# Agentic AI Assistant for Rednote Creative

## Introduction

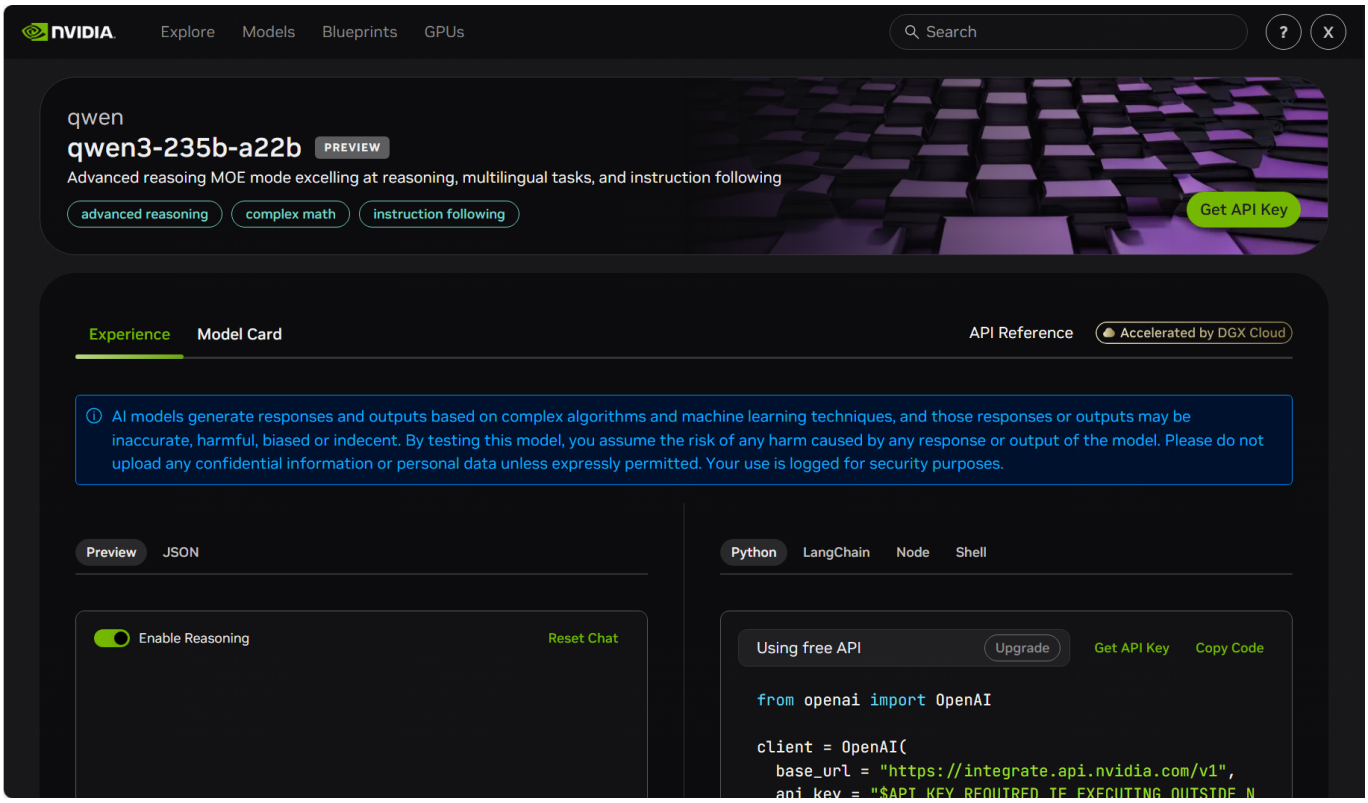
With NVIDIA AI Enterprise to accelerate Agentic AI Agent development, Rednote Creative Assistant is a powerful AI assistant designed to help you conduct deep research, generate comprehensive reports, and create engaging podcasts. It leverages NVIDIA NIM models and integrates seamlessly with Rednote's Model Context Protocol (MCP) services.

## Key Technologies

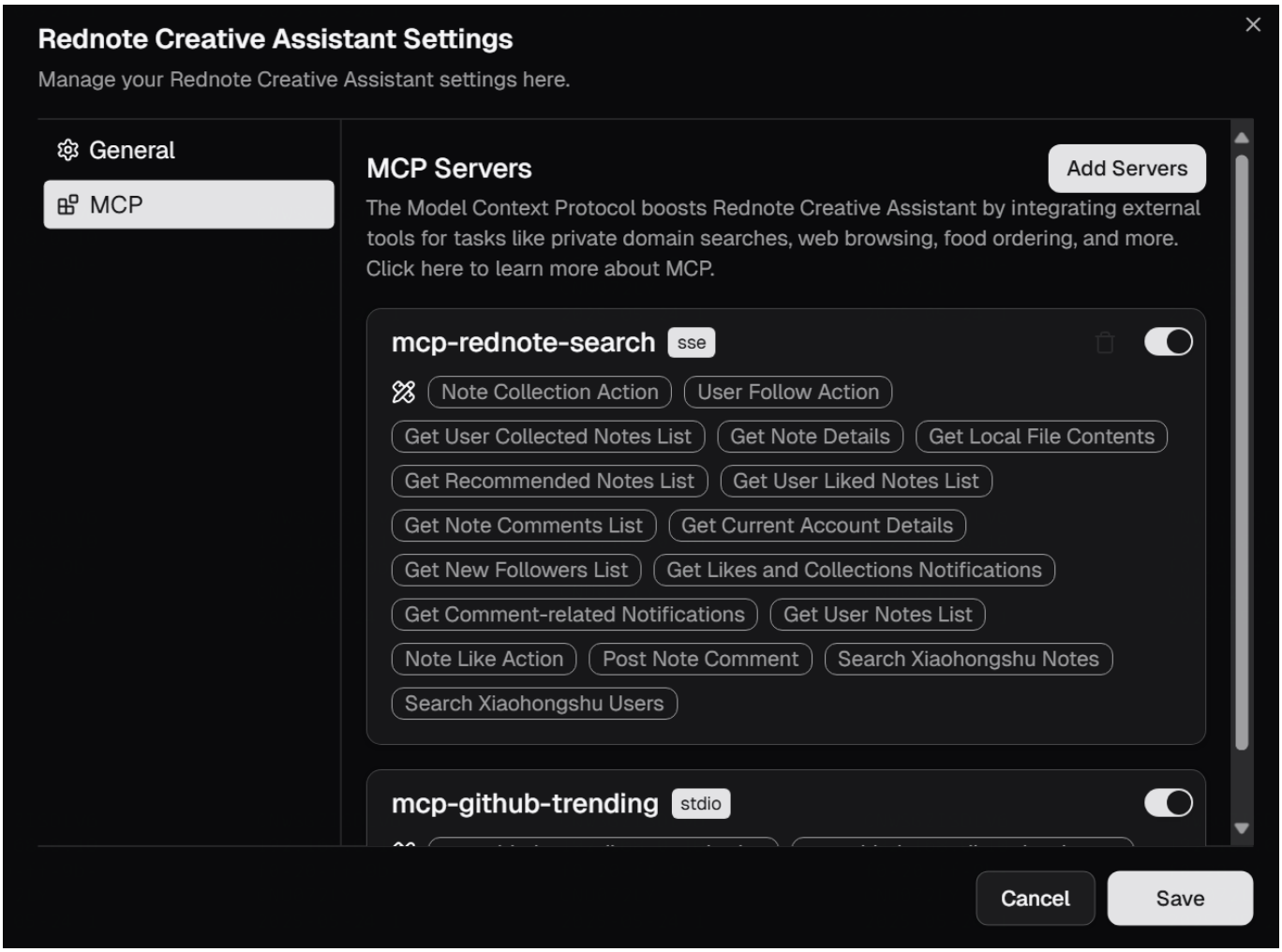
- **NVIDIA NIM Models**: Utilizes NVIDIA's advanced AI models for natural language understanding and generation, including the latest qwen3 models for basic and reasoning tasks, and microsoft latest multimodal model phi4 for vision tasks .
- **Multi-agent Collaboration(MAC)**: Implement multi-agent system with langgraph, Supports multiple specialized agents for complex tasks, including a Developer Agent, Researcher Agent, Podcast Agent, and Publisher Agent.
- **Rednote MCP**: Integrates with Rednote's Model Context Protocol to enhance capabilities for private domain access, knowledge graph, web browsing, and more.
- **Text-to-Speech (TTS)**: Converts research reports to speech using volcengine TTS API.

## Key LLMs & MCP Tools

- **Qwen3**: A powerful LLM model for basic and reasoning tasks, Please refer to the NVIDIA NIM Model card for more details (https://build.nvidia.com/qwen/qwen3-235b-a22b).
- **Phi4**: A multimodal model for vision tasks, capable of processing images and generating text based on visual content, Please refer to the NVIDIA NIM Model card for more details (https://build.nvidia.com/microsoft/phi-4-multimodal-instruct).
- **TTS**: A versatile LLM model for deep research and complex reasoning tasks.
- **Rednote MCP**: Integrates with Rednote's MCP services for search, publishing, and more.



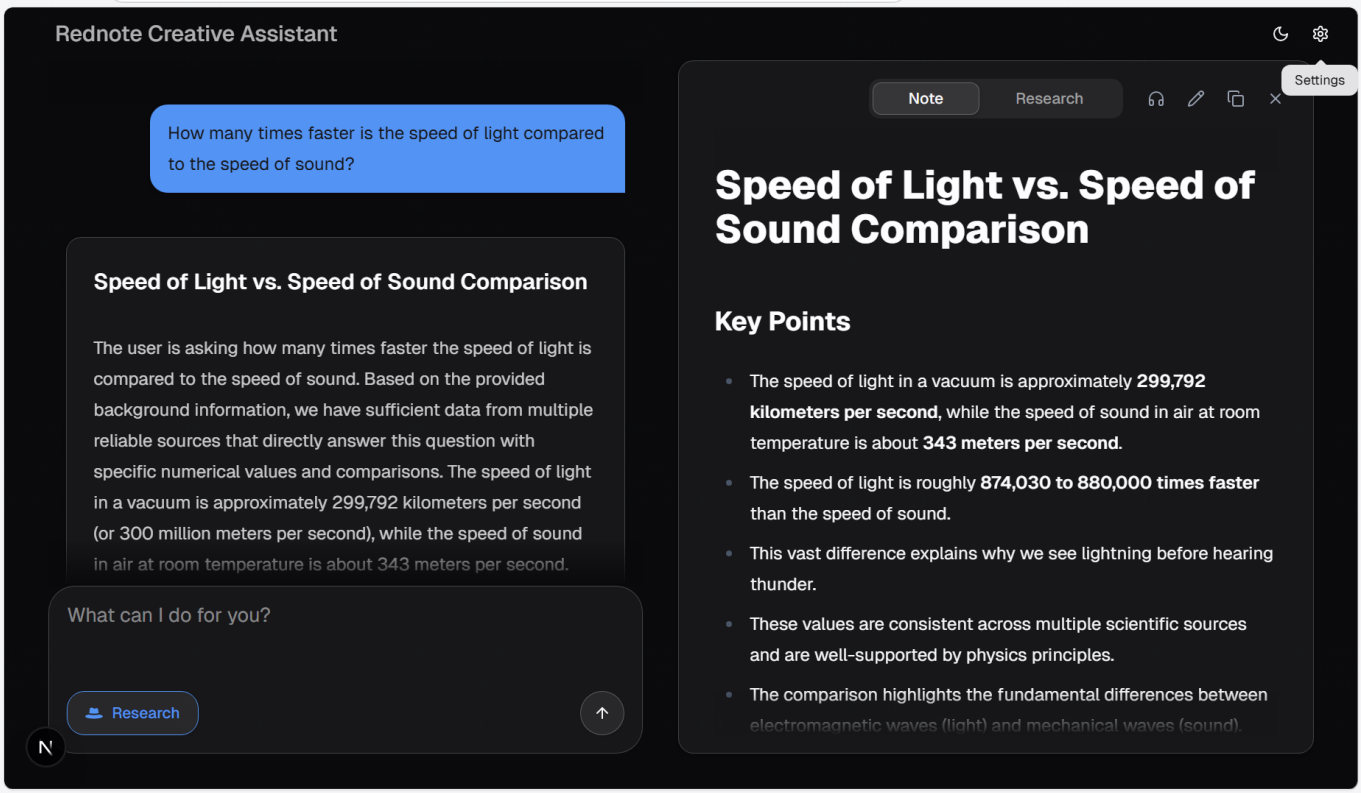

```yaml
BASIC_MODEL:
  base_url: https://integrate.api.nvidia.com/v1
  model: "qwen/qwen3-235b-a22b"
  api_key: xxx
  temperature: 0.2
  #max_tokens: 8192
  top_p: 0.7
REASONING_MODEL:
  base_url: https://integrate.api.nvidia.com/v1
  model: "qwen/qwen3-235b-a22b"
  api_key: xxx
  temperature: 0.2
  #max_tokens: 8192
  top_p: 0.7
VISION_MODEL:
  base_url: https://integrate.api.nvidia.com/v1
  model: "microsoft/phi-4-multimodal-instruct"
  api_key: xxx
  temperature: 0.1
  #max_tokens: 512
  top_p: 0.7
```




```
{
  "mcpServers":{
     "mcp-rednote-search": {
         "url": "http://127.0.0.1:19999/mcp",
         "transport": "sse",
         "add_to_agents": ["researcher"]
      }
  }
}
```

## Demo

This demo showcases the capabilities of Rednote Creative Assistant, including deep research, report generation, and podcast creation. The following images illustrate the user interface and features:




### Video

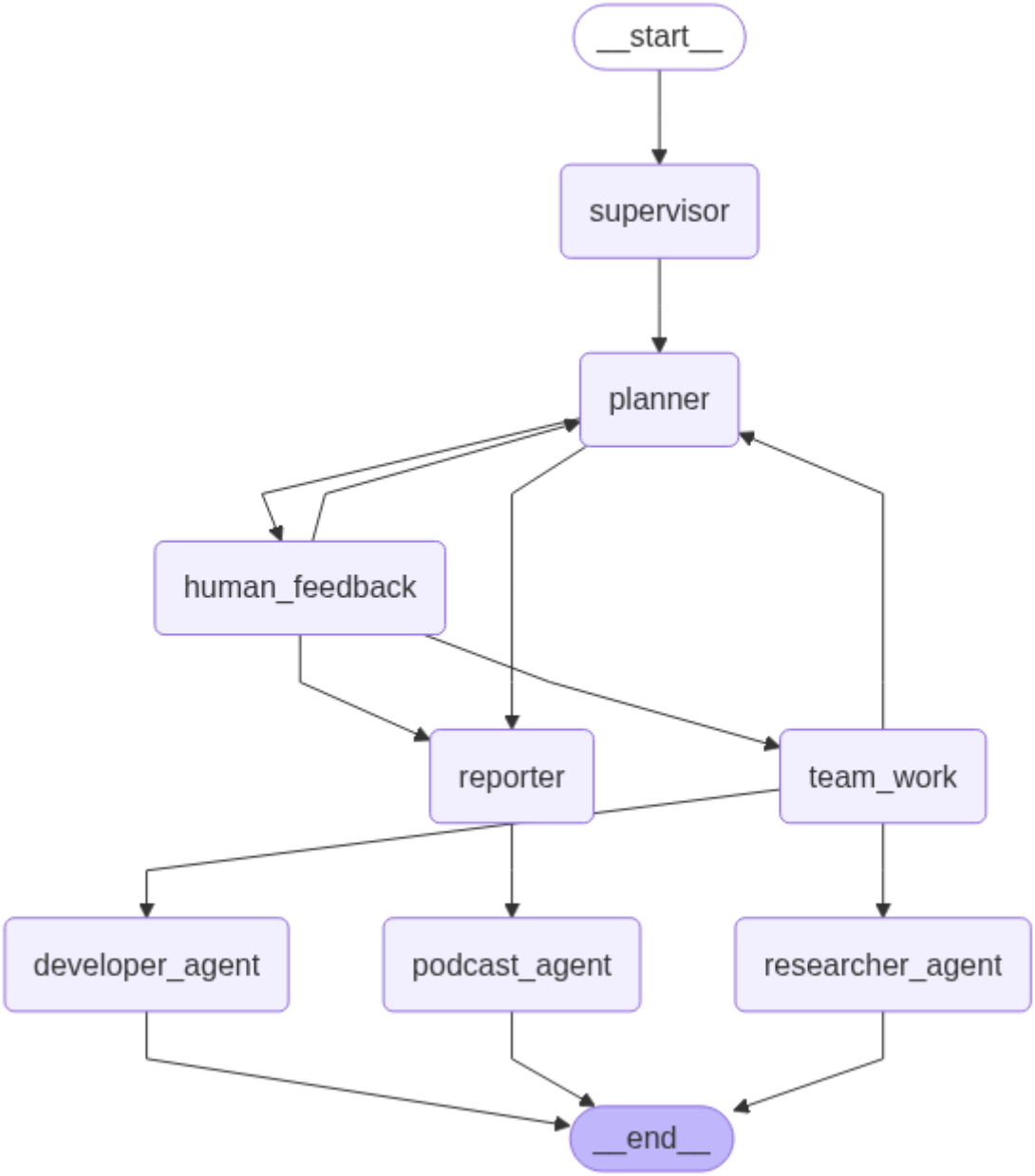
TODO

In this demo, we showcase how to use Agent to:

- Seamlessly integrate with MCP services
- Conduct the Deep Research process and produce a comprehensive report with images
- Create podcast audio based on the generated report

## Architecture

Rednote Creative Assistant implements a modular multi-agent system architecture designed for automated research and code analysis. The system is built on LangGraph, enabling a flexible state-based workflow where components communicate through a well-defined message passing system.




- Predefine workflow with LangGraph.
  - Multi-agent system for specialized complex tasks.
- Multi-Agent Collaboration(MAC)
  - Developer Agent - Write python scripts to generate charts, tables, and other data visualizations. (Ready)
  - Researcher Agent - Conduct web searches and gather information. (Ready)
  - Podcast Agent - Generate podcast audio based on the generated report. (Ready)
  - Publisher Agent - Generate reports and presentations and publish them to Rednote. (TODO)
- MCP Integration
  - mcp-rednote-search - Search Rednote contents. (Ready)
  - mcp-rednote-publish - Publish reports and presentations to Rednote (TODO)

The system employs a streamlined workflow with the following components:

1. **Supervisor**: The entry point that manages the workflow lifecycle
   - Initiates the research process based on user input
   - Delegates tasks to the planner when appropriate

- Acts as the primary interface between the user and the system

2. **Planner**: Strategic component for task decomposition and planning
   - Analyzes research objectives and creates structured execution plans
   - Determines if enough context is available or if more research is needed
   - Manages the research flow and decides when to generate the final report
3. **Team Work**: A collection of specialized agents that execute the plan:
   - **Researcher**: Conducts web searches and information gathering using tools like web search engines, crawling and even MCP services.
   - **Developer**: Handles code analysis, execution, and technical tasks using Python REPL tool. Each agent has access to specific tools optimized for their role and operates within the LangGraph framework
4. **Reporter**: Final stage processor for research outputs
   - Aggregates findings from the research team
   - Processes and structures the collected information
   - Generates comprehensive research reports

## Quick Start

Rednote Creative Assistant is developed in Python, and comes with a web UI written in Node.js. To ensure a smooth setup process, we recommend using the following tools:

### Recommended Tools

- `uv`**:** Simplify Python environment and dependency management. `uv` automatically creates a virtual environment in the root directory and installs all required packages for you—no need to manually install Python environments.
- `nvm`**:** Manage multiple versions of the Node.js runtime effortlessly.
- `pnpm`**:** Install and manage dependencies of Node.js project.

### Environment Requirements

Make sure your system meets the following minimum requirements:

- **Python:** Version `3.12+`
- **Node.js:** Version `22+`

### Installation

```
# Clone the repository
git clone https://github.com/changxubo/rednote-creative-aisstant.git

# Install dependencies, uv will take care of the python interpreter and venv
creation, and install the required packages
```

```
uv sync

# Configure .env with your API keys
# Tavily: https://app.tavily.com/home
# TTS: Add your TTS credentials if you have them
cp .env.example .env

# See the 'Supported Search Engines' and 'Text-to-Speech Integration' sections
below for all available options

# Configure conf.yaml for your LLM model and API keys
# Please refer to 'docs/configuration_guide.md' for more details
cp conf.yaml.example conf.yaml

# Install marp for ppt generation
# https://github.com/marp-team/marp-cli?tab=readme-ov-file#use-package-manager
brew install marp-cli
```

Optionally, install web UI dependencies via pnpm:

```
cd web
pnpm install
```

## Configurations

Please refer to the Configuration Guide for more details.

> [!NOTE] Before you start the project, read the guide carefully, and update the configurations to match your specific settings and requirements.

## Console UI

The quickest way to run the project is to use the console UI.

```
# Run the project in a bash-like shell
uv run main.py
```

## Web UI

This project also includes a Web UI, offering a more dynamic and engaging interactive experience.

> [!NOTE] You need to install the dependencies of web UI first.

```
# Run both the backend and frontend servers in development mode
# On macOS/Linux
./bootstrap.sh -d
```

```
# On Windows
bootstrap.bat -d
```

Open your browser and visit `http://localhost:3000` to explore the web UI.

Explore more details in the `web` directory.

Web UI startup log:

```
> next dev --turbo

   ▲ Next.js 15.3.0 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.255.227:3000

 ✓ Starting...
 ✓ Ready in 3.3s
```

Agent server startup log:

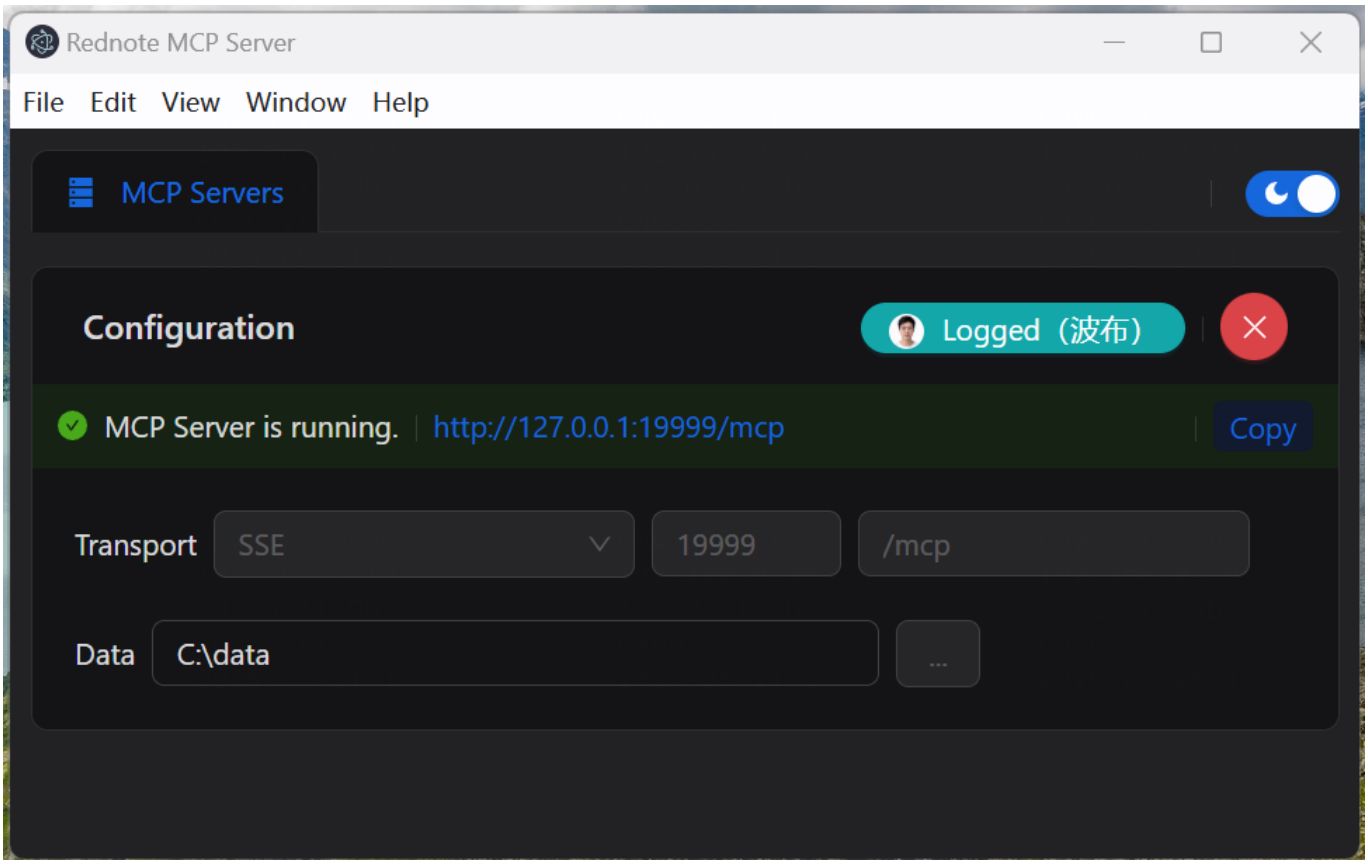
```
2025-05-24 17:00:48,614 - __main__ - INFO - Starting Agent API server on
localhost:8000
INFO:     Uvicorn running on http://localhost:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [49828] using StatReload
INFO:     Started server process [50456]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

## Rednote MCP Servers

Rednote Creative Assistant integrates with Rednote MCP servers to enhance its capabilities. These servers provide access to various services, including search, publishing, and more.

```
$ cd rednote
$ npm i --legacy-peer-deps
```

```
$ npm run build:mcp
$ npm run dev
```




### Add MCP Servers

To add MCP servers, you can use the `mcp-rednote-search` and `mcp-rednote-publish` services. These services allow you to search Rednote contents and publish reports and presentations to Rednote.

```
{
  "mcpServers":{
     "mcp-rednote-search": {
         "url": "http://127.0.0.1:19999/mcp",
         "transport": "sse",
         "add_to_agents": ["researcher"]
      }
  }
}
```

## Features

### Core Capabilities

- **LLM Integration**
  - It supports the integration of most models through litellm.
  - Support for open source models like Qwen
  - OpenAI-compatible API interface
  - Multi-tier LLM system for different task complexities

### Tools and MCP Integrations

- 🔍 **Search and Retrieval**
  - Web search via Tavily, Brave Search and more
  - Crawling with Jina
  - Advanced content extraction
- 🔗 **MCP Seamless Integration**
  - Expand capabilities for private domain access, knowledge graph, web browsing and more
  - Facilitates integration of diverse research tools and methodologies

### Human Collaboration

- 🧠 **Human-in-the-loop**
  - Supports interactive modification of research plans using natural language
  - Supports auto-acceptance of research plans
- 📝 **Report Post-Editing**
  - Supports Notion-like block editing
  - Allows AI refinements, including AI-assisted polishing, sentence shortening, and expansion
  - Powered by tiptap

### Content Creation

- 🎙️ **Podcast and Presentation Generation**
  - AI-powered podcast script generation and audio synthesis
  - Automated creation of simple PowerPoint presentations
  - Customizable templates for tailored content

## Text-to-Speech Integration

Rednote Creative Assistant now includes a Text-to-Speech (TTS) feature that allows you to convert research reports to speech. This feature uses the volcengine TTS API to generate high-quality audio from text. Features like speed, volume, and pitch are also customizable.

### Using the TTS API

You can access the TTS functionality through the `/api/tts` endpoint:

```
# Example API call using curl
curl --location 'http://localhost:8000/api/tts' \
--header 'Content-Type: application/json' \
--data '{
    "text": "This is a test of the text-to-speech functionality.",
    "speed_ratio": 1.0,
    "volume_ratio": 1.0,
    "pitch_ratio": 1.0
}' \
--output speech.mp3
```

# Development

## Testing

Run the test suite:

```
# Run all tests
make test

# Run specific test file
pytest tests/integration/test_workflow.py

# Run with coverage
make coverage
```

## Code Quality

```
# Run linting
make lint

# Format code
make format
```

## Debugging with LangGraph Studio

Rednote Creative Assistant uses LangGraph for its workflow architecture. You can use LangGraph Studio to debug and visualize the workflow in real-time.

### Running LangGraph Studio Locally

Rednote Creative Assistant includes a `langgraph.json` configuration file that defines the graph structure and dependencies for the LangGraph Studio. This file points to the workflow graphs defined in the project and automatically loads environment variables from the `.env` file.

#### Mac

```
# Install uv package manager if you don't have it
curl -LsSf https://astral.sh/uv/install.sh | sh

# Install dependencies and start the LangGraph server
uvx --refresh --from "langgraph-cli[inmem]" --with-editable . --python 3.12
langgraph dev --allow-blocking
```

**Windows / Linux**

```
# Install dependencies
pip install -e .
pip install -U "langgraph-cli[inmem]"

# Start the LangGraph server
langgraph dev --allow-blocking
```

After starting the LangGraph server, you'll see several URLs in the terminal:

- API: http://127.0.0.1:2024
- Studio UI: https://smith.langchain.com/studio/?baseUrl=http://127.0.0.1:2024
- API Docs: http://127.0.0.1:2024/docs

Open the Studio UI link in your browser to access the debugging interface.

**Using LangGraph Studio**

In the Studio UI, you can:

1. Visualize the workflow graph and see how components connect
2. Trace execution in real-time to see how data flows through the system
3. Inspect the state at each step of the workflow
4. Debug issues by examining inputs and outputs of each component
5. Provide feedback during the planning phase to refine research plans

When you submit a research topic in the Studio UI, you'll be able to see the entire workflow execution, including:

- The planning phase where the research plan is created
- The feedback loop where you can modify the plan
- The research and writing phases for each section
- The final report generation

## Enabling LangSmith Tracing

Rednote Creative Assistant supports LangSmith tracing to help you debug and monitor your workflows. To enable LangSmith tracing:

1. Make sure your `.env` file has the following configurations (see `.env.example`):

```
LANGSMITH_TRACING=true
LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
LANGSMITH_API_KEY="xxx"
LANGSMITH_PROJECT="xxx"
```

2. Start tracing and visualize the graph locally with LangSmith by running:

```
langgraph dev
```

This will enable trace visualization in LangGraph Studio and send your traces to LangSmith for monitoring and analysis.

### Human in the Loop

Rednote Creative Assistant includes a human in the loop mechanism that allows you to review, edit, and approve research plans before they are executed:

1. **Plan Review**: When human in the loop is enabled, the system will present the generated research plan for your review before execution
2. **Providing Feedback**: You can:
   - Accept the plan by responding with `[ACCEPTED]`
   - Edit the plan by providing feedback (e.g., `[EDIT PLAN] Add more steps about technical implementation`)
   - The system will incorporate your feedback and generate a revised plan
3. **Auto-acceptance**: You can enable auto-acceptance to skip the review process:
   - Via API: Set `auto_accepted_plan: true` in your request
4. **API Integration**: When using the API, you can provide feedback through the `feedback` parameter:

```
{
  "messages": [{ "role": "user", "content": "What is quantum computing?" }],
  "thread_id": "my_thread_id",
  "auto_accepted_plan": false,
  "feedback": "[EDIT PLAN] Include more about quantum algorithms"
}
```

## FAQ

Please refer to the FAQ.md for more details.

## License

This project is open source and available under the MIT License.