Final Report: Record My Brilliancy

컴퓨터공학부 2024-12818 이병도

1. Project Goal

이 프로젝트의 목표는 Chess.com에서 발생한 Brilliant Move(탁월한 수)의 공유 링크를 입력하면, 해당 수의 기보(PGN) 및 직전 보드 상태를 시각화하여 GitHub Pages 기반 블로그에 아카이빙하는 것이다. 블로그 메인 화면에는 날짜별로 brilliant move가 스트릭 형태로 시각화되며,탁월수 직전 보드의 이미지가 PNG로 만들어져 블로그에 업로드된다. 이 과정을 자동화하여, 나의 탁월수를 쉽게 저장하며 실력 향상 과정을 시각적으로 볼 수 있고, 이전 기록을 남길 수 있다.

2. Project Requirements and Detail Descriptions

입력: Chess.com에서 공유된 brilliant move URL

```
ibyeongdo@ByeongdoLees-MacBook-Pro RecordMyBrilliancy % ./bin/Record_My_Brilliancy
Enter: Brilliant URL:
```

Fig 1. 터미널에서 입력하는 경우

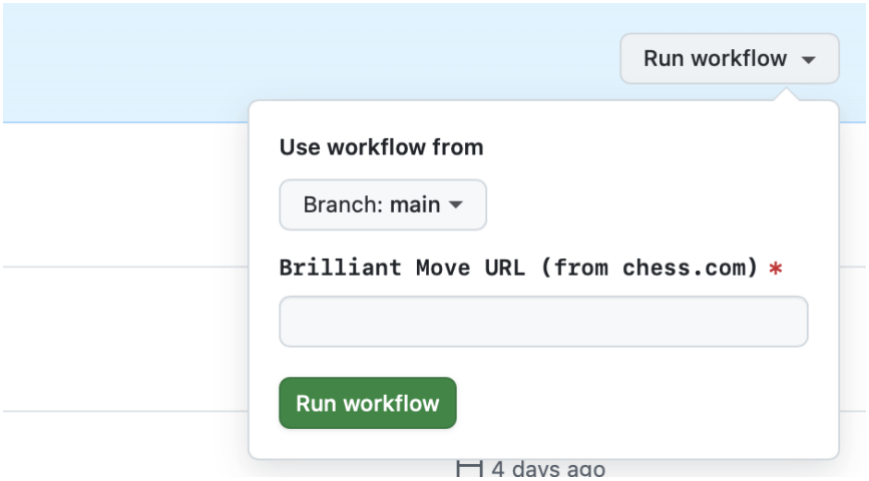

Fig 2. Github Actions로 입력하는 경우

출력: 마크다운 포스트

세부 기능:

- API 호출을 통한 게임 정보 획득
- PGN 파싱 및 brilliant move 직전 보드 상태 추출
- 텍스트 기반 보드를 이미지로 변환 (Python 사용)
- 마크다운 포스트 자동 생성 및 GitHub Pages 연동

- index.md 파일을 이용한 스트릭 UI 구성

3. Project Development Plan

보고서 제출 계획

- 5/20: 프로젝트 Goal 보고
- 5/27: 요구사항 및 세부기능 보고
- 6/3: 개발 계획 및 코드 구조 보고
- 6/10: 개발 진행 상황 보고
- 6/18: 최종 보고서 및 소스코드 제출

개발 계획

1) 프로젝트의 구체적인 동작 과정 설계
2) UML 만들기
3) 기능별 코드 개발 및 테스트
4) 자동화 및 중복 처리 구현
5) GitHub Pages 및 GitHub Actions 적용

4. Project Code Structure

구성 요소	설명
Record_My_Brilliancy.cpp	메인 실행 파일. 전체 흐름을 제어하며 사용자 입력을 받아 하위 모듈 호출
ChessFetcher	C++ 클래스. URL로부터 JSON 게임 데이터를 요청하고 필요한 정보 추출
MoveDecoder	C++ 클래스. moveList 형식의 압축 이동 정보를 디코딩하여 실제 체스 움직임 추출
ChessBoard	C++ 클래스. MoveDecoder를 이용해 내부 체스판에서 게임을 구현
PostManager	C++ 클래스. _posts/ 디렉토리에 마크다운 파일을 생성하고 index.md를 업데이트
GitManager	C++ 클래스. git add/commit/push를 자동 실행하여 리포지토리를 갱신
generate_streak.cpp	C++ 실행 파일. _posts/ 폴더를 분석해 streak 정보를 생성하고 _includes/streak.html을 갱신
txt_to_png.py	Python 스크립트. 텍스트 형태의 체스판 상태를 이미지(png)로 변환

_posts/	brilliant-YYYY-MM-DD.md 형태의 마크다운 파일을 저장하는 폴더
images/	PNG가 되기 전 .txt 파일과 최종 .png 이미지 파일을 저장하는 폴더
index.md	블로그 메인 페이지 역할을 하는 파일. brilliant move 목록과 streak가 표시됨
_includes/streak.html	generate_streak.cpp에서 자동 생성되며 index.md에서 streak UI로 포함되는 HTML
Makefile	전체 프로젝트 빌드를 관리하는 make 명세 파일
.github/workflows/	GitHub Actions 자동화를 위한 워크플로우 YAML 파일이 위치하는 디렉토리

1. **UML**

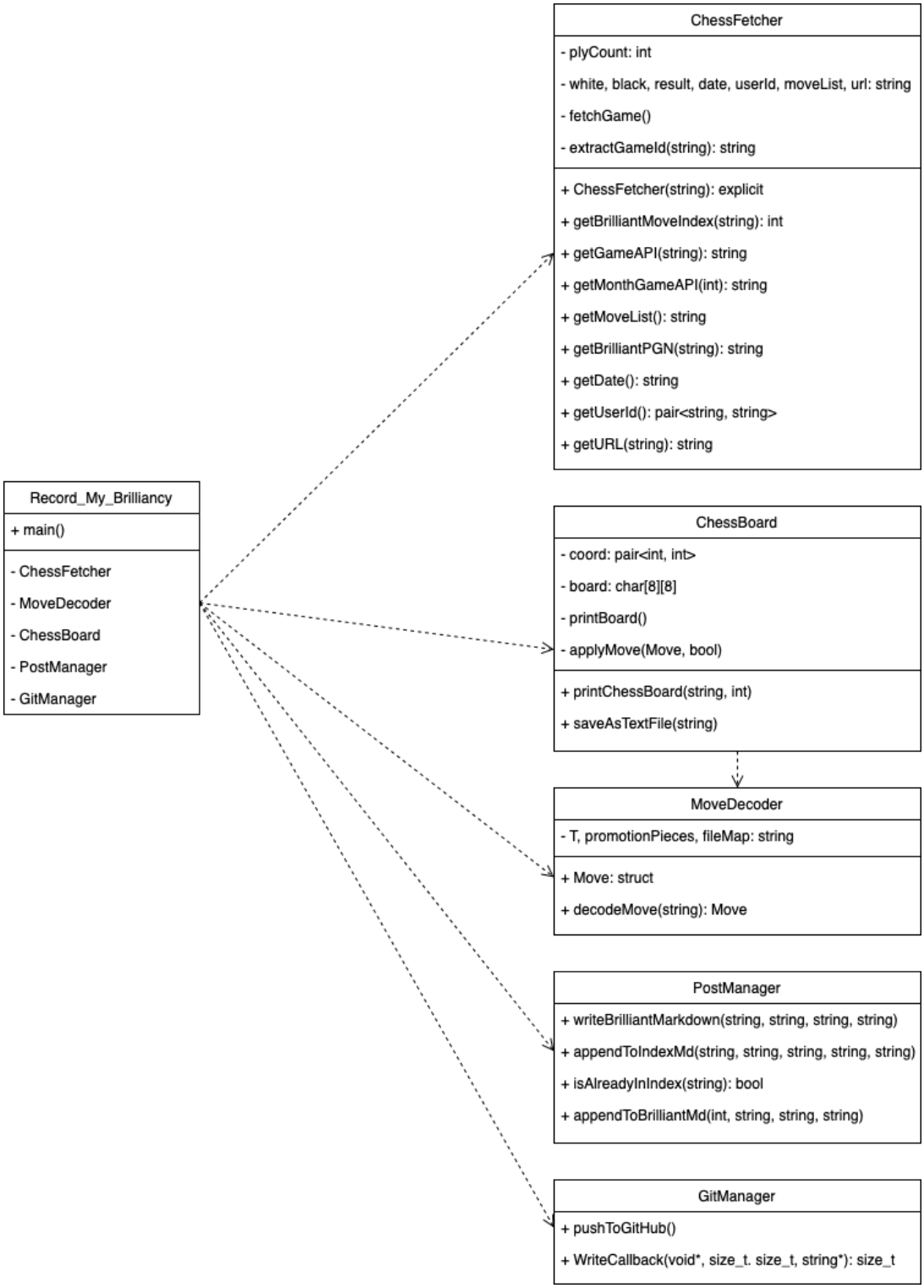

Fig 3. UML

2. Use Case Diagram

[User]

├─── (1) 프로그램 실행 (main)

├──── (2) brilliant move URL 입력
├──── (3) API 호출 및 JSON 파싱 (ChessFetcher)
├──── (4) moveList 해석 및 체스판 재구성 (MoveDecoder + ChessBoard)
├──── (5) 체스판 텍스트 저장 및 이미지 생성 (txt_to_png.py)
├──── (6) 마크다운 포스트 생성 및 index 갱신 (PostManager)
└──── (7) 자동 git push (GitManager)

3. CFG

main()
├──── 사용자 입력: Brilliant URL
├──── ChessFetcher::getGameAPI() → fetchGame()
├──── moveList = getMoveList()
├──── ChessBoard::printChessBoard()
├──── PGN 추출: getBrilliantPGN()
├──── 날짜/유저정보 파싱
├──── 중복 여부 확인: isAlreadyInIndex()
│ └──── 중복이면 종료
├──── 파일 경로 결정 및 txt 저장
├──── system("python3 txt_to_png.py ...")
├──── 마크다운 내용 구성
├──── writeBrilliantMarkdown()
├──── appendToIndexMd()
├──── appendToBrilliantMd()
└──── GitManager::pushToGitHub()

5. Actual Project Development Progress

1) 핵심 기능 구현

Brilliant move URL 파싱 기능을 구현하여 사용자가 복사한 체스 URL에서 게임 ID와 move index를 정확히 추출할 수 있도록 하였다. 또한 API 호출 및 JSON 파싱 로직을 통해 moveList, pgn, user, date 등을 추출하는 클래스를 구현하였다. (ChessFetcher)
moveList는 압축된 특수 문자열이었으며, 이를 디코드하는 로직을 Chess.com 포럼을 통해 확인하고 MoveDecoder클래스로 재구현하였다. (MoveDecoder, ChessBoard)

2) 자동화 흐름 통합

탁월수 직전 체스판 상태를 .txt로 저장한 뒤, Python 스크립트를 통해 .png 이미지로

자동 변환하도록 했다. 또한 블로그 포스트용 .md 파일을 자동 생성하고, index.md를 자동으로 갱신하는 기능을 구현하였다. (PostManager) 마지막으로 커맨드 한 줄로 git 커밋 및 푸시까지 가능한 자동화 시스템을 구현하였다. (GitManager)

3) 반복 및 예외 상황 처리

같은 날짜에 여러 개의 brilliant move가 있을 경우 중복 저장을 방지하기 위해 접미어(-2, -3 등)를 자동 부여하도록 구현하였다. 이전 포스트에 '다음 탁월수 보기' 링크를 추가하는 기능을 구현하여 사용자 탐색 편의성을 높였다. index.md 중복 여부를 검사하여 이미 기록된 brilliant move는 무시하도록 처리하였다.

4) 시각화 및 GitHub Pages 연동

_includes/streak.html을 자동 생성하는 프로그램을 작성하여 streak UI가 자동 반영되도록 구현하였다. GitHub Pages 블로그 구조에 맞도록 _posts/, index.md, images/ 디렉토리를 구성하고 자동으로 업데이트되도록 하였다.

5) 플랫폼 간 테스트 및 배포

Mac과 윈도우 PC에서 WSL 기반 Linux 환경을 테스트해 둘 모두에서 정상 작동하도록 테스트하였다. 바이너리 파일의 플랫폼 종속성 문제를 해결하기 위해 .gitignore 설정을 하였고, 종속 라이브러리(libcurl, pip, pillow) 설치 여부를 점검하고 문서화하여 누구나 쉽게 실행할 수 있도록 하였다.

6. Issues and Their Solutions

문제 1. Chess.com의 moveList가 압축된 2글자 문자열로 되어 있어 처음에는 어떤 방식으로 해석하는지 알 수 없었다.

인터넷에서 MoveList관련 정보를 찾다 Chess.com의 포럼의 댓글(https://www.chess.com/clubs/forum/view/official-chess-com-movelist-pgn-help)에서 디코드하는 로직이 담긴 JS코드를 발견해 이를 cpp의 MoveDecoder로 옮겼다. MoveList는 일반적인 PGN형식이 아니라 체스말이 원래 있던칸, 이동한 칸을 나타내는 식으로 구현되어 있었다.

문제 2. 일부 게임의 고유 ID가 숫자가 아닌 UUID 형태여서 API로 접근할 수 없었다.

ChessFetcher의 getBrilliantPGN함수에서 구현했는데, 일단 Chess.com의 API로 유저의 한달 전체 게임의 정보를 담은 JSON을 얻은 후 UUID와 일치하는 게임에서 MoveList를 얻어냈다.

문제 3. 같은 날짜에 여러 brilliant move가 있을 때 파일 중복

파일에 -2, -3 등 접미어를 자동으로 추가하여 중복되지 않게 저장하고, 만약 앞선 탁월수의 유저와 동일하면 기존 포스트에 '다음 탁월수 보기' 링크를 추가하도록 해 연속된 탁월수를 볼 수 있도록 했다.

문제 4. Mac에서 컴파일한 바이너리를 Windows(WLS)에서 실행할 수 없어 오류 발생.

bin 폴더는 .gitignore에 추가하여 git pull때마다 플랫폼별 충돌하는 문제를 방지했다. make clean 후 WSL에서 다시 make 실행하도록 매뉴얼을 작성했다.

문제 5. 이전에 파일 입출력 설계를 해본 적 없었다.

.md 파일 자동 생성, index.md 자동 갱신, 이미지 저장 및 중복 처리 등 파일 입출력 관련 설계에서 GPT의 로직 제안과 검토를 받았다. (curl사용 등)

7. Manual for Your Program

이 프로그램은 make 명령을 통해 컴파일되며, macOS 또는 Windows(WSL) 환경의 터미널에서 실행할 수 있다. 프로그램은 체스.com의 brilliant move URL을 입력 받아 해당 수의 기보와 체스판 이미지를 자동으로 생성하고, 마크다운 포스트와 streak 정보를 포함한 HTML 파일을 자동으로 갱신한다.

탁월수가 나온 게임에 내가 없으면 프로그램이 중지되도록 구현했기 때문에 Chess.com 닉네임이 ibottledo인 나의 탁월수만 업로드된다.

1) 실행 방법

1. 터미널에서 프로젝트 폴더로 이동한 뒤 다음 명령어를 입력한다.

```
make
./bin/Record_My_Brilliancy
```

2. 프로그램 실행 후 brilliant move URL을 입력하면 다음 파일들이 자동 생성된다.
   - images/2025-MM-DD-brilliant.png
   - _posts/2025-MM-DD-brilliant.md
   - _includes/streak.html
   - index.md (스트릭, 아카이브 갱신)

환경별 주의사항 및 해결방법

1. 다른 운영체제에서 컴파일된 바이너리가 호환되지 않는 경우

운영체제가 다른 환경(macOS → WSL 등)에서는 실행파일 포맷이 달라 실행되지 않을 수 있다. 이 경우 기존 바이너리를 삭제하고 다음과 같이 재컴파일한다.

```
rm bin/Record_My_Brilliancy
rm bin/generate_streak
make clean
make
```

2. 컴파일 시 curl/curl.h 관련 오류가 발생하는 경우

이는 libcurl 개발 헤더 파일이 설치되어 있지 않기 때문이다. 아래 명령어로 설치할 수 있다.

```
sudo apt update
sudo apt install libcurl4-openssl-dev
```

3. 이미지 파일이 생성되지 않는 경우

txt 파일을 PNG로 변환하는 과정에서 pillow가 설치되어 있지 않으면 이미지가 생성되지 않는다. 다음 명령어를 통해 pip 및 pillow를 설치한다.

```
sudo apt update
sudo apt install python3-pip
pip3 install pillow
```

2) GitHub Actions를 통한 실행 방법

워크플로우 파일로 /.github/workflows/brilliant.yml파일을 설정해 프로그램을 GitHub Actions와 연동하면 로컬 터미널에서 수동으로 실행하지 않아도,
GitHub 상에서 자동으로 brilliant move 데이터를 처리하고 블로그에 반영할 수 있다.

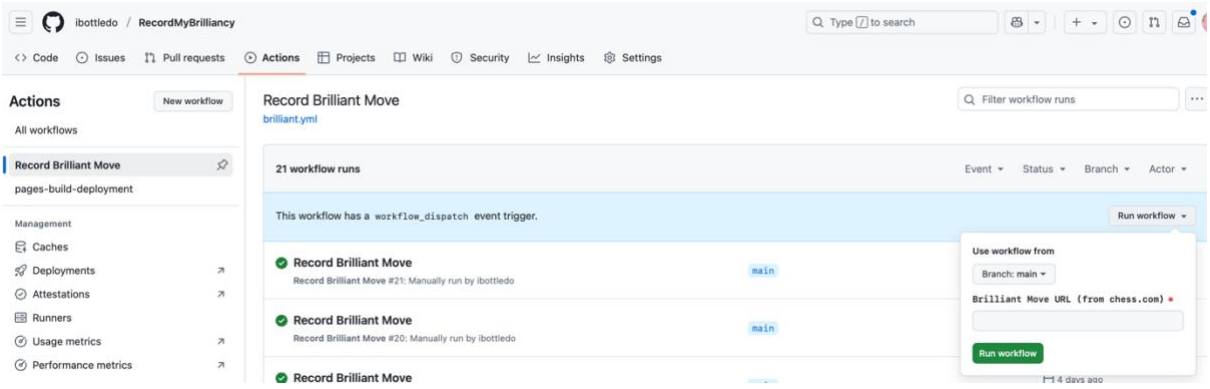

Fig 4. Github Actions탭의 Record Brilliant Move에서 Run workflow클릭

위 사진처럼 링크를 넣고 Run workflow버튼을 누르면 내부에서 코드가 작동해 블로그 문서가 작성되고 업데이트된다.

8. Program Results

나의 깃허브 프로젝트 (https://github.com/ibottledo/RecordMyBrilliancy)의 readme에 있는 링크(https://ibottledo.github.io/RecordMyBrilliancy/)를 통해 블로그에 접속할 수 있다.

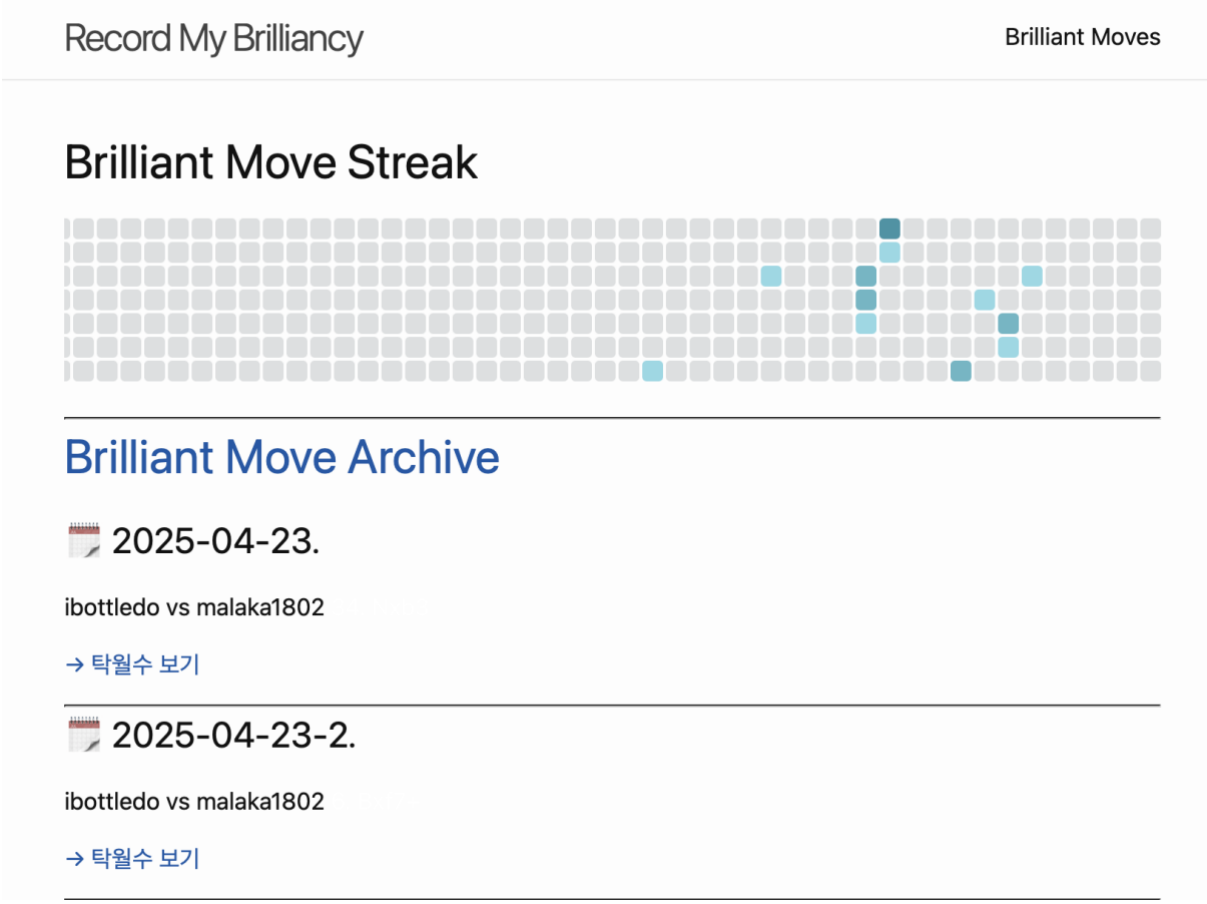

Fig 5. 블로그 메인화면 사진

좌우 스크롤이 가능한 탁월수 스트릭이 있고, 아래에 날짜와 상대의 정보를 담은 아카이브 목록이 있다. 여기서 파란색 [-> 탁월수 보기]를 누르면 각 탁월수에 대한 개별 블로그 문서로 넘어간다. 또는 스트릭의 파란색 정사각형을 누르면 해당 날짜의 첫번째 탁월수에 대한 블로그 문서로 넘어갈 수 있다.

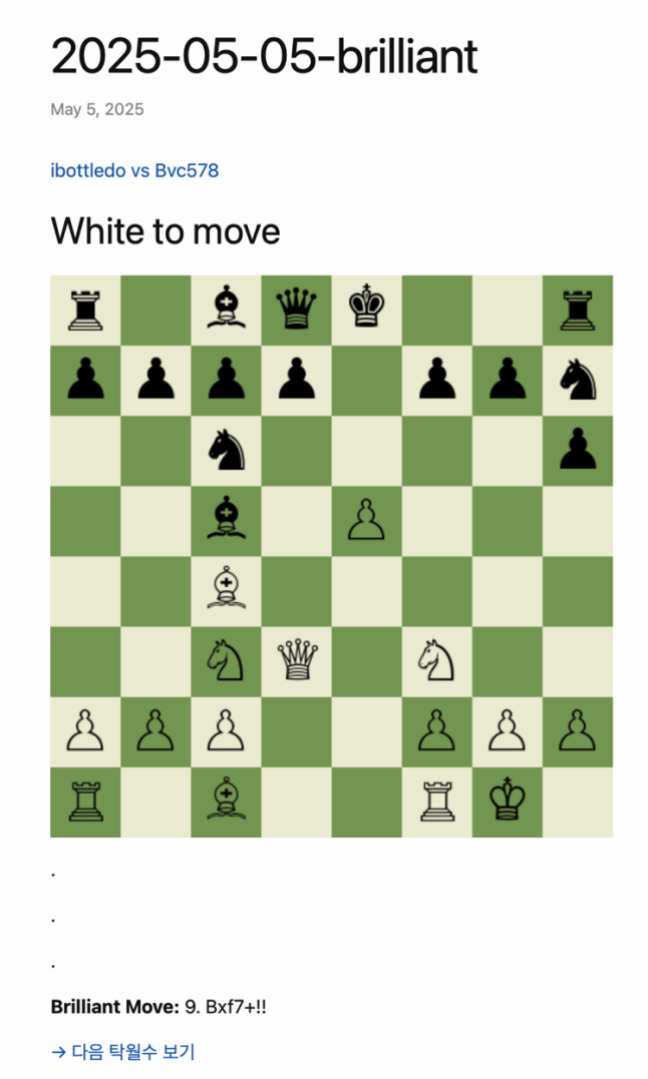

Fig 6. 탁월수의 개별 블로그 문서

여기서 이 게임의 다른 탁월수가 있다면 왼쪽 아래의 [-> 다음 탁월수 보기]를 클릭해 다음 탁월수 문서로 넘어갈 수 있다. API에서 얻어낸 MoveList에서 탁월수 직전의 보드 상황을 txt파일로 재현하고, txt파일을 png이미지로 만드는 과정에서 파이썬을 사용했다.

White to move, Black to move로 흰색 말을 움직이는 탁월수가 있다는 것을 알릴 수 있다. 그 위에 파란 [ibottledo vs Bvc578]를 누르면 Chess.com 분석 탭으로 이동한다.

8. Conclusion

본 프로젝트는 단순한 데이터 수집을 넘어, 웹 자동화, 이미지 처리, GitHub Pages 블로그 연동까지 다양한 기술을 통합해 사용한 첫번째 경험이었다. 특히 체스라는 평소에 관심있는 주제를 정해 탁월수를 시각적으로 아카이빙할 수 있는 프로젝트를 진행해 앞으로 체스 실력 향상 과정을 재미있고 의미 있게 기록할 수 있게 되었다.

+추신

혹시 조교님이 직접 테스트하려고 하신다면 최근 링크 몇개를 아래에 달아둘테니 이 링크로 실행하시면 됩니다.

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/138555859748?move=22&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/138668037848?move=39&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/42ebe382-3eea-11f0-9421-6ea45201000f?move=46&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/e7c16e47-4075-11f0-b08d-535a6d01000f?move=12&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/139268296548?move=26&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/139312156618?move=41&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/139543896472?move=30&tab=review

이 탁월한 수를 보십시오: https://www.chess.com/analysis/game/live/139602837530?move=40&tab=review