TASK-6

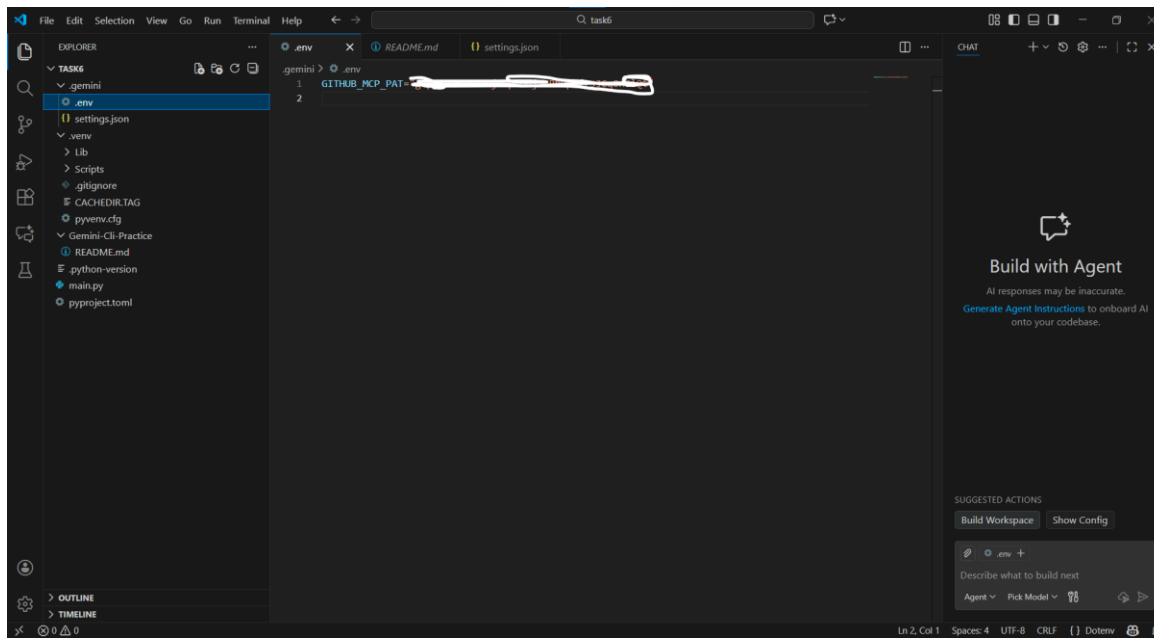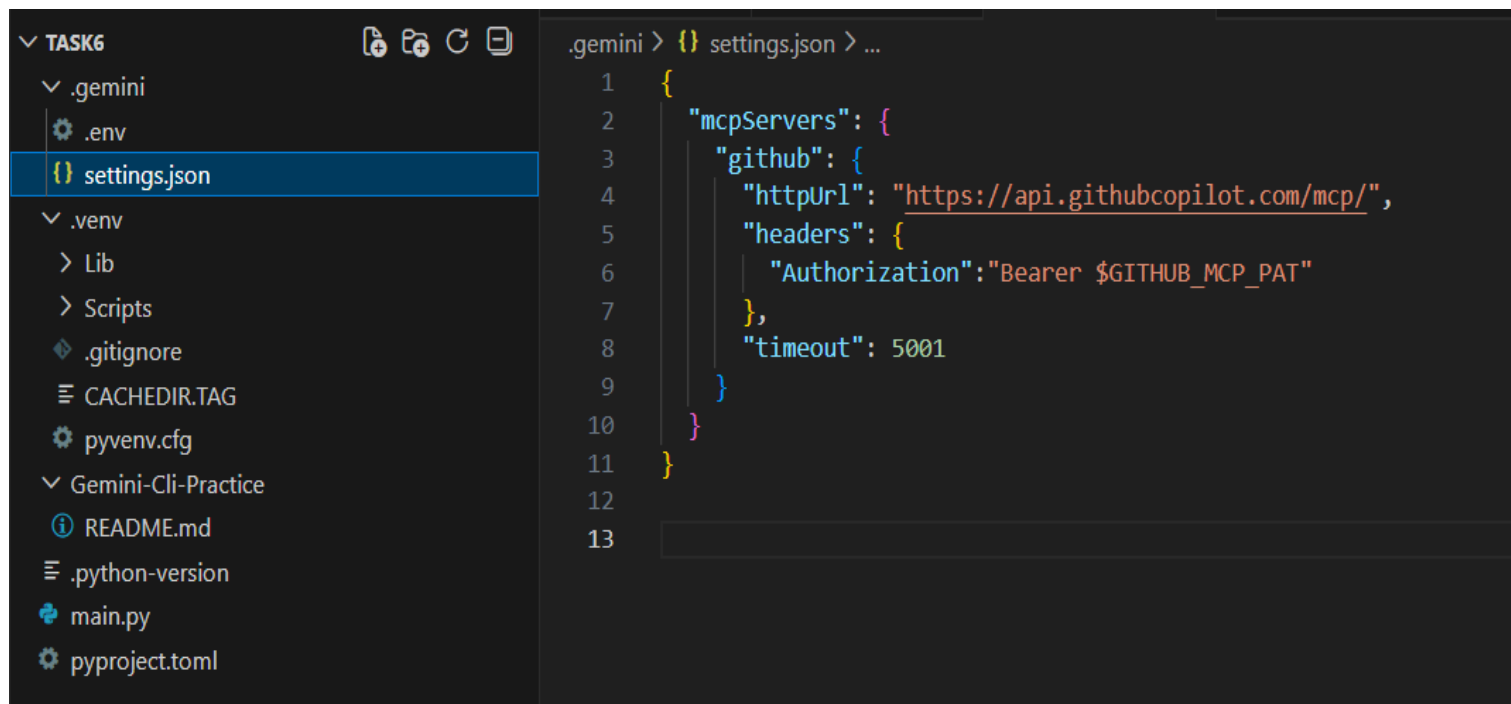**GITHUB MCP INTEGRATION**

1.   CREATE PERSONAL ACCESS TOKEN (PAT) THROUGH GITHUB AND SAVE IN .env file

2. CONFIGURE GEMINI TO GITHUB MCP SERVER

```
.gemini > {} settings.json > ...
  1  {
  2    "mcpServers": {
  3      "github": {
  4        "httpUrl": "https://api.githubcopilot.com/mcp/",
  5        "headers": {
  6          "Authorization":"Bearer $GITHUB_MCP_PAT"
  7        },
  8        "timeout": 5001
  9      }
 10    }
 11  }
 12
 13
```

File explorer:
- TASK6
  - .gemini
    - .env
    - settings.json
  - .venv
    - Lib
    - Scripts
    - .gitignore
    - CACHEDIR.TAG
    - pyvenv.cfg
  - Gemini-Cli-Practice
    - README.md
  - .python-version
  - main.py
  - pyproject.toml

3.   MCP SERVER CONNECTED WITH GITHUB SUCCESSFULLY.

```
- issue_read
- issue_write
- list_branches
- list_commits
- list_issue_types
- list_issues
- list_pull_requests
- list_releases
- list_tags
- merge_pull_request
- pull_request_read
- pull_request_review_write
- push_files
- request_copilot_review
- search_code
- search_issues
- search_pull_requests
- search_repositories
- search_users
- sub_issue_write
- update_pull_request
- update_pull_request_branch
Prompts:
- AssignCodingAgent
- IssueToFixWorkflow


Using: 1 MCP server
```

GEMINI CREATED FOLDER AND MOVE IT INTO GITHUB REPOSITORY

```
    Directory: C:\Users\kk\Desktop\Darakshan\quater4\task6


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        27-Nov-2025    4:04 PM              Gemini-Cli-Practice


  ▣  Shell cd Gemini-Cli-Practice [current working directory C:\Users\kk\Desktop\Darakshan\quater4\task


▣ I've created the Gemini-Cli-Practice folder and moved into it. What's next?

Using: 1 MCP server

 > █ Type your message or @path/to/file
```
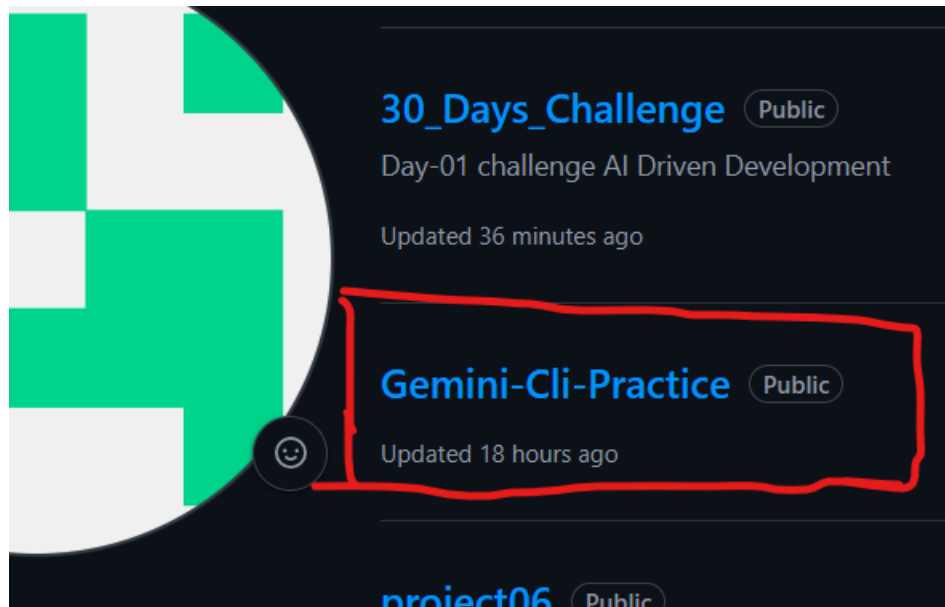
30_Days_Challenge Public

Day-01 challenge AI Driven Development

Updated 36 minutes ago

Gemini-Cli-Practice Public

Updated 18 hours ago

project06 Public

About
This project aims to streamline the process of interacting with and extracting information from Markdown-based documentation using a powerful MCP server backend and the intelligent capabilities of Google's Gemini models. It's built with extensibility and performance in mind, offering a foundational component for AI-assisted development workflows.

Resources
-   [GitHub Repository (Placeholder)](https://github.com/owner/repo) - Replace with actual repository link
-   [Gemini API Documentation (External)](https://ai.google.dev/docs)
-   [Model Context Protocol (MCP) Specification (External)](https://github.com/google/model-context-protocol)

GitHub Metrics (Placeholder)
![GitHub Stars](https://img.shields.io/github/stars/owner/repo.svg?style=social)
![GitHub Watchers](https://img.shields.io/github/watchers/owner/repo.svg?style=social)
![GitHub Forks](https://img.shields.io/github/forks/owner/repo.svg?style=social)
![GitHub Releases](https://img.shields.io/github/downloads/owner/repo/total.svg)
![GitHub Packages](https://img.shields.io/github/packages/owner/repo/total.svg)

Footer
© 2025 Your Name or Organization. All rights reserved.
Generated by Gemini CLI.

README.md is done. Task complete.

▶ ReadFolder .

Listed 7 item(s).

GitHub MCP Server and Gemini CLI

Introduction
This project focuses on providing a robust Model Context Protocol (MCP) server that integrates seamlessly with the Gemini CLI. It processes Markdown content, parsing and transforming it, centralizing the interface for Markdown file handling. The core functionality involves ingesting Markdown files and preparing their content for further manipulation or conversion, ultimately enabling advanced interactions through the Gemini CLI. This solution bridges the gap between structured documentation and AI-driven processing, offering a powerful tool for developers and content creators.

Features
- **Markdown Processing:** Efficiently parses and transforms Markdown content into various formats, including HTML.
- **Configurable Markdown Rendering:** Supports basic and advanced Markdown rendering using libraries like `marked` and `Prism` for syntax highlighting.
- **Content Extraction:** Extracts key information from Markdown such as titles, descriptions, example snippets, and schema arguments.
- **MCP Server Integration:** Acts as an MCP server, exposing tools for interaction.
- **Gemini Model Integration:** Leverages Gemini models for advanced AI capabilities, including content generation and understanding.
- **Schema Conversion:** Converts OpenAPI schemas to Gemini Schema objects for consistent tool definitions.
- **README Content Management:** Fetches and parses `README.md` files from GitHub repositories, extracting structured data.

Gemini CLI Integration
The project is designed to be a backend service for the Gemini CLI, enabling it to interact with and process various types of content, especially Markdown documentation.
- **Model Interaction:** Utilizes `Gemini` and `GeminiModel` classes to interact with different Gemini models (`gemini-1.5-flash`, `gemini-2.0-flash-001`, etc.) for content generation and understanding.
- **Tool Definition:** Translates tool schemas into Gemini-compatible formats using `gemini_to_json_schema` for effective tool calling within the Gemini ecosystem.
- **Asynchronous Operations:** Supports asynchronous content generation and real-time streaming with Gemini models.

About