.** Consider archiving instead.

3.7 idea.archive

Shelve an idea without deleting it. Archived ideas are excluded from daily reviews but remain in the database and can be unarchived.

Parameters:

Name	Type	Required	Description
`id`	string	**Yes**	Idea ID

3.8 idea.errors

View recent errors from daily review runs or tool operations.

Parameters:

Name	Type	Required	Description
`limit`	number	No	Max errors to show (1-50, default 10)

Returns a JSON array with timestamp, source (which idea or component), and error message.

3.9 idea.daily-review

Trigger an immediate daily review scan across all active (non-archived, non-project) ideas. Each idea is searched independently; errors in one idea do not affect others.

Parameters: None.

Returns: A summary table:

```
my-idea: 5->7 (+2), hits=2 [CANDIDATE]
stale-idea: 4->3 (-1), hits=0
```

Ideas flagged `[CANDIDATE]` have crossed the promotion threshold (score >= 7) and should be reviewed for promotion.

3.10 idea.research

Run a deep multi-search research pass on a specific idea and save the results to a timestamped markdown file.

Parameters:

Name	Type	Required	Description
`idea_id`	string	**Yes**	Idea ID to research

How it works:

1. Builds 3-5 varied queries from the idea's keywords and tags
2. Adds a broad context query and a "news" query if not already present
3. Searches each query via Brave Search (capped at 5 results per query)
4. Saves a formatted markdown file to `~/.ideas/research/<id>-<timestamp>.md`
5. Returns the file path

3.11 idea.promotion-candidates

List ideas whose maturity score is >=7 and that are not already projects or archived.

Parameters: None.

Returns: A JSON array with candidate details including:

Field	Description
`id`	Idea ID
`title`	Idea title
`maturity_score`	Current score
`keywords_matched`	Idea's tracked keywords
`top_signals`	Recent search result titles from daily reviews
`promotion_count`	Consecutive days flagged as promotable
`last_reviewed`	Date of last review

3.12 idea.promote

Act on a promotion candidate: either approve it to graduate to a project, or request more research.

Parameters:

Name	Type	Required	Description
`action`	string	**Yes**	`"approve"` or `"request-info"`
`idea_id`	string	**Yes**	Idea ID
`notes`	string	No	Directive text for request-info; optional notes for approve

Action: approve

- Sets the idea status to `"project"`
- Creates `~/.ideas/projects/<id>/project-brief.md` with:
  - Summary and metadata
  - Research history (links to existing research files)
  - Key signals from recent daily reviews
  - Top 3 reasons to build now
  - Suggested next step

Action: request-info

- Sets the idea's `review_directive` field
- The directive is included as a search query in the next daily review
- Useful for gathering targeted information ("Find competitor pricing", "Check for regulatory changes")

4 The Maturity Scoring Engine

The scoring engine is the brain of IdeaFactory. It determines whether an idea is getting hotter or growing stale.

4.1 Signal Keywords

When search results come back from a daily review, IdeaFactory scans them for these 20 signal keywords:

Category	Keywords
Funding	`funding`, `investment`, `ipo`
Growth	`launch`, `growth`, `expansion`, `record`, `hiring spree`
Business	`acquisition`, `partnership`, `milestone`, `breakthrough`, `announced`
Technology	`new tool`, `regulation`, `open source`, `release`, `patent`, `approval`, `disruption`

Matching is case-insensitive and searches both title and snippet text.

4.2 Scoring Formula

Each daily review computes a score delta:

```
delta = min(keyword_hits, 2) - floor(max(0, days_since_last_review) / 30)
```

Factor	Range	Details
Keyword hits bonus	+0 to +2 per review	Capped at 2 hits regardless of how many match
Time decay	-1 per 30 days	Rounds down: 60 days = -2, 90 days = -3, etc.

Factor	Range	Details
No decay before 30 days	0	Decay only applies after a full 30-day window

The new score is clamped to [0, 10].

4.3 Notes Score Bonus

A one-time +1 bonus is applied when notes are substantive:

- Multi-line notes (>=2 non-empty lines), **or**
- Notes exceeding 80 characters

This bonus is applied when the notes field is set, not recalculated every review cycle.

4.4 Manual Bump

A constant `MANUAL_BUMP = 2` is available for when you want to manually signal that an idea is being actively pursued. This is applied externally via the `idea.update` tool.

4.5 Examples

Prior Score	Hits	Days Since Review	Delta	New Score
5	2	1	+2	7
7	0	30	-1	6
3	1	60	-1	2
8	2	90	-1	7
5	0	1	0	5

5 The Daily Review Pipeline

The daily review is the core automated process that keeps idea scores current.

5.1 Flow

```
Start
  |
  |- List all non-archived, non-project ideas
  |
  |- For each idea (parallel-safe, sequential in practice):
  |     |
  |     |- Build search queries from keywords + review_directive
  |     |
  |     |- Run Brave Search for each query
  |     |     (stop early on 429 rate limit)
  |     |
  |     |- Count signal keyword hits across all results
```

```
|     |
|     |- Compute time decay from last_reviewed date
|     |
|     |- Apply delta -> new maturity score
|     |
|     |- Update idea row in database
|     |
|     |- Log daily_review row (date, results, hits, errors)
|     |
|     \- Flag if promotion candidate (score >= 7)
|
\- Return summary results
```

5.2 Error Isolation

Each idea is reviewed in its own try/catch block. If one idea's search fails (network error, timeout, unexpected response), it is logged to the error_log table and processing continues with the next idea. The failed idea retains its previous score.

5.3 Rate Limiting

The Brave Search free tier allows 2,000 queries/month. If a 429 response is received, the review for that idea stops early and subsequent ideas in the same run continue. The rate limit error is recorded per-idea.

5.4 CLI Mode

Run the review manually at any time:

```
idea-factory --daily-review
```

Example output:

```
Daily review complete --- 3 ideas reviewed.

  ai-app: 5->7 (+2)  hits=2 * CANDIDATE
  blog-tool: 3->3 (0)  hits=0
  3d-printer: 8->7 (-1)  hits=0

1 idea(s) ready for promotion!
Run idea.promotion-candidates in Claude Code to review.
```

6 Promotion & Project Briefs

When an idea repeatedly scores >=7, it's time to consider making it a real project.

6.1 Reviewing Candidates

Use `idea.promotion-candidates` to see which ideas are ready. Each candidate shows:

- Its current maturity score

- Keywords that were searched
- Top signals from recent search results (titles only)
- How many consecutive days it has been flagged as promotable

6.2 Approving a Promotion

Run `idea.promote` with `action: "approve"`. This does three things:

1. **Creates a project brief** at `~/.ideas/projects/<id>/project-brief.md` containing a structured summary with research links, signal history, and next steps.
2. **Sets status to `"project"`**, removing the idea from daily reviews.
3. **Retains the idea** in the database as a record.

6.3 Requesting More Info

If an idea is borderline or needs more validation, use `action: "request-info"` with a directive like "Find competitor pricing." The directive becomes a search query in subsequent daily reviews, helping narrow the research.

7 Research Tool

The research tool (`idea.research`) provides a deeper, one-off investigation of an idea beyond the daily review's automated scan.

7.1 How Queries Are Built

Given an idea with keywords `["machine learning", "fintech"]` and tags `["saas", "api"]`, the research tool builds up to 5 queries:

1. `machine learning` (first keyword)
2. `fintech` (second keyword)
3. `saas api machine learning` (tags + first keyword)
4. `machine learning news` (news variant)
5. `fintech news` (if applicable)

7.2 Output

Results are saved to `~/.ideas/research/<idea-id>-<ISO-timestamp>.md` with full metadata: query used, result title, URL, snippet, and a note of any errors encountered.

8 Database Schema

IdeaFactory uses SQLite via `bun:sqlite` with WAL journal mode for concurrent read safety (important when cron and MCP may access simultaneously).

8.1 ideas

Column	Type	Description
`id`	TEXT PK	URL-safe slug from title
`title`	TEXT	Idea name
`status`	TEXT	`idea`, `researching`, `promotion-candidate`, `project`, `archived`
`tags`	TEXT	JSON array of tags
`created`	TEXT	ISO date
`last_reviewed`	TEXT	ISO date, updated on each review
`maturity_score`	INTEGER	0-10
`promotion_count`	INTEGER	Consecutive days flagged as candidate
`keywords`	TEXT	JSON array of search keywords
`notes`	TEXT	Free-form notes
`review_directive`	TEXT	Targeted research directive (nullable)

8.2 daily_reviews

Column	Type	Description
`id`	INTEGER PK	Auto-increment
`date`	TEXT	Review date
`idea_id`	TEXT FK	References ideas(id)
`results`	TEXT	JSON array of search results (max 10)
`keyword_hits`	INTEGER	Signal keywords matched (capped at 2)
`errors`	TEXT	Error message if any (nullable)

8.3 error_log

Column	Type	Description
`id`	INTEGER PK	Auto-increment
`ts`	TEXT	Datetime of error
`source`	TEXT	Component that generated the error
`message`	TEXT	Error message (max 500 chars)
`details`	TEXT	Additional context (max 2000 chars)

9 Error Handling

9.1 Logging

All errors from daily reviews and tool operations are recorded in the `error_log` table with a timestamp, source, and message. The log persists across sessions.

9.2 Monitoring

The `idea.list` tool automatically appends a warning if errors have been logged in the last 24 hours:

```
[!] 3 error(s) in the last 24h. Use idea.errors to see details.
```

9.3 SMTP Alerts (Future)

The configuration file supports SMTP credentials for email alerts when errors exceed thresholds. This is wired into the config schema but the email dispatch is not yet implemented.

9.4 Safe-by-Design

- Error logging itself cannot crash the process (it catches its own exceptions and falls back to stderr)
- Per-idea error isolation in daily reviews means one broken search doesn't cascade
- Rate-limited searches stop per-idea but don't halt the entire review

10 Operational Workflows

10.1 Daily Routine

The systemd timer runs the daily review automatically. In the morning:

1. **Check scores**: `idea.list` shows all ideas and their current scores.
2. **Review candidates**: `idea.promotion-candidates` for anything >=7.
3. **Promote or query**: Use `idea.promote` to approve deserving ideas or request more info on borderline ones.
4. **Deep research**: `idea.research` on high-potential ideas to gather material before promoting.
5. **Triage errors**: `idea.errors` if the score display shows warnings.

10.2 Idea Curation

- **Add regularly**: Capture ideas as they come with `idea.create`. Even a title alone is enough — keywords can be added later.
- **Set good keywords**: The quality of daily reviews depends on keywords. Specific > generic. "machine learning fraud detection" > "AI startup."
- **Use review directives**: For ideas that need targeted validation, set a review directive with `request-info` or `idea.update`.
- **Prune stale ideas**: If an idea decays to 0 and shows no signal, consider archiving it.

10.3 Promotion Checklist

Before approving a promotion:

☐ Score >= 7 for multiple consecutive days (check `promotion_count`)
☐ Review signals: do the search results show real market activity?
☐ Run a deep research pass: `idea.research`
☐ Review the generated project brief
☐ Create the actual project repo and start building

11 Complete Example

11.1 1. Capture Ideas

```
idea.create -> title: "AI-Powered Resume Parser"
```

```
               keywords: ["resume parsing", "HR tech", "AI recruitment"]
               tags: ["saas", "b2b"]

idea.create -> title: "Personal Knowledge Graph"
               keywords: ["knowledge graph", "note-taking", "personal AI"]
               tags: ["consumer", "ai"]
```

11.2 2. Let the Daily Review Run

Over days/weeks, the daily review searches the web for each idea's keywords. When funding rounds, product launches, or industry news match, the score climbs.

11.3 3. Check Candidates

```
idea.promotion-candidates
-> AI-Powered Resume Parser (score 8, 4 days as candidate)
   top signals: "HR Tech raises $40M Series B",
                "AI recruitment market projected to reach $1.2B by 2028"
```

11.4 4. Deep Research

```
idea.research -> idea_id: "ai-powered-resume-parser"
-> Research saved to ~/.ideas/research/ai-powered-resume-parser-2026-07-24.md
```

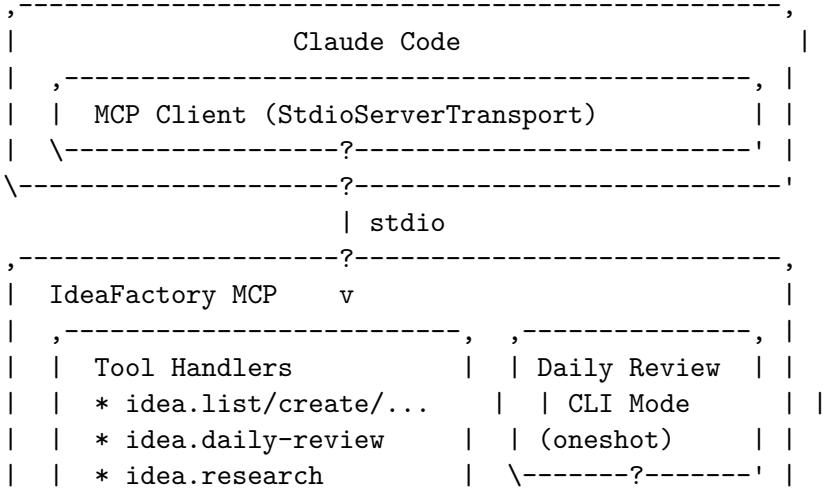
11.5 5. Promote

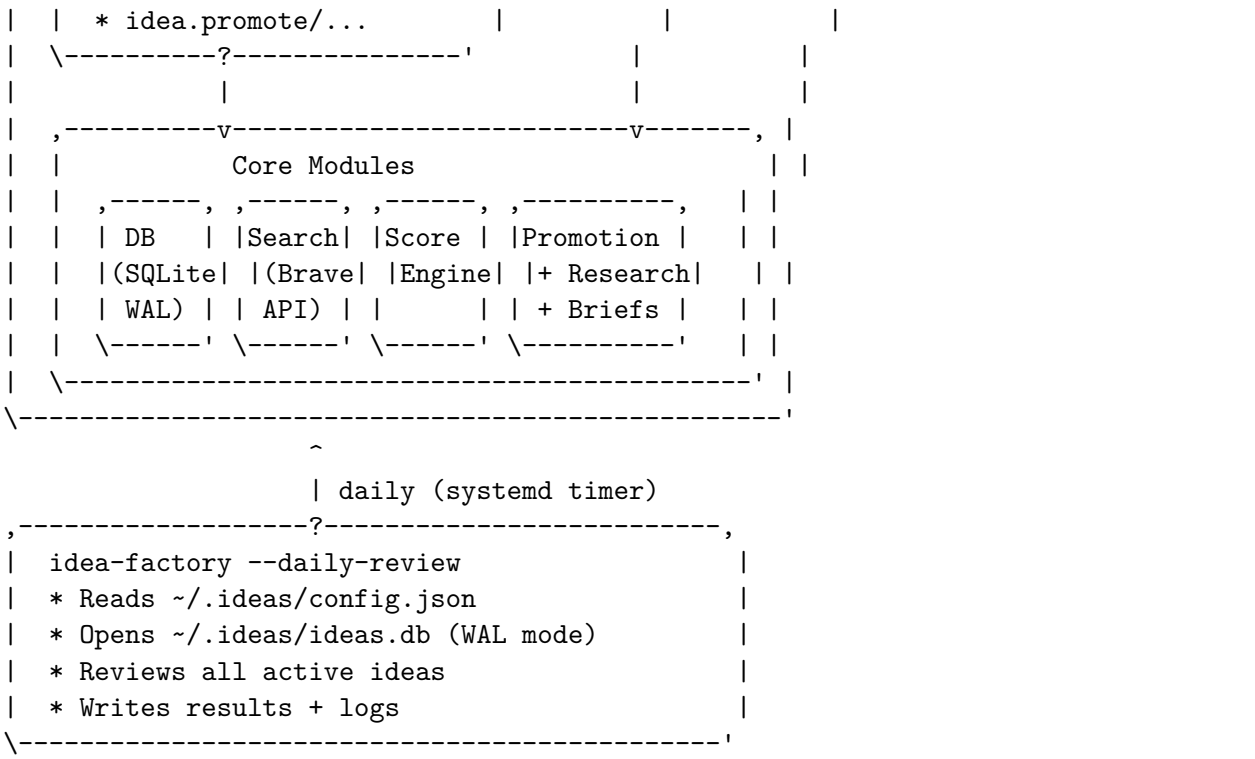
```
idea.promote -> action: "approve", idea_id: "ai-powered-resume-parser"
-> Promoted to project. Brief: ~/.ideas/projects/ai-powered-resume-parser/project-brief.md
```

11.6 6. Build

Read the project brief, create a repo, and start Phase 1.

12 Technical Architecture





12.1 Key Design Decisions

Why MCP? MCP provides a natural interface for AI-assisted idea development. Tools appear directly in Claude Code and can be composed with other operations.

Why SQLite + WAL? SQLite is zero-config, portable, and the WAL journal mode allows concurrent reads from the daily-review cron job and interactive MCP access without locking conflicts.

Why Brave Search? Free tier (2,000 queries/month) is sufficient for daily reviews of dozens of ideas. With ~3 queries per idea per day, 20 ideas use ~1,800 queries/month.

Why a compiled binary? `bun build --compile` produces a single ~100 MB standalone ELF binary with no runtime dependencies. No Node.js, no Python, no package management — just one file on the path.

13 Deployment Script

The `scripts/deploy.sh` script automates installation:

```
#!/usr/bin/env bash
# 1. Builds the binary (bun build --compile)
# 2. Installs to /usr/local/bin/idea-factory
# 3. Installs systemd units (service + timer)
# 4. Enables and starts the daily-review timer
# 5. Configures global MCP in ~/.claude/mcp.json
```

Run it after making changes to redeploy.

Generated from the IdeaFactory codebase v0.1.0