

# **Software Engineering**

## **Hands-On Lecture – 04**

Software Engineering in the era of AI

# Goal

- Base64 ↔ File Converter

## REST API Model

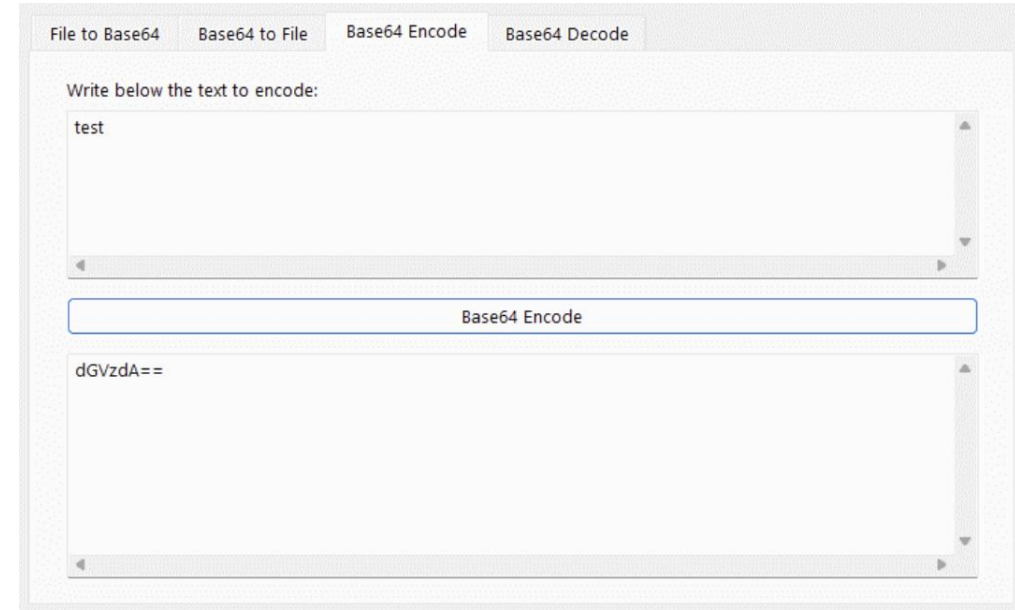


```
def file_to_b64(path: str) -> str:  
    with open(path, "rb") as f:  
        raw = f.read()  
    return base64.b64encode(raw).decode()
```

```
def b64_to_file(s: str, out_path: str) -> str:  
    data = base64.b64decode(s)  
    Path(out_path).write_bytes(data)  
    return out_path
```

# Goal

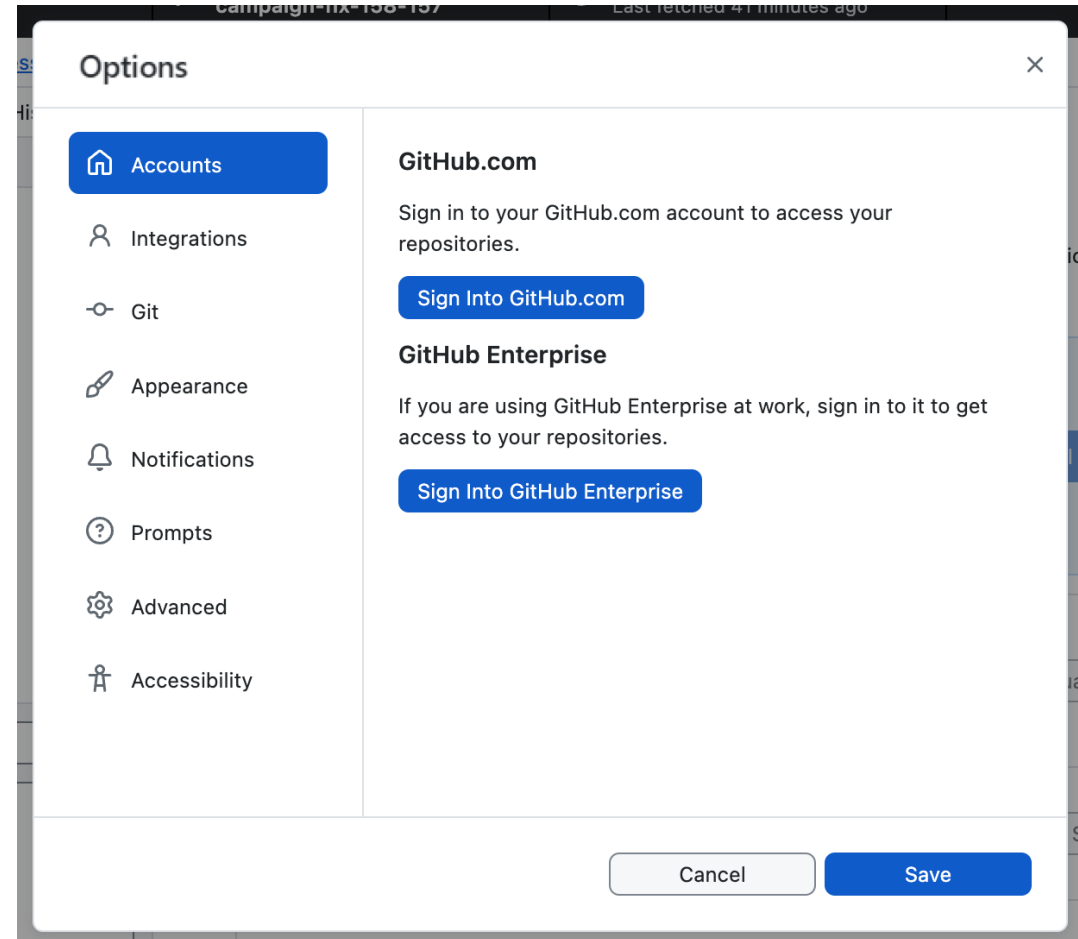
- PyQt6 GUI Toolkit 사용
- Github Repository 연결
- File to Base64 구현 및 테스트
- 배치 파일 생성
- 확장



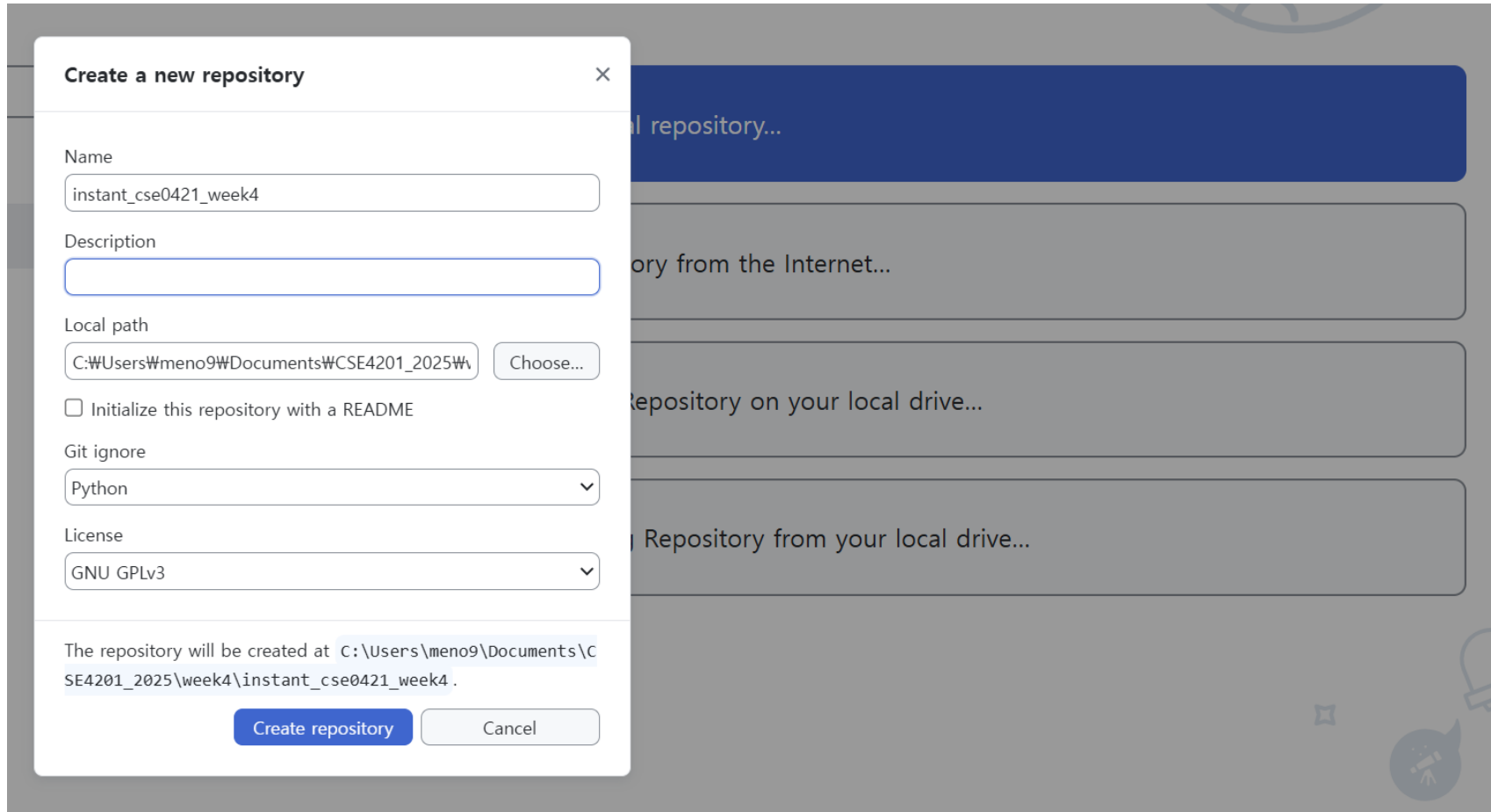
# Github Desktop

# Github Desktop 설치

- [Download GitHub Desktop | GitHub Desktop](#)



# Create a new repo



The screenshot shows a 'Create a new repository' dialog box with the following fields and options:

- Name:** instant\_cse0421\_week4
- Description:** (empty text box)
- Local path:** C:\Users\meno9\Documents\CSE4201\_2025\week4 (with a 'Choose...' button)
- ☐ Initialize this repository with a README
- Git ignore:** Python (dropdown menu)
- License:** GNU GPLv3 (dropdown menu)

At the bottom, it states: 'The repository will be created at C:\Users\meno9\Documents\CSE4201\_2025\week4\instant\_cse0421\_week4.'

Buttons: 'Create repository' (blue) and 'Cancel' (grey).

The background shows a blurred interface with options like 'Create repository...', 'Repository from the Internet...', 'Repository on your local drive...', and 'Repository from your local drive...'.

# 실습1 : 가상환경 생성 및 PyQt6 설치

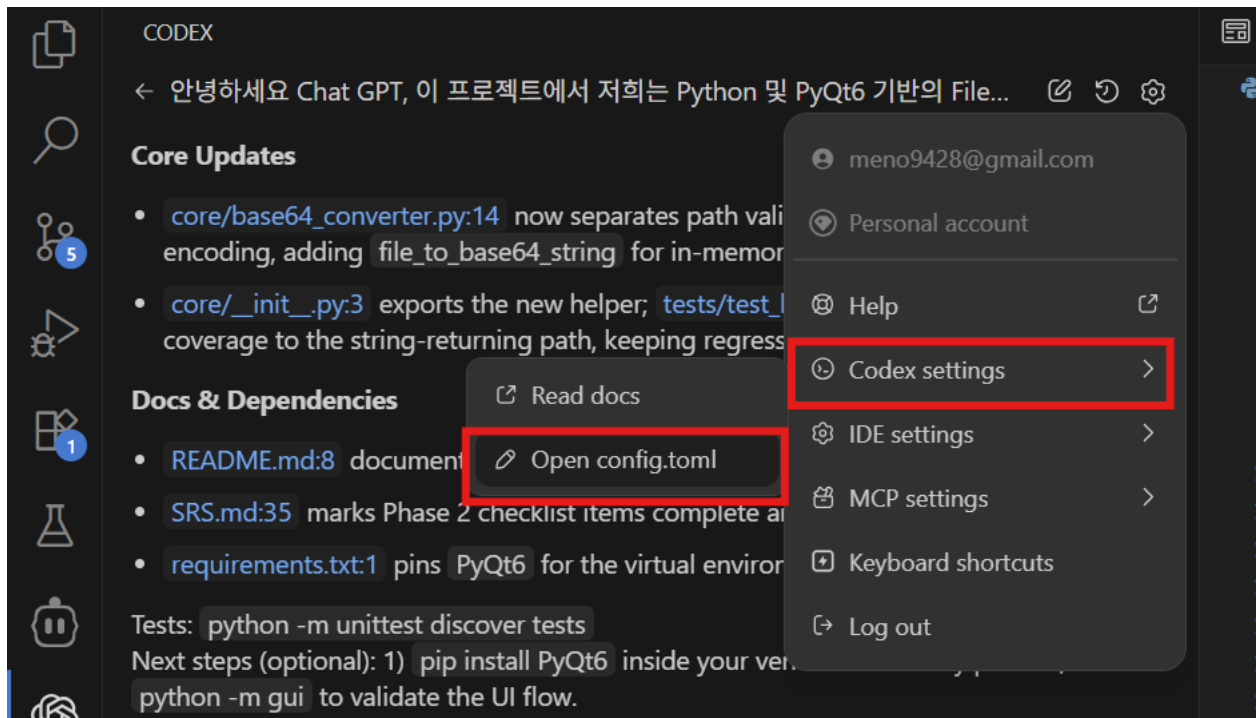
- 생성한 폴더에서 터미널 Open
- 파이썬의 경우 `python -m venv {name}` 를 통해 간단히 가상환경 생성
  - `python -m venv venv`                      -> (mac) : `python3 -m venv venv`
  - `venv\scripts\activate`                  -> (mac) : **source** `venv/bin/activate`
- **[라이브러리 설치]** `pip install PyQt6`
- **[VSCODE 실행]** code

# 실습2 : 목표 분석 및 PRD 설계

- 메모장 / 텍스트 편집기 활용
- 포함해야 하는 것:
- [1] 최종적으로 무엇을 만들려고 하는 것인지 -> PyQt6 기반의 File to Base64 Converter
- [2] 구현 계획은 어떻게 되는가 ?
  - 우선 기능 검증을 위해 CLI 에서 실행 가능한 File to Base64 변환 코드를 만들 것 이다
  - 입력 : 파일 경로, 출력 : .txt 파일
  - 이후 PyQt6 을 기반으로 하는 GUI 형태로 확장 할 것
- [3] 구현을 위한 SRS 및 체크리스트 markdown 문서 작성 요청



# [Codex CLI] Windows Approval Issue



```
C: > Users > meno9 > .codex > config.toml
1  model = "gpt-5-codex"
2  approval_mode = "full-auto"
```

**approval\_mode = "full-auto"**

추가 (복붙 말고 직접 작성)

-> Mac / Linux (WSL) 에서는 문제 없으나 Windows  
의 경우 config.toml 수정 필요

# Claude Code 샘플

> 안녕하세요 Claude, PyQt6 기반의 File to Base64 Converter를 만들려고 하는데 우선적으로 기능 검증을 위해 CLI 에서 실행 가능한 File to Base64 변환 코드(input: filepath, output: .txt file)를 구현한 뒤 이후 순차적으로 PyQt6 기반의 GUI 툴로 확장하려고 합니다. 이를 위하여 단계별 구현 및 참조가 가능하도록 SRS markdown 문서를 만들고, 각 구현 단계를 체크리스트화 하여 관리하려고 합니다. 먼저 markdown 파일을 만들어주시겠어요? 필요한 부분이 있다면 저에게 질문하여 명세를 구체화해주세요. █

> CLI 버전 관련: \

1. 파일 형식에는 제한이 없습니다.\
2. 출력 .txt 파일은 테스트를 위한것이므로 temp.txt로 통일합니다.\
3. 업로드 파일은 4MB로 제한됩니다. 4MB를 초과하는 파일은 Exception을 발생시킵니다.\
4. 에러 처리의 경우 파일 없음, 권한 없음 수준으로 처리합니다. \

\  
PyQt6 GUI는 추후 구현단계에서 재논의 하며, 이를 SRS 문서에 기술합니다.\  
\  
일반적인 요구사항: 프로젝트 명은 pyqtbase64, 대상 버전은 3.10, 추가 의존성 라이브러리 사용에는 제한이 없습니다. █

- 안녕하세요! File to Base64 Converter 프로젝트를 위한 SRS 문서를 작성하기 전에 몇 가지 질문을 드려서 명세를 구체화하겠습니다.

- 프로젝트 명세를 구체화하기 위해 다음 사항들을 확인하고 싶습니다:

## CLI 버전 관련:

1. 지원할 파일 형식에 제한이 있나요? (모든 파일 형식 vs 특정 형식만)
2. 출력 .txt 파일의 이름 규칙은 어떻게 할까요? (예: 원본파일명\_base64.txt)
3. 파일 크기 제한이 있나요?
4. 에러 처리는 어떤 수준까지 필요한가요? (파일 없음, 권한 없음, 메모리 부족 등)

## PyQt6 GUI 버전 관련:

1. 드래그 앤 드롭 기능이 필요한가요?
2. 여러 파일 동시 변환 기능이 필요한가요?
3. 변환 진행률 표시가 필요한가요?
4. Base64에서 파일로 역변환 기능도 필요한가요?
5. 변환 이력 저장/관리 기능이 필요한가요?

## 일반적인 요구사항:

1. 프로젝트 이름은 어떻게 정하시겠어요?
2. 타겟 Python 버전은 무엇인가요?
3. 추가 의존성 라이브러리 사용에 제한이 있나요?

이 정보들을 바탕으로 상세한 SRS 문서를 작성하겠습니다.

# Claude Code 샘플

## 2.3 구현 체크리스트

### Phase 1.1: 기본 구조 설계

- [] 프로젝트 디렉토리 구조 설정
- [] 메인 모듈 (main.py) 생성
- [] Base64 변환 함수 설계

### Phase 1.2: 핵심 기능 구현

- [] 파일 읽기 기능 구현
- [] Base64 인코딩 기능 구현
- [] temp.txt 파일 생성 및 저장 기능

### Phase 1.3: 에러 처리 구현

- [] 파일 존재 여부 확인
- [] 파일 접근 권한 확인
- [] 파일 크기 검증 (4MB 제한)
- [] 사용자 정의 Exception 클래스 구현

### Phase 1.4: CLI 인터페이스 구현

- [] 명령행 인자 처리
- [] 사용법 도움말 구현
- [] 진행 상황 표시 (옵션)

### Phase 1.5: 테스트 및 검증

- [] 다양한 파일 형식 테스트
- [] 에러 상황 테스트
- [] 성능 테스트 (4MB 파일)
- [] 코드 리뷰 및 최적화

## 3. 2단계: PyQt6 GUI 확장 (추후 구현)

일반적인 요구사항:

> Phase1 구현을 시작합니다. 프로젝트 디렉토리 구조에서 base64 모듈은 ./core 폴더 내에 위치해야 합니다. 각 단계를 진행 할 때 마다 저에게 다음 구현 단계를 수행해도 될지 계속 물어봐주세요. █

PyQt6 GUI는 추후 구현단계에서 재논의 하며, 이를 SRS 문서에 기술합니다.\

일반적인 요구사항: 프로젝트명은 pyqtbase64, 대상 버전은 3.10, 추가 의존 라이브러리 사용에는 제한이 없습니다.

- Write(pyqtbase64\_srs.md)
  - └ Wrote 1 lines to pyqtbase64\_srs.md
    - # PyQtBase64 - Software Requirements Specification (SRS)
    - ## 1. 프로젝트 개요
    - ### 1.1 프로젝트명
    - PyQtBase64
    - ### 1.2 프로젝트 목적
    - 파일을 Base64 인코딩으로 변환하는 CLI 도구를 구현하고, 이후 PyQt6 기반의 GUI 애플리케이션으로 확장하는 단계별 개발 프로젝트

- SRS 문서를 생성했습니다! pyqtbase64\_srs.md 파일에 요구사항을 정리하고 체크리스트를 포함했습니다.

문서의 주요 구성:

- 1단계 CLI 구현을 위한 상세한 체크리스트
- 에러 처리 요구사항 (파일 없음, 권한 없음, 4MB 제한)
- 2단계 GUI 확장 계획 (추후 재논의)
- 성공 기준 및 리스크 분석

이제 1단계 CLI 구현을 시작하시겠습니까?

>

> for shortcuts

### Phase 1.1: 프로젝트 디렉토리 구조 설정

다음과 같은 디렉토리 구조를 생성하려고 합니다:

```
pyqtbase64/
├── main.py
├── core/
│   ├── __init__.py
│   └── base64_converter.py
└── pyqtbase64_srs.md
```


# Claude Code 샘플

## 7. 구현 진행 상황

### 7.1 완료된 작업 (2025-09-22)

- ☒ 프로젝트 구조 설계 및 디렉토리 생성
- ☒ Base64Converter 클래스 완전 구현
  - 파일 크기 검증 (4MB 제한)
  - 바이너리 파일 읽기
  - Base64 인코딩
  - temp.txt 저장
  - 통합 워크플로우 메서드
- ☒ 사용자 정의 Exception (FileSizeExceedError)
- ☒ 완전한 타입 어노테이션 및 docstring
- ☒ CLI 기본 구조 (main.py)

### 7.2 진행 중인 작업

-  CLI와 코어 모듈 연동

### 7.3 남은 작업

- ⌚ 실제 변환 기능 통합
- ⌚ 종합 테스트

- 이제 에러 시나리오들을 테스트해보겠습니다:

- 1. 파일 없음 에러 테스트:

```
Bash(python main.py nonexistent_file.txt)
└─ Error: [INFO] Converting file: nonexistent_file.txt
   [ERROR] File not found: File not found: nonexistent_file.txt
```

- 2. 4MB 제한 테스트를 위한 큰 파일 생성:

```
Bash(python -c "with open('test_files/large_file.bin', 'wb') as f: f.write(b'x' * (5 * 1024 * 1024))")
└─ (No content)
```

```
Bash(python main.py test_files/large_file.bin)
└─ Error: [INFO] Converting file: test_files/large_file.bin
   [ERROR] File size error: File size (5.00MB) exceeds 4MB limit:
   test_files/large_file.bin
```

- 3. 도움말 표시 테스트:

```
Bash(python main.py --help)
└─ usage: PyQtBase64 [-h] [--output OUTPUT] [--version] filepath

Convert files to Base64 encoding
... +9 lines (ctrl+r to expand)
```

- 완벽합니다! 모든 테스트가 성공적으로 완료되었습니다.

#### 테스트 결과 요약:

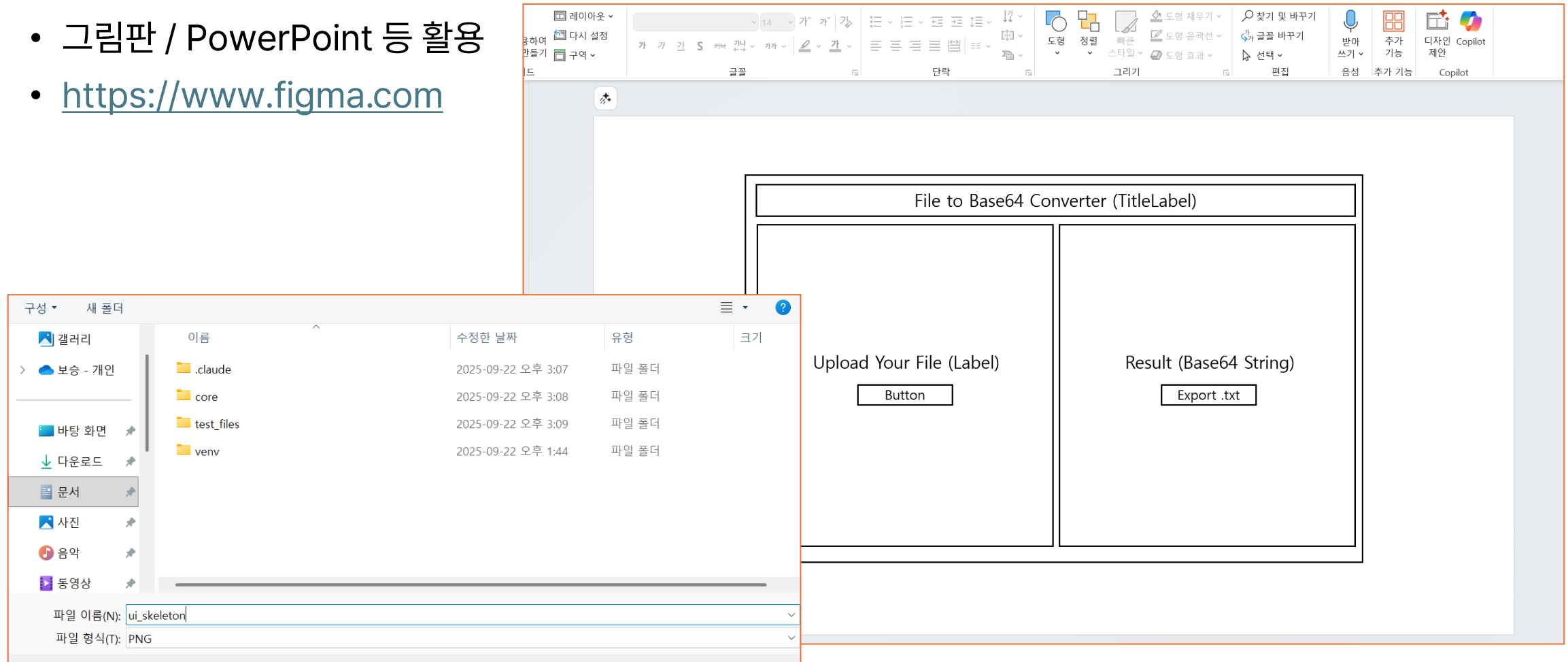
☒ 텍스트 파일 변환: 정상 작동, Base64 인코딩 검증 완료 ☒ 바이너리 파일 변환: 정상 작동, 커스텀 출력 파일 기능 확인 ☒ 파일 없음 에러: 적절한 에러 메시지 출력 ☒ 4MB 크기 제한: 5MB 파일에서 정확히 에러 발생 ☒ CLI 인터페이스: 도움말 및 버전 정보 정상 표시

다음 단계: PyQtBase64 CLI 구현이 완전히 완료되었습니다!

Phase 1의 모든 요구사항이 충족되었으므로, 이제 PyQt6 GUI 구현 단계로 넘어갈 수 있습니다. GUI 구현을 시작하시겠습니까, 아니면 다른 개선사항이 필요하신가요?

# UI 스켈레톤 제작

- 그림판 / PowerPoint 등 활용
- <https://www.figma.com>

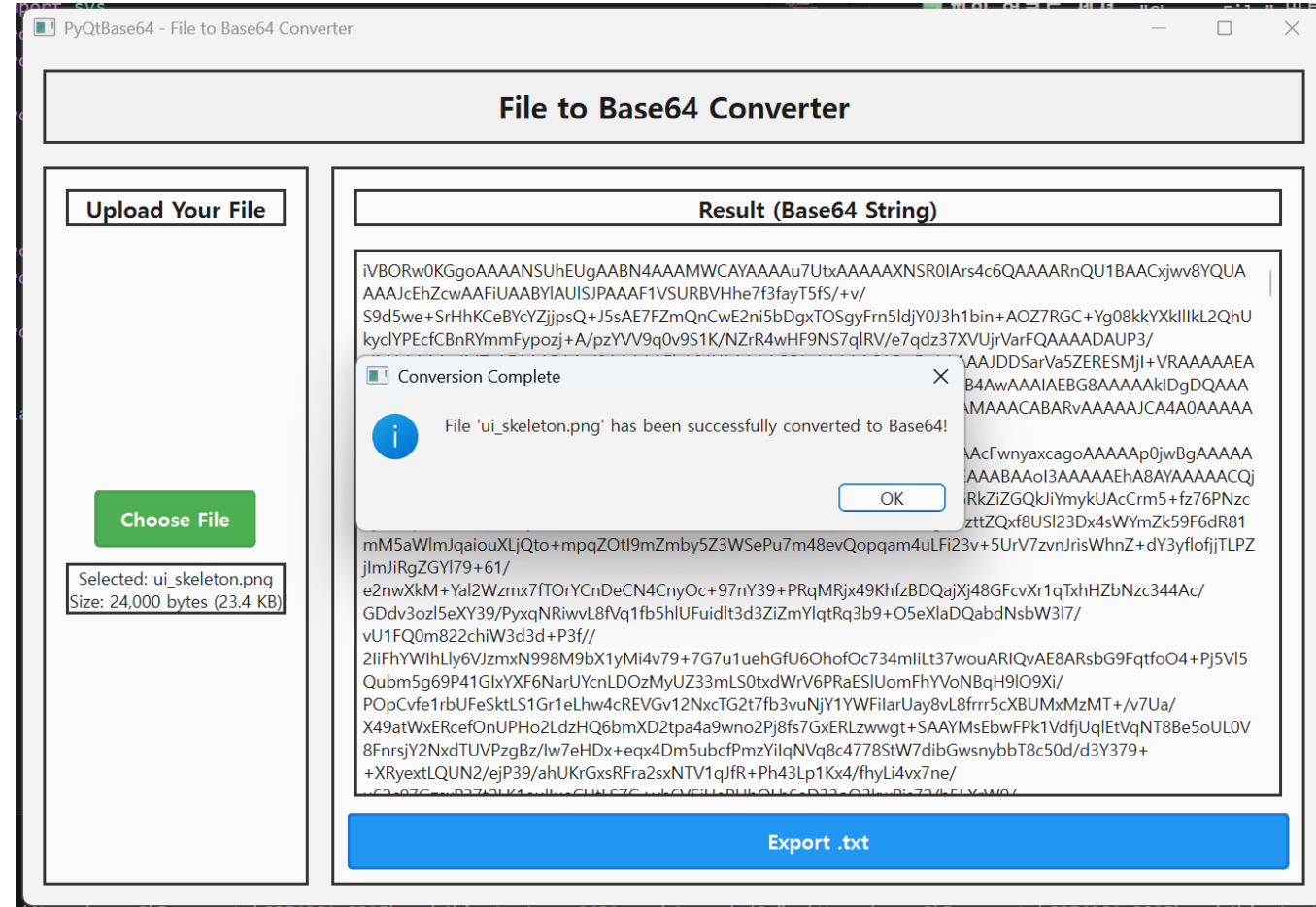


# UI 스켈레톤 입력

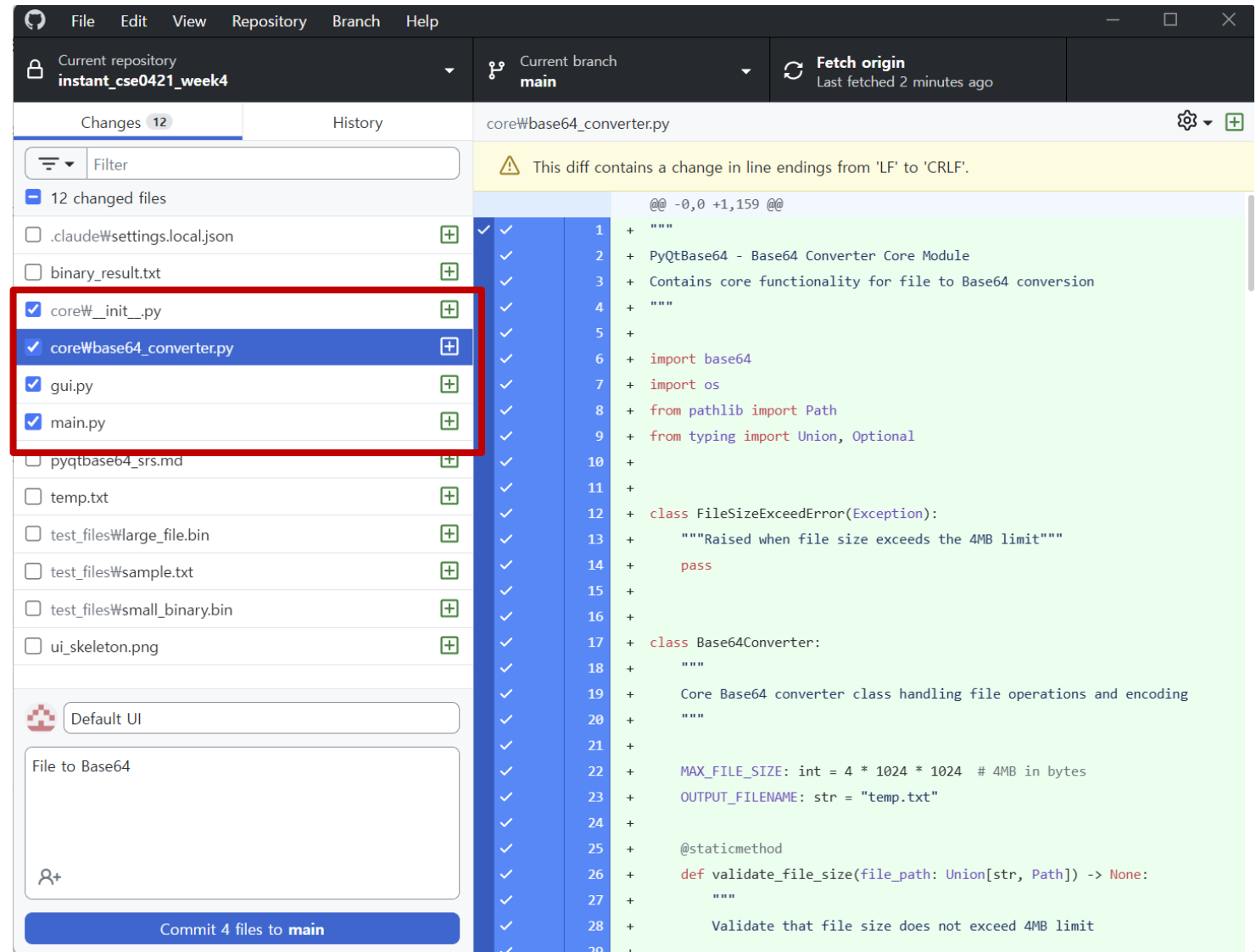
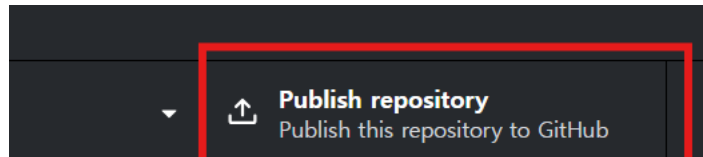
> 네 ./ui\_skeleton.png 파일을 참고하여 PyQt6 GUI 구현을 수행합니다.

- Read(ui\_skeleton.png)  
└ Read image (23.4KB)

- UI 스케치를 확인했습니다! 이제 PyQt6 GUI 구현을 시작하겠습니다.



# Git Push (Publish)



# Cont.

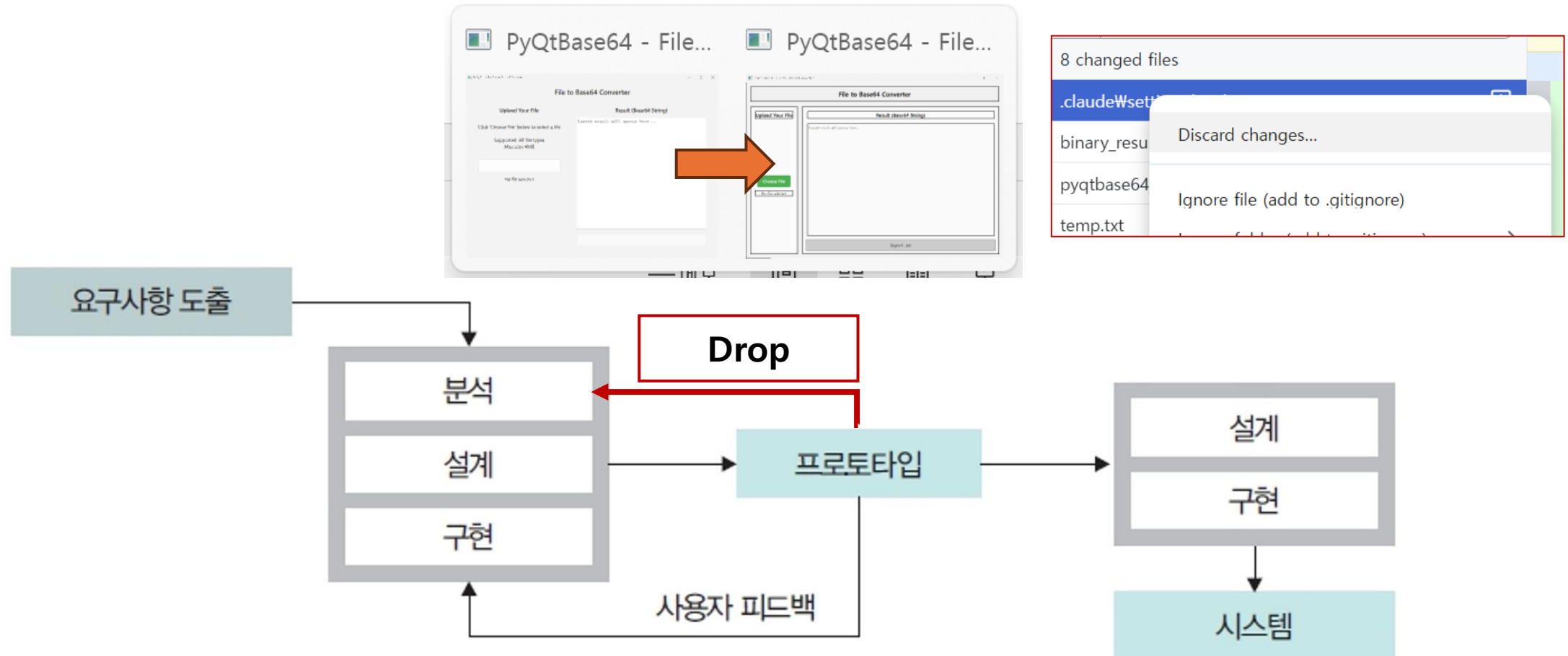


그림 3-5 프로토타입 모델에 의한 소프트웨어 개발 프로세스



# TODO

- [1] 내 File to Base64 Convert Tool 이미지 + 간단 평가 (문제점) – **1페이지 고정**
- [2] Base64 to File 을 함께 지원하는 도구로의 확장 방안을 담은 **SRS 문서\***