# **🎓 Documentation: AI Content Creator Agent (MCP Server)**

Contents

[**🎓 Documentation: AI Content Creator Agent (MCP Server)** 1](#_Toc198494407)

[**1. Introduction** 2](#_Toc198494408)

[**2. Problem Statement** 2](#_Toc198494409)

[**3. Objective** 3](#_Toc198494410)

[**4. System Design & Architecture** 3](#_Toc198494411)

[4.1 High-Level Architecture 3](#_Toc198494412)

[4.2 Core Technologies 5](#_Toc198494413)

[**5. Agent Roles** 5](#_Toc198494414)

[5.1 Outliner Agent 5](#_Toc198494415)

[5.2 Writer Agent 5](#_Toc198494416)

[5.3 Proofreader Agent 5](#_Toc198494417)

[5.4 Citation Inserter Agent 5](#_Toc198494418)

[5.5 SEO Agent 6](#_Toc198494419)

[5.6 Social Agent 6](#_Toc198494420)

[5.7 Editor Agent 6](#_Toc198494421)

[**6. System Components and Feature Walkthrough** 6](#_Toc198494422)

[**7. Dashboard & Insights** 11](#_Toc198494423)

[**8. Output Structure** 13](#_Toc198494424)

[**9. Execution Flow** 13](#_Toc198494425)

[**10. Future Enhancements** 14](#_Toc198494426)

[**11. Conclusion** 16](#_Toc198494427)

[**12. Appendix** 16](#_Toc198494428)

## **1. Introduction**

In today’s digital landscape, content creation is both essential and labor-intensive, demanding a blend of creativity, research, and technical skill. With the advent of large language models (LLMs), there is a transformative opportunity to streamline and automate parts of this process. This project introduces the **AI Content Creator Agent** — an intelligent, end-to-end content automation system designed to generate, enhance, and publish blog content with minimal human input.

Inspired by the emerging concept of **Model Context Protocol (MCP) servers**, this system adopts a **multi-agent architecture** where specialized AI agents (e.g., Writer, Proofreader, SEO Optimizer, Social Media Strategist) collaborate to produce high-quality, SEO-optimized blog articles. Although it does not implement the official MCP SDK, the design reflects key MCP principles such as modularity, prompt orchestration, task-specific agents, and autonomous workflow execution.

The solution is delivered through a user-friendly **Streamlit interface**, allowing users to input blog topics, choose tone and audience, and optionally preview AI-generated outlines. Once generated, the blogs are saved in multiple formats, published to GitHub Pages (with RSS support), and auto-syndicated to Dev.to. It also supports manual Medium imports.

This project demonstrates how AI-powered multi-agent systems can significantly accelerate digital publishing workflows, offering a scalable blueprint for intelligent content automation.

## **2. Problem Statement**

Manual content creation involves:

* Research and topic ideation
* Structuring and outlining
* Writing and proofreading
* Optimizing for SEO
* Generating social media content
* Publishing and maintaining consistency

These steps require significant time, coordination, and tools. The problem lies in the lack of a unified, automated pipeline that bridges content strategy, creation, and distribution.

## **3. Objective**

To build an AI-powered server that:

* Accepts a topic and outputs a complete blog ecosystem
* Integrates agents for each stage of content generation
* Publishes output in multiple formats and platforms
* Provides analytics and planning tools

## **4. System Design & Architecture**

### 4.1 High-Level Architecture

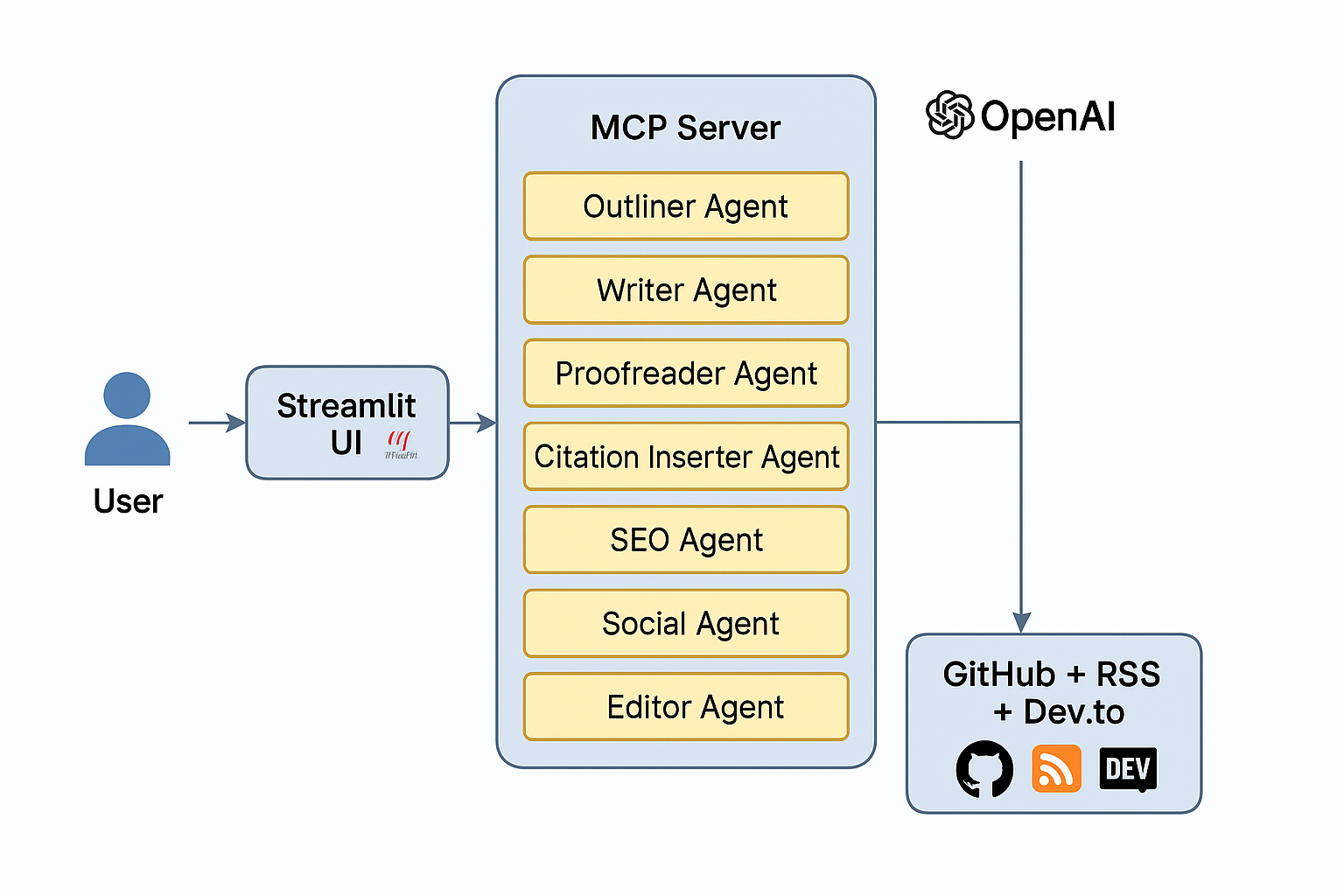


Figure 4. 1 High Level Architecture Diagram of the System

The high-level architecture of the AI Content Creator Agent follows a modular pipeline that connects user input to intelligent content generation and automated publishing. At its core, the system is structured as a web-accessible application powered by **Streamlit**, serving as the user interface layer.

Users initiate the workflow by submitting a blog topic and configuration parameters (tone, audience) through the UI. This input is passed into a coordinated **backend orchestration layer**, implemented in **Python** using the **LangChain** framework and powered by **OpenAI GPT-4o**.

A sequence of intelligent agents is triggered to process the input in stages—from outlining and writing to proofreading and optimizing. Each agent performs a dedicated role, and intermediate outputs are passed downstream to ensure contextual continuity.

Finalized content and metadata are stored in structured directories and rendered into static **HTML pages**. These pages are deployed to **GitHub Pages**, while an automatically maintained **RSS feed** (rss.xml) facilitates external syndication—particularly for platforms like **Dev.to**. The system also supports optional manual publishing to **Medium**.

This architecture ensures clean separation of concerns, scalability, and end-to-end automation, while maintaining user oversight through the UI and queue-based content generation management.

### 4.2 Core Technologies

| **Component** | **Role** |
| --- | --- |
| Python | Backend logic & AI orchestration |
| LangChain | Agent routing, prompt management |
| OpenAI GPT-4o | LLM for all agent outputs |
| Streamlit | Web interface |
| GitHub Pages | Hosting for generated HTML + RSS |
| Dev.to | Auto-publishing via RSS |
| Matplotlib/Calplot | Visual analytics for dashboard |

## **5. Agent Roles**

### 5.1 Outliner Agent

Creates a structured outline based on topic and target audience.

### 5.2 Writer Agent

Writes a full draft following the outline, tone, and audience preferences.

### 5.3 Proofreader Agent

Polishes grammar, clarity, and flow.

### 5.4 Citation Inserter Agent

Adds reputable references and links.

### 5.5 SEO Agent

Generates:

* Meta description
* SEO tags
* Estimated reading time

### 5.6 Social Agent

Creates:

* Twitter thread
* LinkedIn post
* Instagram caption

### 5.7 Editor Agent

Summarizes the blog into key bullet points for previews.

## **6. System Components and Feature Walkthrough**

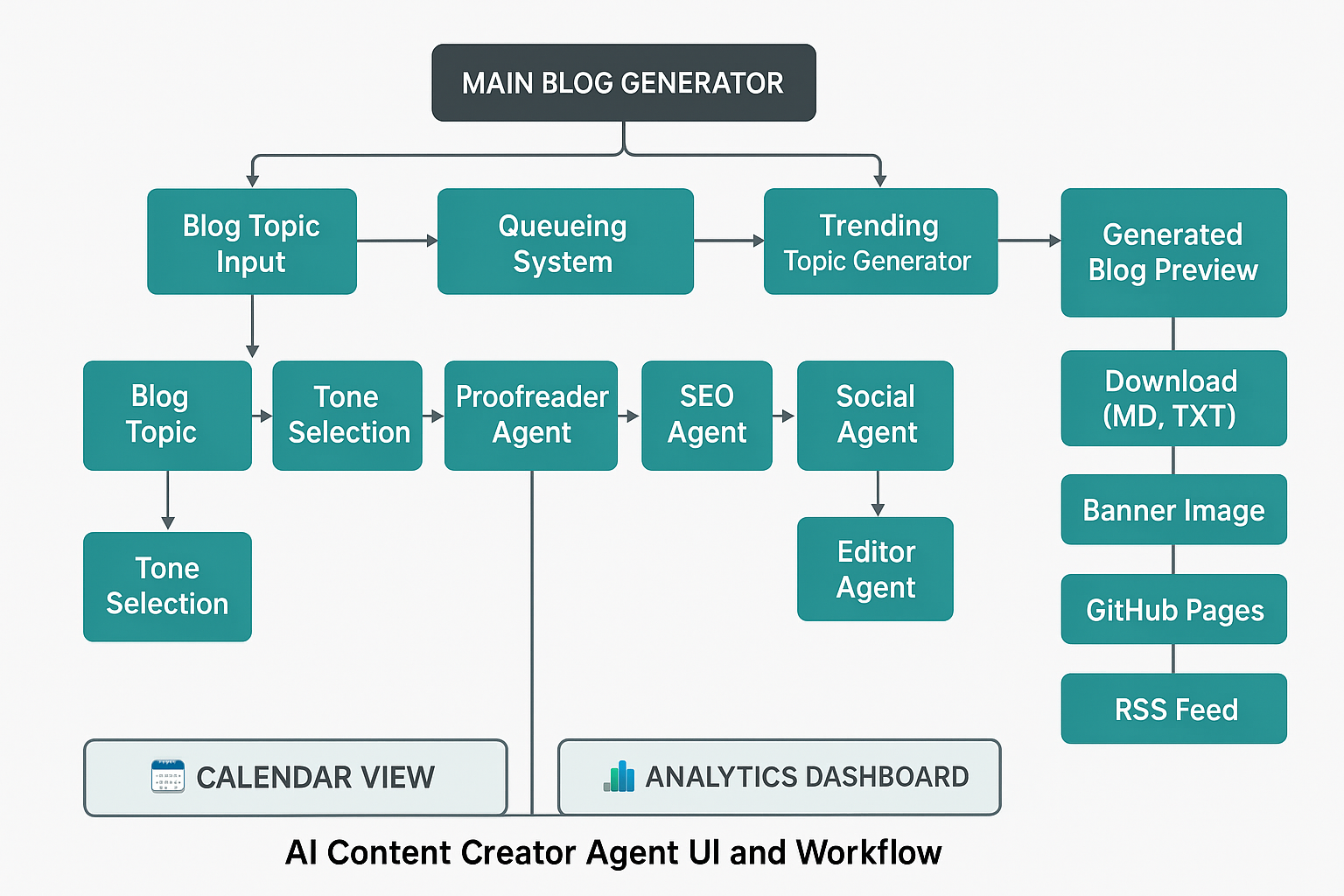


Figure 6. 1 Component Flow Diagram

The AI Content Creator Agent includes a user-friendly, modular interface powered by **Streamlit**, offering a seamless content generation workflow. Each component is designed to be intuitive, interactive, and optimized for productivity.

**🧭 Navigation & Tabs**

* **Main Blog Generator Tab**: Central hub for entering topics, selecting tone/audience, previewing outlines, and generating full content.
* **📆 Calendar View**: Visual heatmap showing daily publishing activity.
* **📊 Analytics Dashboard**: Summary of metrics, tag frequency, trends, and archive filters.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6. 2 Main Blog Generator Tab

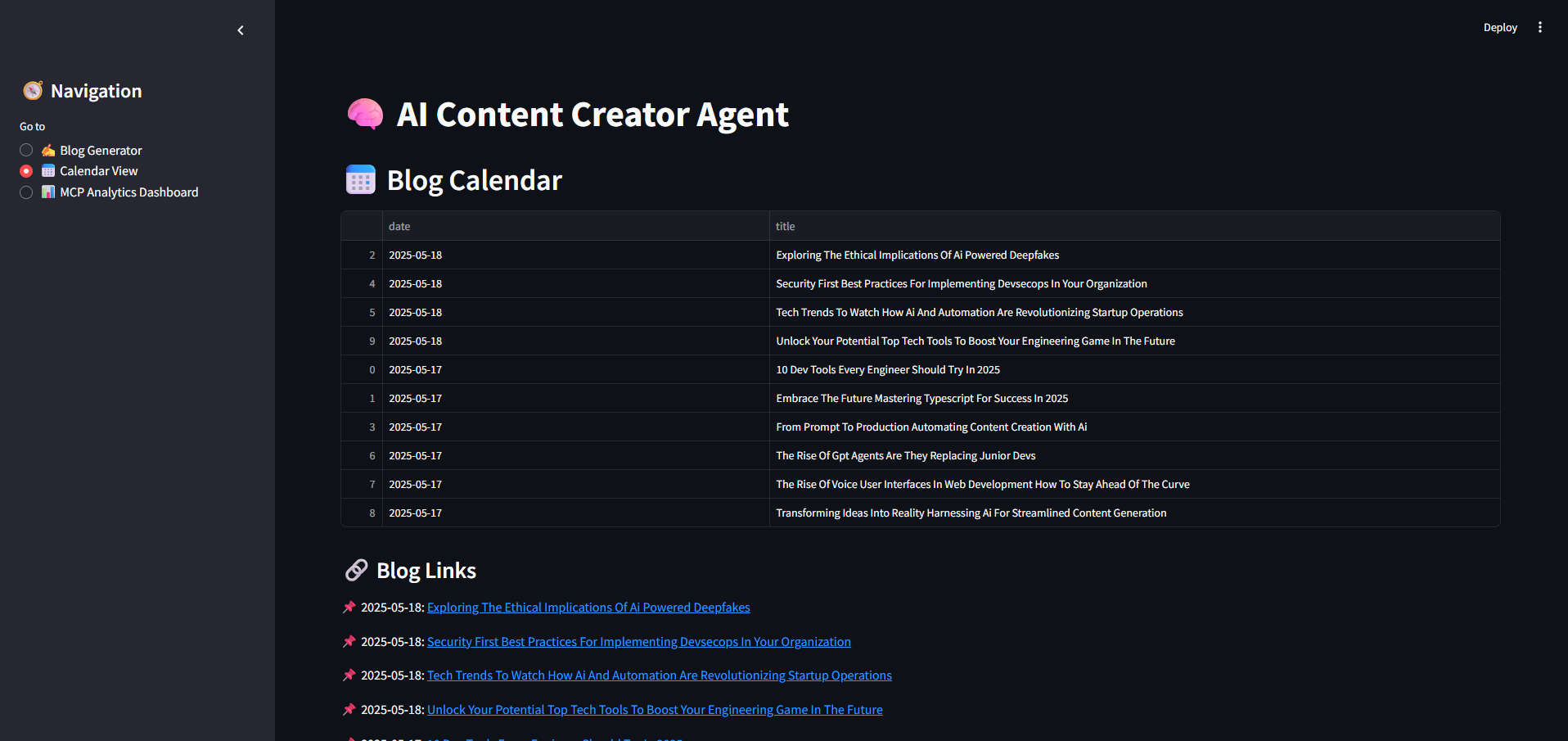


Figure 6. 3 Calender View

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6. 4 Analytics Dashboard

**✍️ Blog Generation Section**

* **Blog Topic Input**: Users can enter any custom blog idea or topic.
* **Tone Selection Dropdown**: Choose writing tone (e.g., professional, witty, inspirational).
* **Target Audience Dropdown**: Select the intended reader group (e.g., beginners, developers, CTOs).

**📚 AI-Generated Outline**

* **Outliner Agent Integration**: Suggests a structured blog outline before writing begins.
* **Editable Outline Box**: Users can refine the suggested structure before content generation.

**📈 Trending Topic Generator**

* **Category Selector**: Choose from tech, AI, DevOps, startups, etc.
* **Generate Suggestions**: Invokes AI to propose hot topics in the selected domain.
* **One-Click Selection**: Insert suggestions directly into the blog topic field.

**🚂 Queueing System**

* **Add to Queue**: Accumulate multiple blog topics for batch generation.
* **Generate Next**: Runs one queued topic through the content pipeline.
* **Run Full Queue**: Automatically processes all queued topics end-to-end.

**🔁 Automation Workflow:**

1. **Scheduled Execution**:
   * A Python-based scheduler (e.g., using schedule or cron) runs daily\_generate.py once a day.
   * A new topic is either auto-suggested or fetched from a queue.
2. **Content Generation**:
   * The full agent pipeline is triggered:
     + Writer → Proofreader → SEO → Captions → Editor → Citations
3. **GitHub Commit & Push**:
   * The new .html, .md, and rss.xml files are automatically committed and pushed to the GitHub repo.
   * This updates the GitHub Pages site in real time.
4. **Dev.to Auto-Publish**:
   * Since Dev.to monitors the RSS feed, it detects the new blog within minutes.
   * The blog appears as a draft on the Dev.to dashboard, ready for review/publishing.
5. **Email Notification**:
   * Once picked up by Dev.to, an email is sent confirming draft creation.

**🧠 AI Agent Pipeline**

Upon content generation trigger, the following sequence occurs:

1. **Writer Agent** – Drafts full content
2. **Proofreader Agent** – Refines language and tone
3. **Citation Inserter Agent** – Suggests references and links
4. **SEO Agent** – Adds meta description and keyword tags
5. **Social Agent** – Produces captions for Twitter, LinkedIn, Instagram
6. **Editor Agent** – Summarizes content into bullet points

**📤 Output & Publishing**

* **Generated Blog Preview**: Display of the final, polished post
* **Social Media Captions View**: Inline display of captions for each platform
* **Reading Time & Summary**: Auto-generated metrics shown next to blog
* **Share Banner Image**: Visual asset for social sharing
* **GitHub Pages Auto-Publishing**: Blog is saved as .html and hosted
* **RSS Feed Update**: rss.xml automatically refreshed for Dev.to pickup

**💾 Download Options**

* **Markdown (.md)** – Original blog text
* **Text (.txt)** – Social captions
* **JSON (.json)** – Metadata (SEO, citations, tags, summary)
* **PNG (.png)** – Blog-specific social banner
* **CSV (.csv)** – Full export of all blog metadata from the dashboard

## **7. Dashboard & Insights**

* Heatmap of blog publishing days
* Weekly/monthly generation trends
* Top SEO tags chart
* Tag-based filtering and keyword search
* Metadata export as CSV



Figure 7. 1 Weekly Monthly Blog Generation Trends Section

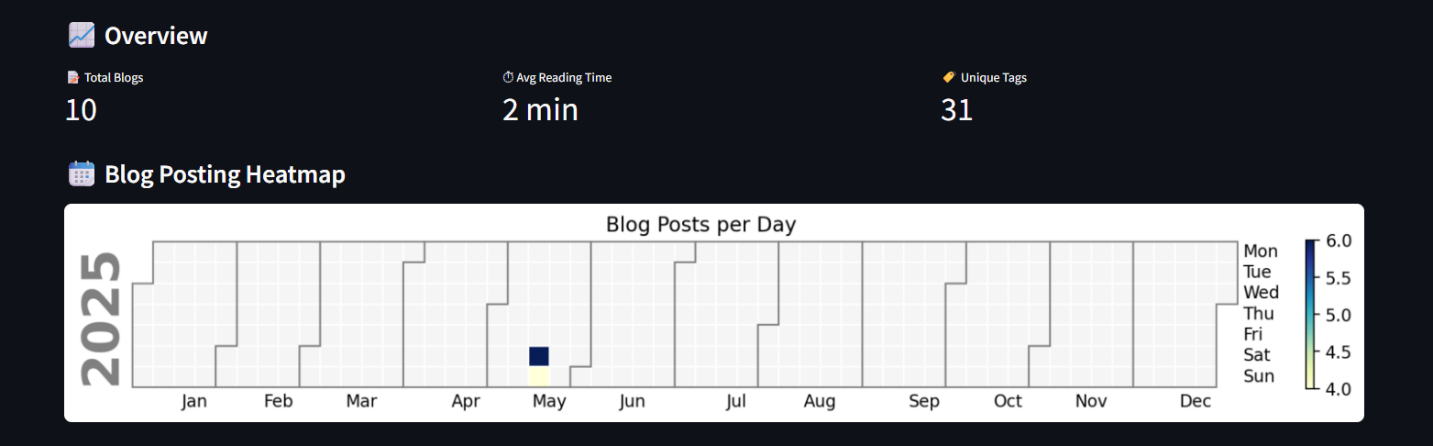


Figure 7. 2 Meta Data and Blog Posting Heatmap

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7. 3 Top SEO Tags Chart

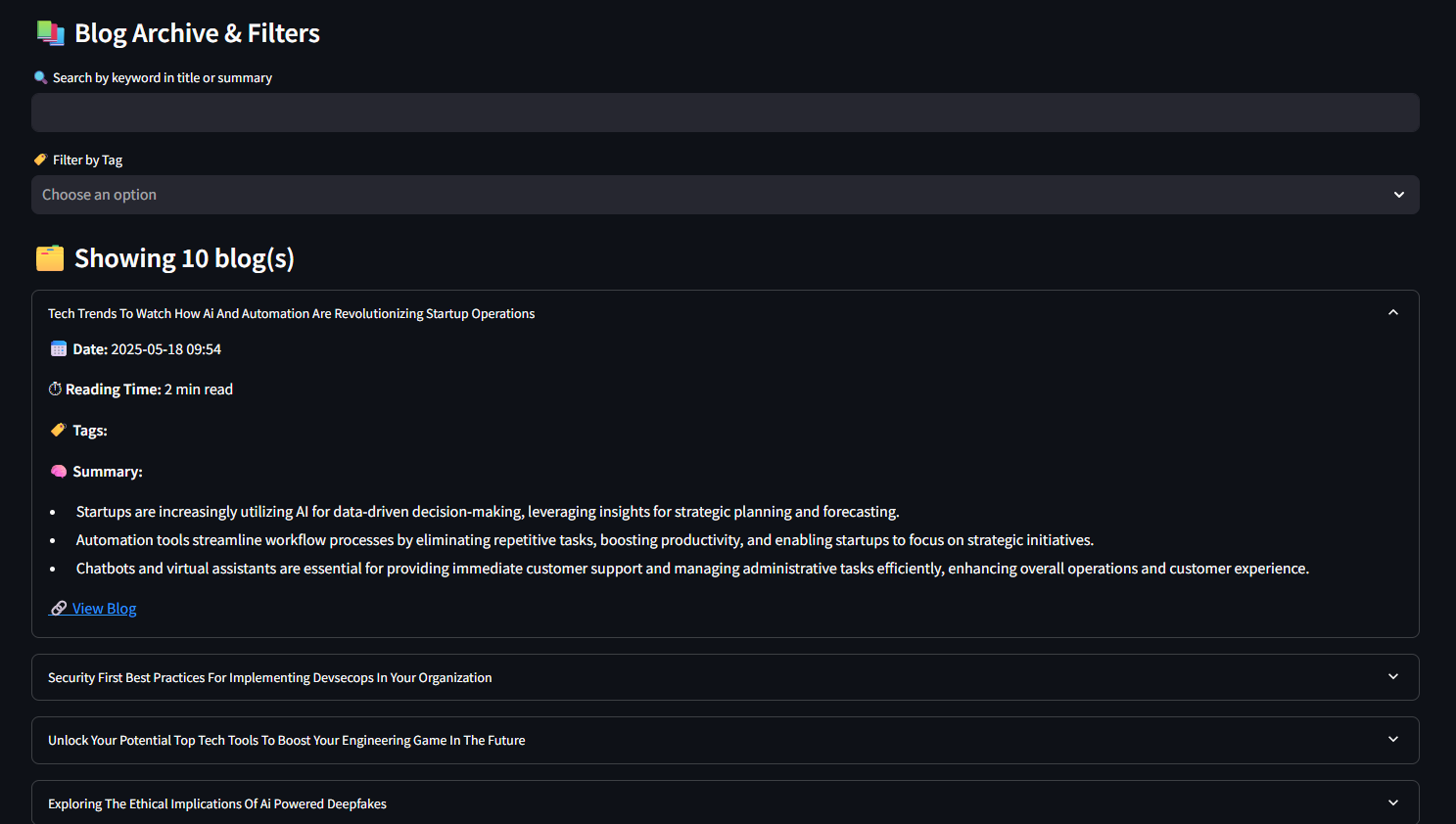


Figure 7. 4 Tag Based Filtering Keyword Search and Blog Archives

## **8. Output Structure**

├── blogs/ ➝ Markdown blog posts

├── captions/ ➝ Social media text files

├── metadata/ ➝ SEO, summary, citation data

├── docs/ ➝ GitHub Pages (HTML + rss.xml)

├── banners/ ➝ Auto-generated social share banners

## **9. Execution Flow**

1. Enter topic via UI
2. System checks for duplicates and suggests rewrites
3. Agents generate:
   * Outline
   * Blog
   * Captions
   * Metadata
   * Citations
   * Banner
4. Blog is saved and HTML is generated
5. RSS is updated, Dev.to detects new post
6. User can view/export content, analytics, or queue more

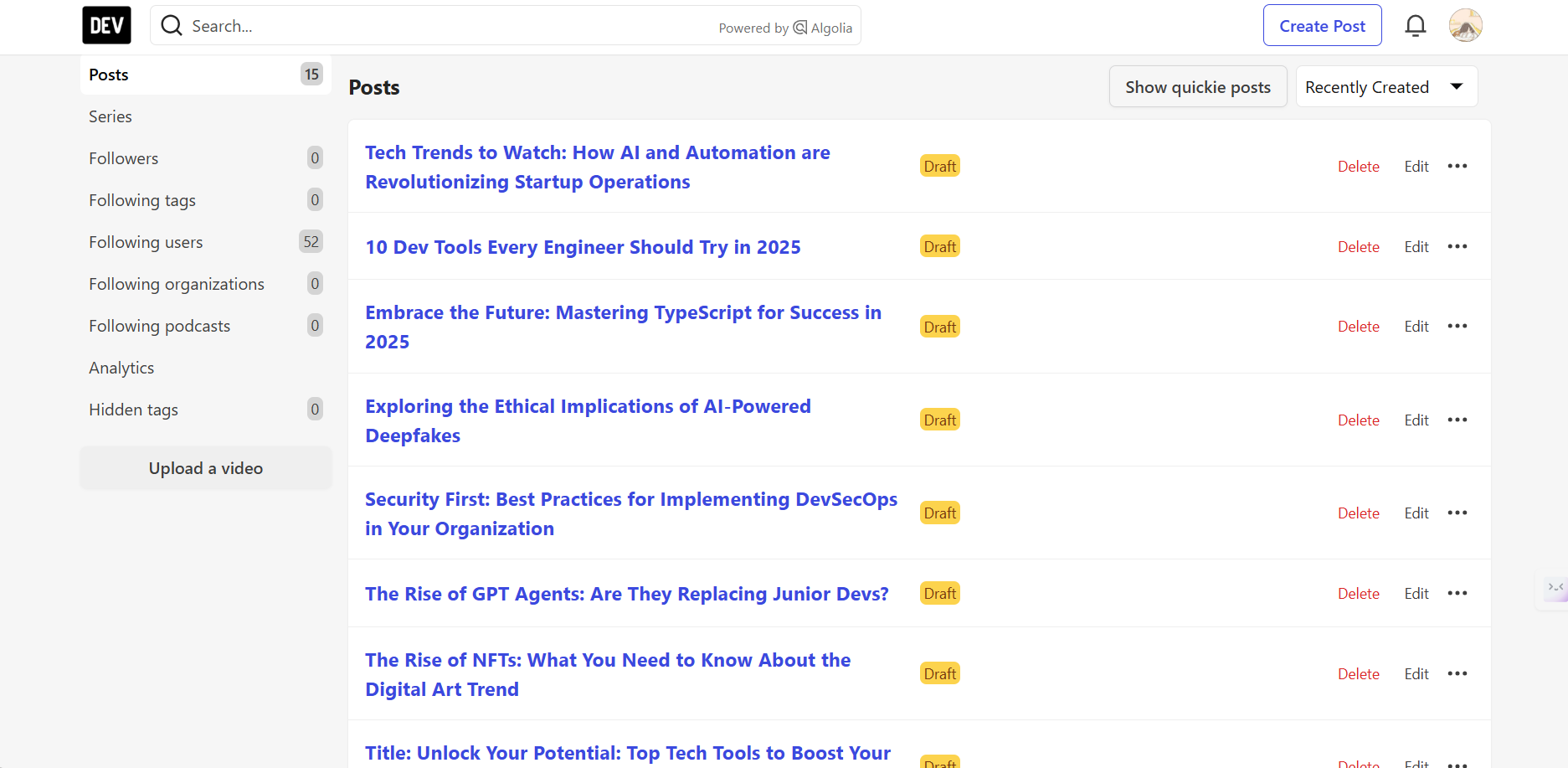


Figure 9. 1 New Posts Updated to Dev.to through RSS

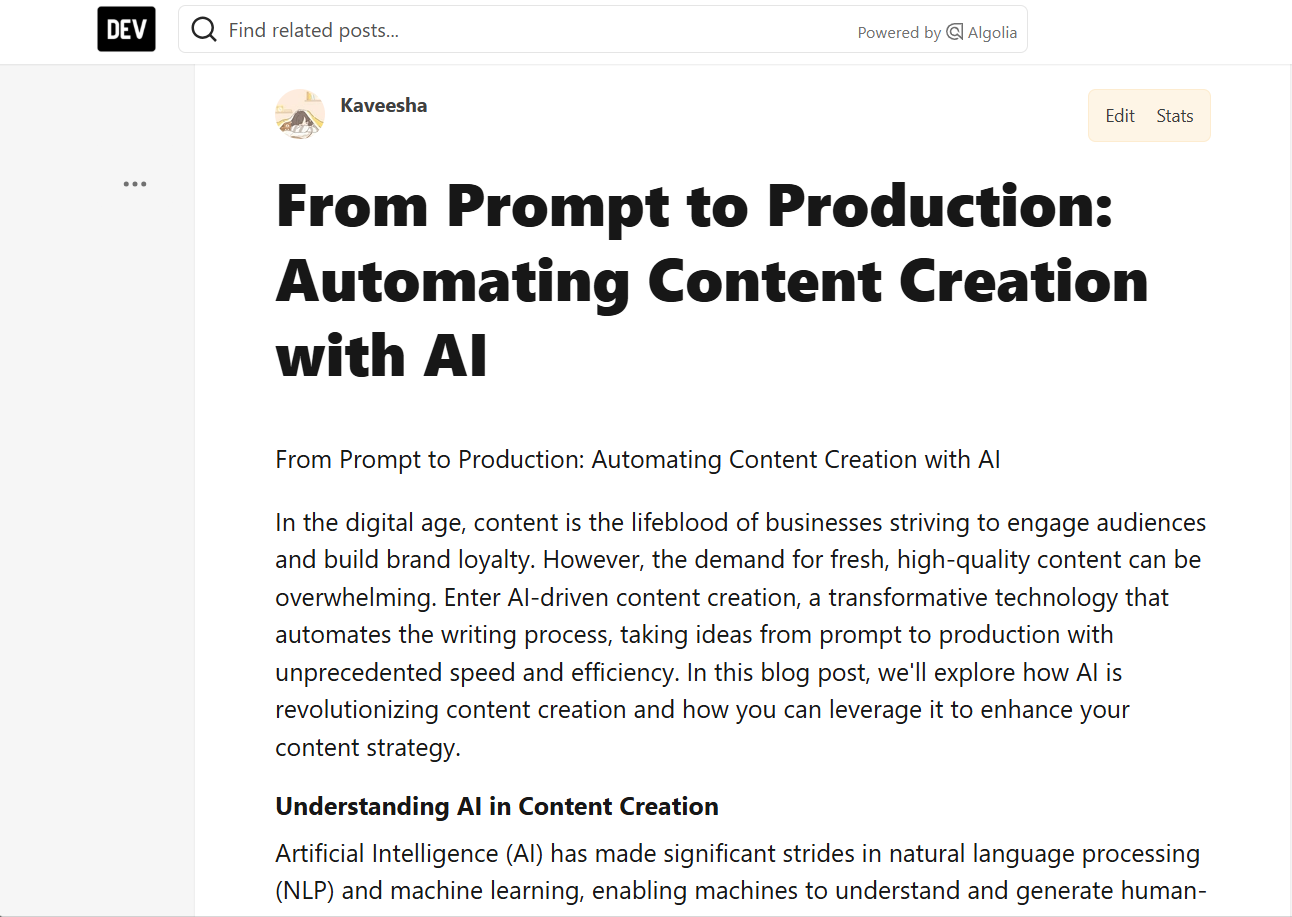


Figure 9. 2 AI Content Creator Generated Blog Posted on Dev.to

## **10. Future Enhancements**

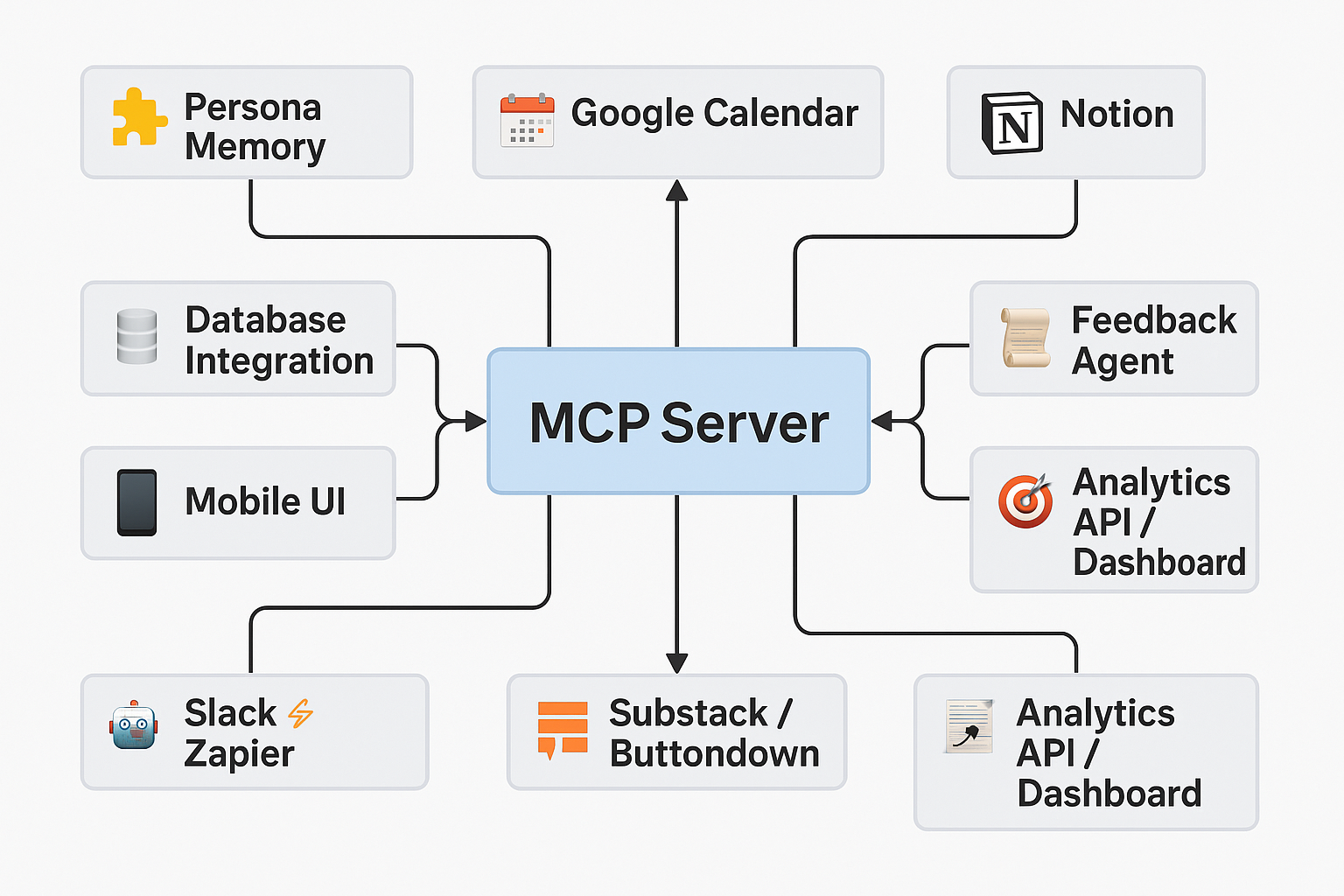


Figure 10. 1 Future Enhancements Overview Diagram

| **Feature** | **Description** |
| --- | --- |
| 🧩 **Persona Memory** | Enable memory for writing tone, author style, and preferred structure based on user profile. |
| 📱 **Mobile UI** | Responsive Streamlit layout or mobile companion app for on-the-go blog generation. |
| 🤖 **Slack/Zapier Integration** | Schedule or trigger blog generation via Slack commands or Zapier automations (e.g., generate a blog every Monday at 9AM). |
| 📰 **Newsletter Export (Substack/Buttondown)** | One-click export to email marketing platforms to send blog content as newsletters. |
| 🧠 **Agent Chaining & Parallel Planning** | Allow asynchronous/parallel execution of agents and planning of next content ideas based on trending searches. |
| 🗄️ **Database Integration** | Store all generated content, metadata, user profiles, and feedback in a database (PostgreSQL or Supabase). Enables querying, analytics, and persistent memory. |
| 📅 **Google Calendar Integration** | Auto-schedule blog publishing, reminders, and collaboration events tied to each blog topic. |
| 📬 **Gmail Integration** | Email generated blogs to editors, teams, or oneself. Could also accept blog ideas from starred emails. |
| 📓 **Notion Integration** | Sync blog content, drafts, and to-dos to a Notion workspace (content planning, scheduling, collaboration). |
| 🎯 **Analytics API / Webhook Dashboard** | Push performance data or publishing status to a dashboard or analytics tracker (e.g., Grafana, Superset). |
| 🧾 **Feedback Collection Agent** | Let users give feedback on generated content and tune models or prompt behavior over time. |

## **11. Conclusion**

The AI Content Creator Agent is a powerful example of how modern LLMs and agent-based orchestration can fully automate a creative pipeline. This MCP server design ensures modularity, extensibility, and end-to-end automation for content creators, marketers, and technical bloggers.

It transforms an idea into a digital asset—optimized, shareable, and ready to publish—without manual intervention.

## **12. Appendix**

* GitHub Repo: <https://github.com/kaveeshagim/ai-content-creator-agent>
* Live Blog Page: <https://kaveeshagim.github.io/ai-content-creator-agent>
* Dev.to Feed: [Kaveesha - DEV Community](https://dev.to/kaveesha_c74582728492e034)