

Software Engineering

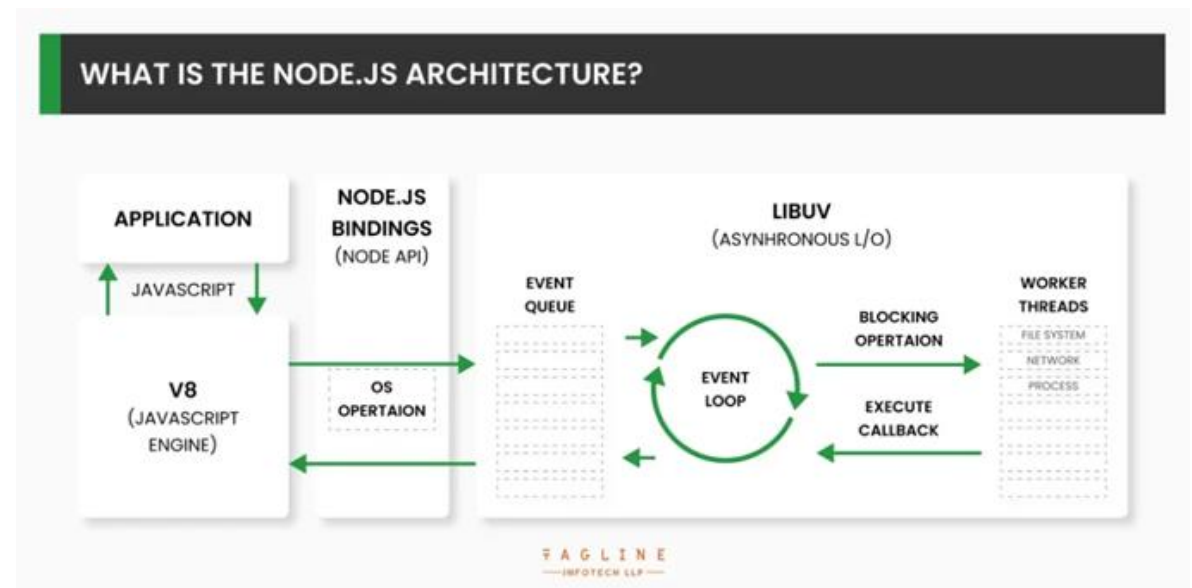
Hands-On Lecture – 02

Software Engineering in the era of AI

Install Gemini CLI & Python 3.12.10

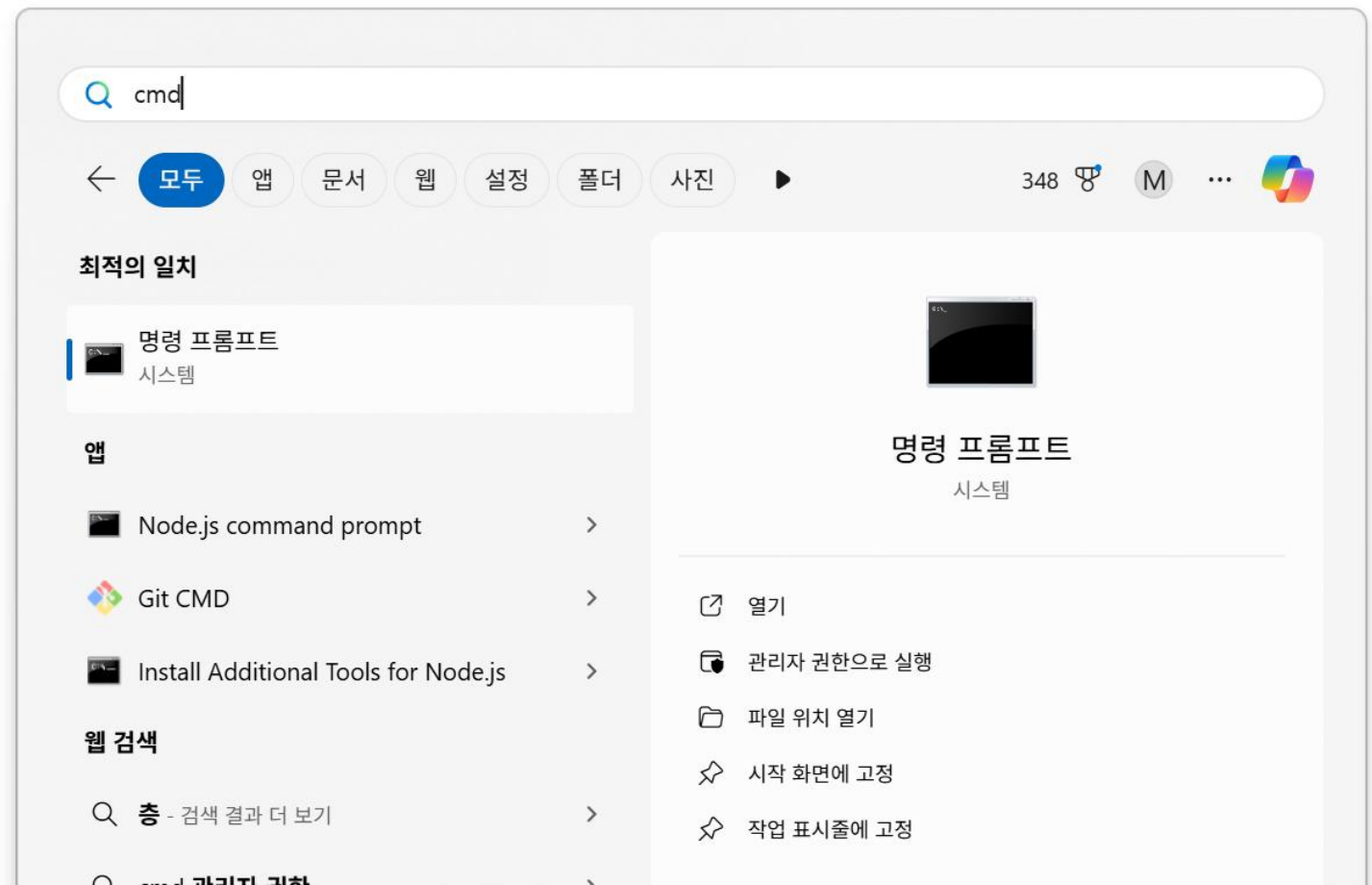
Install Gemini CLI

- [Node.js — Node.js® 다운로드](#)
- JavaScript 런타임 환경 (AI 서비스가 터미널에서 File System 접근을 위해 사용)



Install Gemini CLI

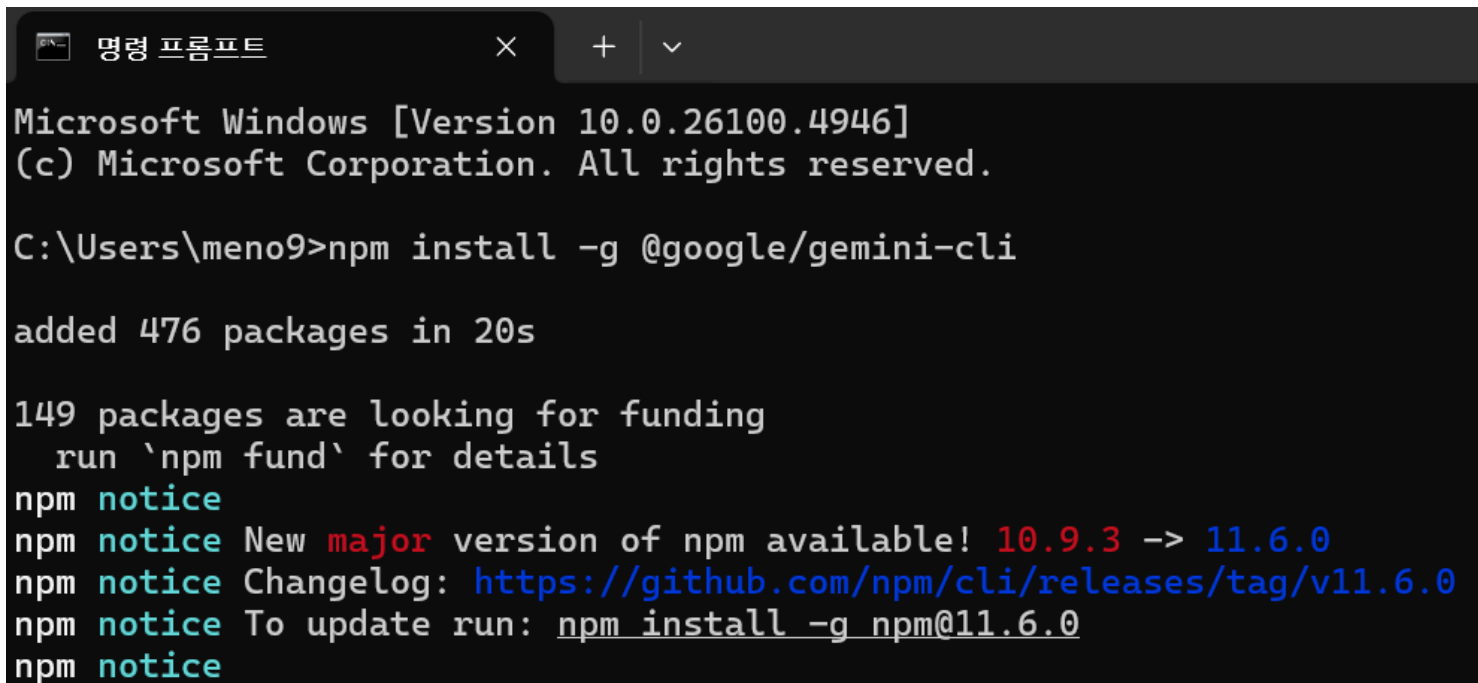
- 터미널(명령 프롬프트) 실행



Install Gemini CLI

- `npm install -g @google/gemini-cli` 입력
- 설치 완료 후 명령 프롬프트 창 닫고 재실행

팁 : PowerShell 쓰려고 하면 머리털 빠짐



```
명령 프롬프트
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\meno9>npm install -g @google/gemini-cli

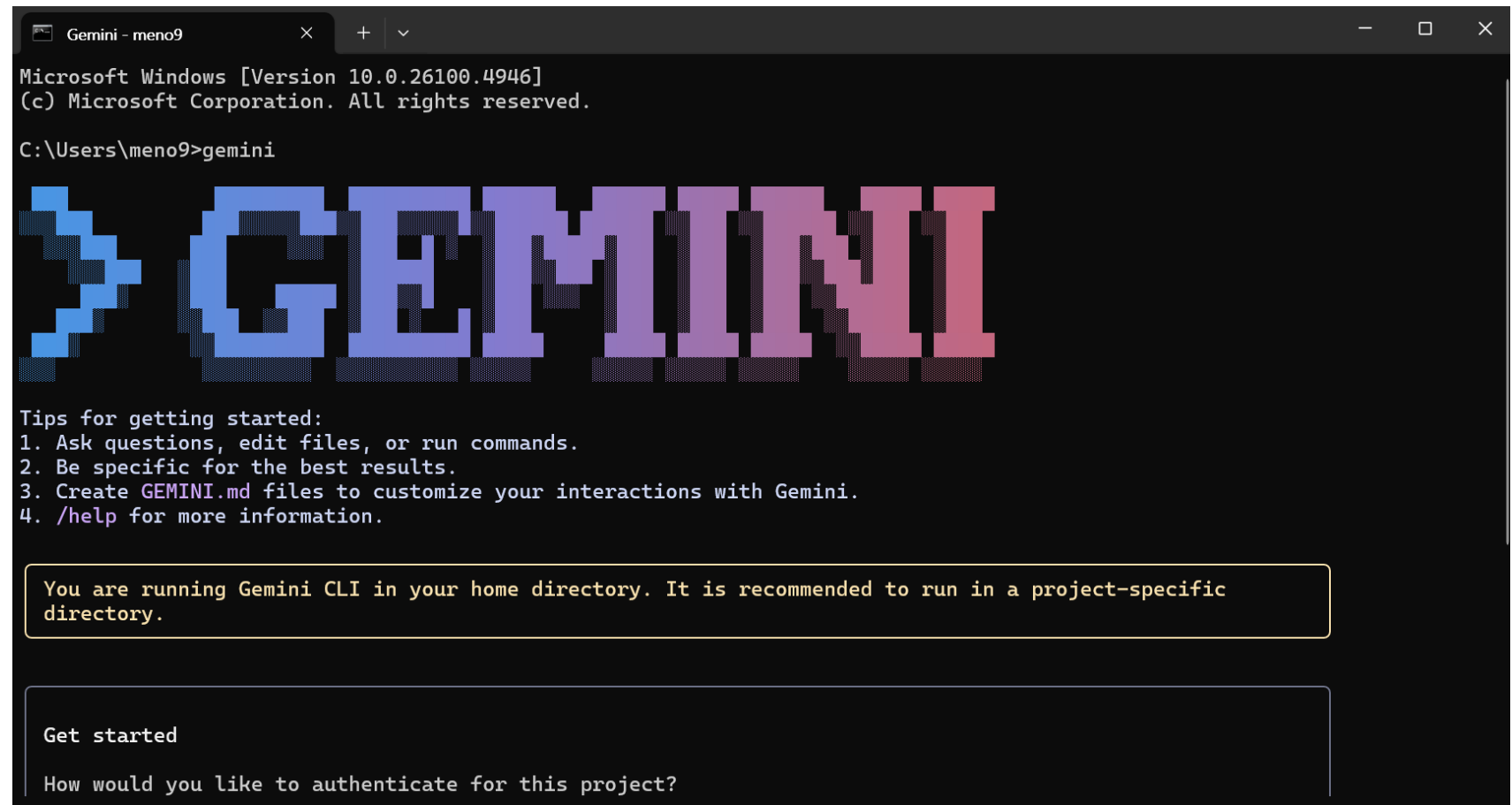
added 476 packages in 20s

149 packages are looking for funding
  run `npm fund` for details

npm notice
npm notice New major version of npm available! 10.9.3 -> 11.6.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.0
npm notice To update run: npm install -g npm@11.6.0
npm notice
```

Install Gemini CLI

- 터미널에 gemini 입력하여 작동 확인



The screenshot shows a Windows terminal window titled "Gemini - meno9". The terminal output displays the Microsoft Windows version (10.0.26100.4946) and the user's location (C:\Users\meno9). After typing the command "gemini", a large, colorful ASCII art logo for "GEMINI" is shown. Below the logo, there are "Tips for getting started" and a message indicating that the user is running Gemini CLI in their home directory. At the bottom, there is a "Get started" section with a prompt to authenticate for the project.

```
Gemini - meno9
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\meno9>gemini

> GEMINI

Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create GEMINI.md files to customize your interactions with Gemini.
4. /help for more information.

You are running Gemini CLI in your home directory. It is recommended to run in a project-specific
directory.

Get started

How would you like to authenticate for this project?
```

Install Gemini CLI

- **1. Login with Google** 에서 Enter 입력

Get started

How would you like to authenticate for this project?

- 1. Login with Google
- 2. Use Gemini API Key
- 3. Vertex AI

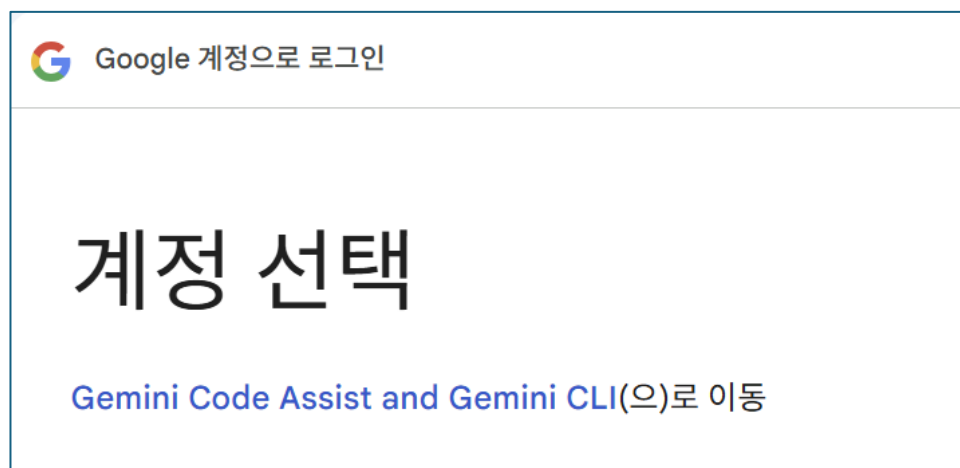
(Use Enter to select)

Terms of Services and Privacy Notice for Gemini CLI

<https://github.com/google-gemini/gemini-cli/blob/main/docs/tos-privacy.md>

Install Gemini CLI

- 웹브라우저에서 Gemini 구독중인 계정으로 로그인



Install Python

- 기존에 사용하던 Python 버전이 있으면 해당 버전을 사용
- [Python Release Python 3.12.10 | Python.org](#)

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore	SBOM
Gzipped source tarball	Source release		35c03f014408e26e2b06d576c19cac54	25.9 MB	SIG	.sigstore	SPDX
XZ compressed source tarball	Source release		0c0a6cce86d7611aed691d61e0599de0	19.6 MB	SIG	.sigstore	SPDX
macOS 64-bit universal2 installer	macOS	for macOS 10.13 and later	8f4989592c9412e51fcad70273442c22	43.6 MB	SIG	.sigstore	
Windows installer (64-bit)	Windows	Recommended	5eddb0b6f12c852725de071ae681dde4	25.7 MB	SIG	.sigstore	SPDX
Windows installer (32-bit)	Windows		ff3e7e2582811ade44916d9c3d2a64b2	24.5 MB	SIG	.sigstore	SPDX
Windows installer (ARM64)	Windows	Experimental	37df52717faa6c514342b8d43cae7026	25.0 MB	SIG	.sigstore	SPDX
Windows embeddable package (64-bit)	Windows		fe8ef205f2e9c3ba44d0cf9954e1abd3	10.6 MB	SIG	.sigstore	SPDX
Windows embeddable package (32-bit)	Windows		93beb9042be23d7b5cd528fa15e8fea7	9.5 MB	SIG	.sigstore	SPDX
Windows embeddable package (ARM64)	Windows		60ce2b5cb5a4dfcd2873a323ffe541bc	9.9 MB	SIG	.sigstore	SPDX

Install Python

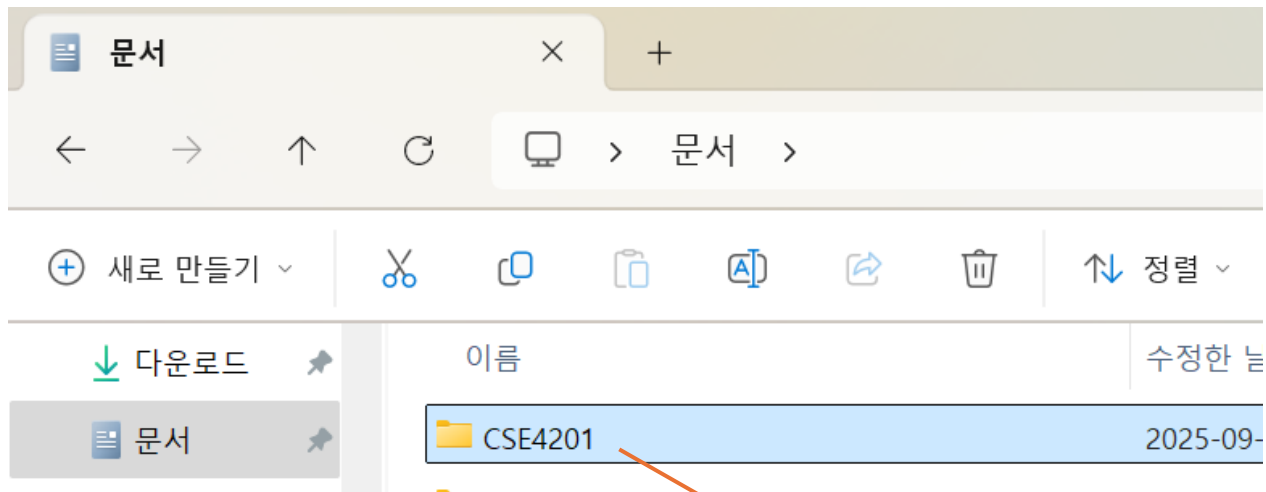
- Add python.exe to PATH 체크 후 설치



프로젝트 폴더 생성

Make Directory

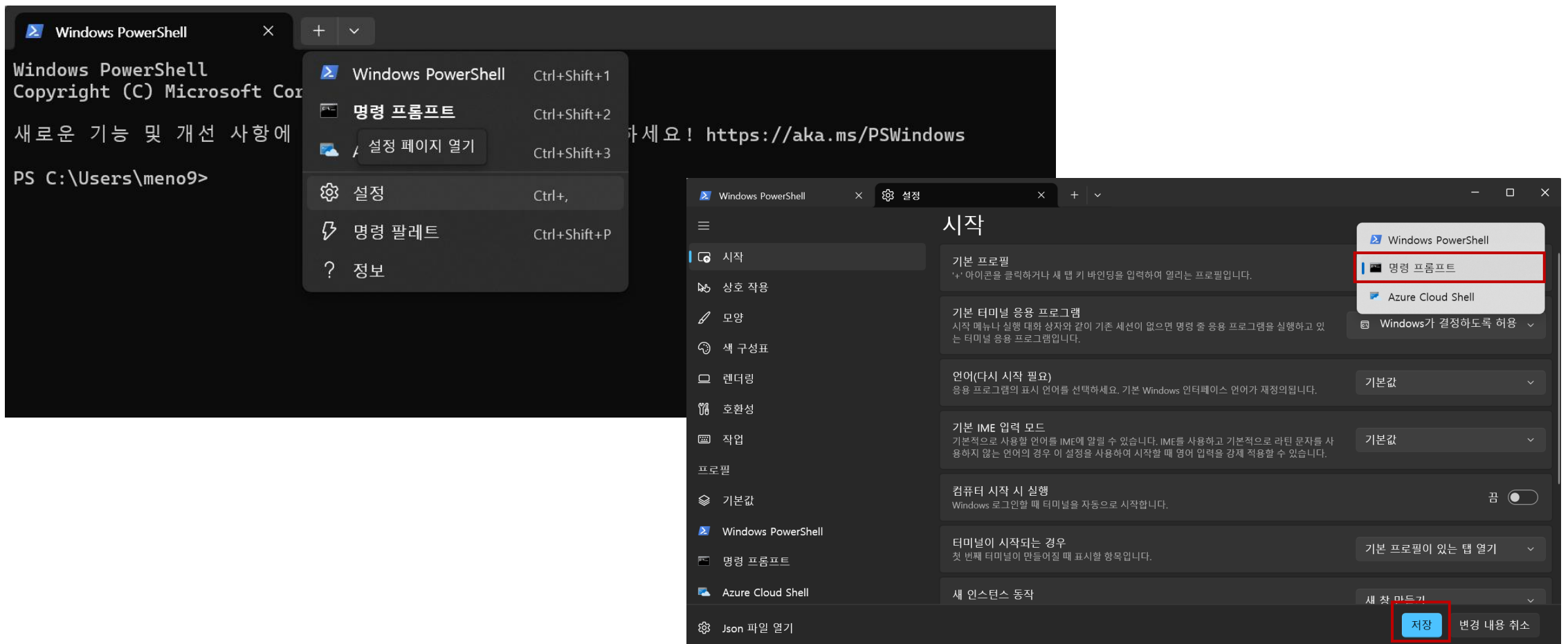
- 적당한 폴더 선택



맥북 사용자는 **폴더 우 클릭 > 서비스 > 폴더에서 새로운 터미널 탭 열기**

Make Directory

- PowerShell 나타나는 경우 > 설정 메뉴 진입 > 기본 프로필 명령 프롬프트 변경 > 저장 > 닫고 재실행



Run Gemini

- gemini 입력하여 Gemini CLI 실행



The screenshot shows a Windows terminal window titled "Gemini - CSE4201". The terminal displays the following text:

```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\meno9\Documents\CSE4201>gemini
```

Below the command prompt, the word "GEMINI" is displayed in a large, stylized, pixelated font. The letters are colored in a gradient from blue to purple to pink. To the left of the word is a blue pixelated arrow pointing right.

Below the logo, the text "Tips for getting started:" is followed by a list of four items:

1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create `GEMINI.md` files to customize your interactions with Gemini.
4. `/help` for more information.

At the bottom of the terminal, there is a text input field with a prompt character ">" and a cursor. The text inside the field is "Type your message or @path/to/file".

At the very bottom of the terminal window, the status bar shows the current directory as `~\Documents\CSE4201`, the mode as `no sandbox (see /docs)`, and the model as `gemini-2.5-pro (100% context left)`.

Rough Request

- Python을 이용하여 간단한 할일 + 메모장 프로그램을 만들어주세요.

```
> Python을 이용하여 간단한 할일 + 메모장 프로그램을 만들어주세요.■
```

```
~\Documents\CSE4201
```

```
no sandbox (see /docs)
```

```
gemini-2.5-pro (100% context left)
```

◆ 네, Python을 사용하여 간단한 콘솔 기반 할일 및 메모장 프로그램을 만들어 드리겠습니다.

하나의 Python 파일(`main.py`)에 모든 기능이 포함되며, 프로그램 실행 시 할일과 메모가 각각 `todos.json`, `notes.json` 파일에 자동으로 저장됩니다.

먼저 `main.py` 파일을 생성하겠습니다.

```
: Creating the Program Script (esc to cancel, 23s)
```

Plan Mode & Auto Mode

- [1] Plan Mode : 매 CRUD 발생 상황마다 User의 승인 요구
- [2] Auto Mode : 사용자의 Accept 요청하지 않고 논스톱 수행

먼저 `main.py` 파일을 생성하겠습니다.

? WriteFile Writing to main.py ←

... first 33 lines hidden ...

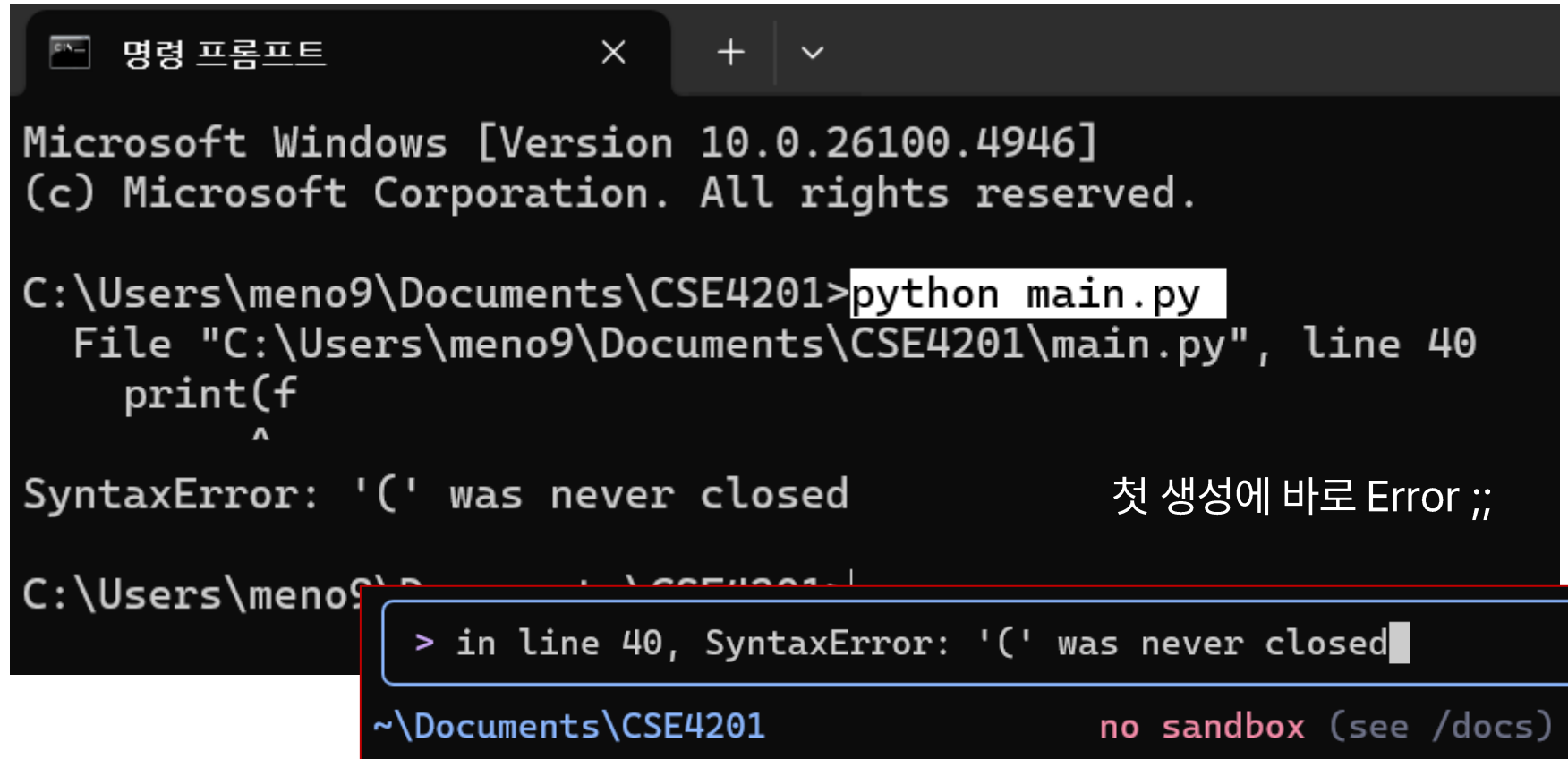
```
34     print("-----")
35
36 def add_todo(todos):
37     """새로운 할 일을 추가합니다."""
38     task = input("추가할 할 일을 입력하세요: ")
39     todos.append(task)
40     print(f
```

Apply this change?

- 1. Yes, allow once
- 2. Yes, allow always
- 3. Modify with external editor
- 4. No, suggest changes (esc)

Run Program

- 새로운 명령 프롬프트 창 실행
- Gemini의 설명에 따라 프로그램 실행 (예시 : `python main.py`)



```
명령 프롬프트
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\meno9\Documents\CSE4201>python main.py
File "C:\Users\meno9\Documents\CSE4201\main.py", line 40
    print(f
          ^
SyntaxError: '(' was never closed

C:\Users\meno9\Documents\CSE4201>
```

첫 생성에 바로 Error ;;

```
> in line 40, SyntaxError: '(' was never closed
~\Documents\CSE4201 no sandbox (see /docs)
```

Generate Document

- 프로젝트 폴더 내 적절한 위치 + 파일명으로 Document 작성 요청

Using: 1 GEMINI.md file

> 프로젝트 폴더 위치에 프로그램의 각 method에 대한 설명을 포함하는 "GEMINI.md" 파일을 생성 해 주세요.■

~\Documents\CSE4201

no sandbox (see /docs)

gemini-2.5-pro (99% context left)

main.py 프로그램 설명

이 문서는 `main.py` 스크립트에 포함된 각 함수(메서드)의 기능에 대해 설명합니다.

공통 함수

`load_data(filename)`

- 설명:** `todos.json` 또는 `notes.json` 같은 JSON 파일을 불러와 Python 리스트 객체로 변환합니다. 파일이 존재하지 않거나, 내용이 비어있거나, 형식이 올바르지 않을 경우, 오류를 방지하기 위해 빈 리스트(`[]`)를 반환합니다.
- 매개변수:** `filename` (str) - 불러올 파일의 이름.
- 반환값:** `list` - 파일에서 불러온 데이터가 담긴 리스트.

Limitation

- 사용자는 Console 수준 Program 작성을 요청하지 않음
- LLM 모델이 사용자의 요청을 바탕으로 적당한 코드를 확률적으로 추론해냄

SRS (Software Requirements Specification)

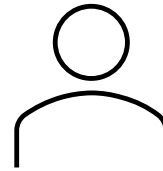
Blueprint Design

- Vibe Coding으로 Software Development를 하기 전 먼저 **프로젝트에 대한 명세서를 작성**
- 이론에서 배우게 되겠지만, 소프트웨어 공학에는 BRD, PRD, SRS 라는 요구사항 공학(Requirements Engineering)이 포함됨 -> Why (BRD) + What (PRD) + How (SRS) ? 를 정의하는 과정
- 우리는 **Fully-Agentic Coding**을 하려는 것이 아니라 귀찮은 작업을 AI 개발자에게 하청을 주는 것이 목적임
- 따라서, **자연어 기반 프로그래밍**이라는 것에는 필연적으로 요구사항 명세서의 주입과 의견 교환이 필요하게 됨

Blueprint Design

도메인 지식이 충분할 때

기초 PRD+SRS 직접 작성



도메인 지식이 부족할 때

LLM와 PRD 개발을 위한
Q&A 진행

안녕하세요 Gemini, 사용자가 GUI 기반으로 간단하게 사용할 수 있는 할일 리스트(제목) + Markdown 지원 메모장 프로그램을 만들려고 합니다. 여기에 대한 Product Requirements Document를 당신과 함께 작성하고자 합니다.

PRD 문서는 주니어 개발자도 쉽게 이해할 수 있도록 작성되어야 합니다. PRD를 작성하는 과정에서 핵심 요구사항을 식별한 뒤 1.1, 1.2, 1.3 같이 번호를 붙여 저에게 질문을 하며 작성 및 수정해주세요.



Deep Research



동영상



이미지



Canvas

...



Blueprint Design

안녕하세요 Gemini, 사용자가 GUI 기반으로 간단하게 사용할 수 있는 할일 리스트(제목) + Markdown 지원 메모장 프로그램을 만들려고 합니다. 여기에 대한 Product Requirements Document를 당신과 함께 작성하고자 합니다.

PRD 문서는 주니어 개발자도 쉽게 이해할 수 있도록 작성되어야 합니다. PRD를 작성하는 과정에서 핵심 요구사항을 식별한 뒤 1.1, 1.2, 1.3 같이 번호를 붙여 저에게 질문을 하며 작성 및 수정해주세요.

1. 할일(To-Do) 목록 핵심 기능

왜 중요한가? 이 부분은 프로그램의 가장 기본적인 골격입니다. 사용자가 작업을 추가, 완료, 수정하는 방식이 프로그램의 전체적인 사용성을 결정합니다.

- **1.1. 새로운 할일 추가 방식:** 사용자가 새로운 할일을 목록에 추가할 때 어떤 인터페이스를 제공할까요?
 - A) 화면 상단 또는 하단에 있는 '+' 버튼을 클릭하면 새 항목이 추가된다.
 - B) 목록의 마지막에 항상 비어있는 입력창이 있어, 텍스트를 입력하고 Enter 키를 누르면 추가된다.
 - C) 다른 방식 제안
- **1.2. 할일 완료 처리:** 사용자가 할일을 '완료' 상태로 변경했을 때 시각적으로 어떻게 처리할까요?
 - A) 할일 텍스트에 취소선(strikethrough)을 긋고 목록에 그대로 둔다.
 - B) '완료된 항목'이라는 별도의 섹션으로 이동시킨다.
 - C) 목록에서 즉시 삭제(숨김) 처리한다.
- **1.3. 할일 수정 및 삭제:** 기존에 생성된 할일의 제목을 수정하거나 목록에서 완전히 삭제하는 기능은 어떻게 구현할까요?
 - A) 할일 항목을 더블클릭하면 제목을 수정할 수 있고, 항목 위에 마우스를 올리면 나타나는 'x' 버튼으로 삭제한다.
 - B) 할일 항목을 우클릭하면 '수정', '삭제' 옵션이 포함된 컨텍스트 메뉴가 나타난다.
 - C) 다른 방식 제안

위 질문들에 대한 답변을 기준으로 PRD를 구체화하고, 다음 단계인 Markdown 메모장 기능과 데이터 저장 방식에 대한 요구사항을 함께 정의해 나가겠습니다. 어떤 방향으로 진행하고 싶으신가요?

Blueprint Design

1.1. 새로운 할일 추가 방식: (C) UI를 2분할 하고, 좌측 최상단에 "새 할일 (+)" 버튼을 추가하여 쉽게 새 페이지를 작성할 수 있도록 할 것 입니다.

1.2. 할일 완료 처리: (B)

1.3. 할일 수정 및 삭제: (B)

+  Deep Research  동영상  이미지  Canvas ...

2.1. Markdown 렌더링 방식: (B)

2.2. 지원할 Markdown 문법 범위: (A)

3.1. 데이터 저장 형식: (B)

3.2. 데이터 저장 및 불러오기 시점: (A) + (B) -> 사용자가 할일을 추가/수정/삭제하면 즉시 파일 저장이 발생합니다. 그러나 저장은 기본적으로 사용자가 프로그램을 종료하거나, 저장 버튼을 눌렀을 때 수행되며 이 때 새로운 버전이 저장됩니다. 메모를 수정할 때 자동 저장이 되지만, 이는 임시 파일로 저장하도록 하며 File Read/Write를 최소화 하기 위해 변경 사항이 있고 일정 시간동안 입력이 없을 때 임시 파일을 업데이트 하도록 합니다.

+  Deep Research  동영상  이미지  Canvas ...



Blueprint Design

The image shows a Notepad++ editor window on the left and a file explorer window on the right.

Notepad++ Editor:

- Tab: ## PRD (v1.0) 간단한 할일 목록 + M
- Menu: 파일, 편집, 보기
- Content:
 - ## **PRD (v1.0): 간단한 할일 목록 + Markdown
 - ### **1. 개요 (Overview)**
본 문서는 데스크톱 환경에서 사용할 수 있는 것을 정의합니다. 사용자는 할일(To-Do) 목록을 상문법을 지원하는 상세 메모를 작성할 수 있습니다.
 - ### **2. 목표 (Goals)**
 - * **직관적인 사용성:** 사용자는 별도의 학습 없이 사용할 수 있습니다.
 - * **핵심 기능 집중:** 할일 관리와 메모 작성에 집중합니다.
 - * **Markdown 지원:** 사용자가 서식을 갖춘 글을 작성할 수 있습니다.
- Status Bar: 줄 51, 열 102 | 2,402자 | 일반 텍스트

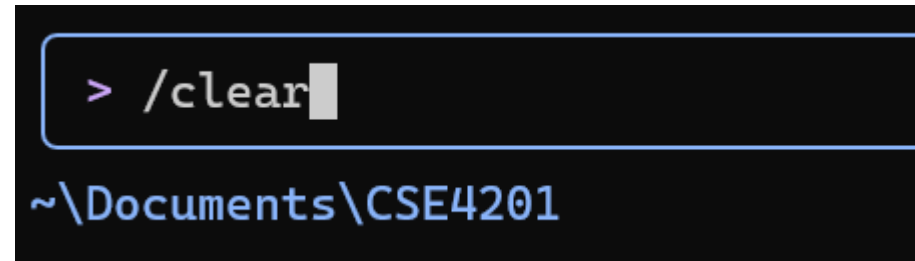
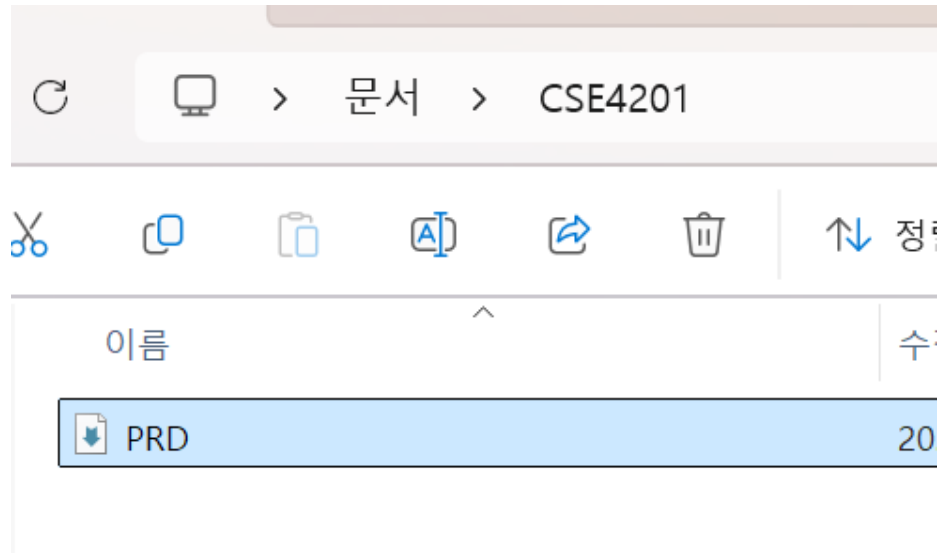
File Explorer Window:

- Title: 다른 이름으로 저장
- Path: > 문서 > CSE4201
- Buttons: 구성, 새 폴더
- Left Panel: 갤러리, 보충 - 개인
- Right Panel (Files):

이름	수정한
main	2025-0
todos	2025-0
- File Name: 파일 이름(N): PRD.md
- File Type: 파일 형식(T): 모든 파일
- Buttons: ^ 폴더 숨기기
- Encoding: 인코딩: UTF-8

Software Requirements

- 불필요한 기존 파일들을 전부 제거하고, Gemini에서 /clear 입력



Software Requirements

- Gemini에서 SRS 파일 작성 요청
- 개발 단계 관리가 용이하도록 체크박스 형태로 작성하도록 요구 -> TODO List 관리

Using: 1 GEMINI.md file

> 안녕하세요 Gemini, TODO List + 메모장 기능이 담긴 Python 프로그램 하나를 만들려고 합니다. PRD.md 파일을 살펴보고 코드 구현을 위한 Software Requirements markdown 파일을 만들어주세요. 각 요구사항은 구현이 완료되었는지 확인할 수 있도록 체크박스 형태로 작성 바랍니다.■

~\Documents\CSE4201

no sandbox (see /docs)

gemini-2.5-pro (100% context left)

Users > meno9 > Documents > CSE4201 > Software_Requirements.md > # Software Requirements (v1.0) > ## 4.4. 데이터 관리

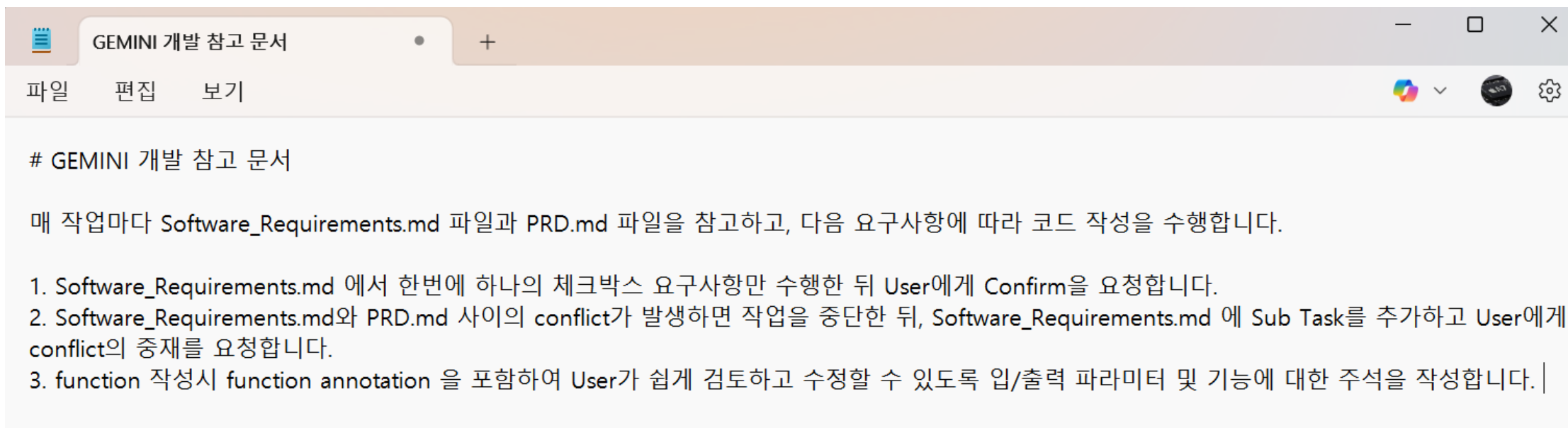
Software Requirements (v1.0)

4.1. 화면 구성 (UI Layout)

- [] ****4.1.1. 2분할 레이아웃:**** 전체 화면을 좌측 '할일 목록' 패널과 우측 '메모 편집' 패널로 나눕니다.
- [] ****4.1.2. 좌측 '할일 목록' 패널:****
 - [] ****컨트롤 버튼:**** 패널 최상단에 "새 할일 (+)" 버튼과 "저장" 버튼이 나란히 위치합니다.
 - [] ****할일 리스트:**** 생성된 할일들이 세로로 나열됩니다.
 - [] ****완료된 항목 섹션:**** '할일 리스트' 하단에 구분선과 함께 '완료된 항목' 섹션이 존재합니다.
- [] ****4.1.3. 우측 '메모 편집' 패널:**** 좌측에서 선택된 할일의 상세 내용을 보여주는 편집 공간입니다.

Software Requirements

- 텍스트 편집기 및 메모장에서 Task 관리 문서 작성



Implementation

Using: 1 GEMINI.md file

> 구현을 시작하기 전, 프로젝트 디렉터리에서 PRD.md, Software_Requirements.md, GEMINI.md 를 읽은 뒤
요구사항에 따라 순차적으로 구현을 시작합니다. 사용하는 GUI 라이브러리는 Tkinter로 하겠습니다. 외부
라이브러리에 대해서는 보수적으로 접근 해 주세요. █

~\Documents\CSE4201

no sandbox (see /docs)

gemini-2.5-pro (99% context left)

C: > Users > meno9 > Documents > CSE4201 > Software_Requirements.md > # Software Requirements (v1.0) > ## 4.4. 데이터 관리

```

1  # Software Requirements (v1.0)
2
3  ## 4.1. 화면 구성 (UI Layout)
4  - [x] **4.1.1. 2분할 레이아웃:** 전체 화면을 좌측 '할일 목록' 패널과 우측 '메모 편집' 패널로 나눕니다.
5  - [x] **4.1.2. 좌측 '할일 목록' 패널:**
6      - [x] **컨트롤 버튼:** 패널 최상단에 "새 할일 (+)" 버튼과 "저장" 버튼이 나란히 위치합니다.
7      - [x] **할일 리스트:** 생성된 할일들이 세로로 나열됩니다.
8      - [x] **완료된 항목 섹션:** '할일 리스트' 하단에 구분선과 함께 '완료된 항목' 섹션이 존재합니다.
9  - [x] **4.1.3. 우측 '메모 편집' 패널:** 좌측에서 선택된 할일의 상세 내용을 보여주는 편집 공간입니다.
10
11 ## 4.2. 할일(To-Do) 관리 기능
12 - [x] **1.1. 새로운 할일 추가:** "새 할일 (+)" 버튼 클릭 시, '할일 리스트' 최상단에 "새로운 할일" 항목이 생성되고 즉시 이름 변경 가능 상태가 됩니다.
13 - [ ] **1.2. 할일 완료 처리:** 할일 앞의 체크박스를 클릭하면 해당 항목은 '완료된 항목' 섹션으로 이동하며 취소선이 적용됩니다.
14 - [ ] **1.3. 할일 수정 및 삭제:** 할일 항목 위에서 마우스 우클릭 시 나타나는 컨텍스트 메뉴를 통해 '이름 바꾸기', '삭제' 기능을 제공합니다.
15

```

Conflict Handling

```

Gemini - CSE4201  ×  +  ▾

다음은 '4.3. Markdown 메모 편집 기능' 입니다. 요구사항에는 WYSIWYG(위지윅) 방식의 렌더링이 명시되어
있습니다.

하지만 기본 Tkinter 라이브러리만으로는 WYSIWYG Markdown 편집기를 구현하는 것이 매우 어렵고 복잡합니다. 이
기능을 구현하려면 외부 라이브러리를 사용하거나, 웹 렌더링 엔진을 직접 만드는 수준의 작업이 필요하게 되어
"외부 라이브러리에 대한 보수적 접근" 원칙에 어긋날 수 있습니다.

따라서 다음과 같은 대안을 제안합니다.

제안: 메모 편집기는 Markdown 문법을 실시간으로 렌더링하는 대신, 일반 텍스트 편집기처럼 작동하게 합니다.
사용자가 입력한 Markdown 텍스트는 그대로 .md 파일에 저장됩니다. 렌더링된 결과는 VS Code, Typora 등과 같은
전문 Markdown 뷰어를 통해 확인할 수 있습니다.

이 방식은 핵심 기능인 'Markdown 형식으로 메모 저장'을 만족시키면서, 외부 라이브러리 없이 구현을 단순하게
유지할 수 있는 가장 현실적인 방법입니다.

이렇게 진행해도 괜찮을까요?

Using: 1 GEMINI.md file

> █ Type your message or @path/to/file

~\Documents\CSE4201 > 잠시 구현을 중단하고,tkinter 기반 Ui에서 WYSIWYG Markdown 편집기를 구현하는 가장 적절한 방법이
무엇일지 의견을 부탁드립니다█

~\Documents\CSE4201 no sandbox (see /docs) gemini-2.5-pro (97% context left)

```

Separation of Responsibility

◆ main.py 파일을 읽었습니다. 이제 데이터 관리의 첫 단계를 구현하겠습니다.

1. json 및 os 라이브러리를 가져옵니다.
이 내용들을 main.py에 반영하겠습니다. 이는 꽤 큰 변경이 될 것입니다.

? Edit main.py: ... => import tkinter as tk... <

```
... first 290 lines hidden ...
211 +         self.save_all_memos()
212 +         self.root.destroy()
213 +
214 if __name__ == "__main__":
215     root = tk.Tk()
216     app = TodoApp(root)
```

Apply this change?

1. Yes, allow once
2. Yes, allow always
3. Modify with external editor
- 4. No, suggest changes (esc)

Press ctrl-s to show more lines
" Waiting for user confirmation...

Using: 1 GEMINI.md file
~\Documents\CSE4201

Using: 1 GEMINI.md file

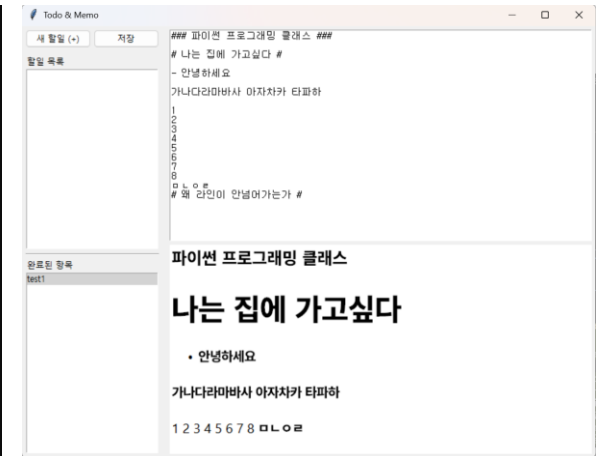
> main.py에 기능이 너무 많이 집약되었습니다. Software_Requirements.md 를 수정하여 UI와 기능을 서로 다른 클래스 및 파일로 관리할 수 있도록 수정해주세요.■

~\Documents\CSE4201

no sandbox (see /docs)

gemini-2.5-pro (99% context left)

```
36 +
37 + ## 5. 아키텍처 (Architecture)
38 + - [ ] **5.1. 파일 및 클래스 분리:** UI와 데이터 로직을 분리하여 유지보수성을 향상시킵니다.
39 + - [ ] **`main.py`:** 애플리케이션 진입점. `Tk` 루트 윈도우를 생성하고 `TodoApp`을
    실행합니다.
40 + - [ ] **`app.py`** (`TodoApp` 클래스):** 메인 애플리케이션 클래스. UI와 데이터 관리
    클래스를 조정하고, 모든 이벤트 핸들러(비즈니스 로직)를 포함합니다.
41 + - [ ] **`ui.py`** (`AppUI` 클래스):** 모든 Tkinter 위젯의 생성, 배치 및 UI 구성을
    담당합니다. 이벤트 발생 시 `TodoApp`의 핸들러를 호출합니다.
42 + - [ ] **`data_manager.py`** (`DataManager` 클래스):** 파일(I/O) 및 데이터(JSON, .md) 관련
    모든 처리를 담당합니다.
```



Discussion

Human-AI collaboration