

1 GPT 활용

2 → ●● 코드 리뷰 도우미

3 [] 미니 프로젝트 *

인공지능소프트웨어학과

presenter : = 김현종 ; / 20192746

CONTENTS



동양미래대학교

1

PART 01

- ① 프로젝트 배경 및 문제 정의
- ② 프로젝트 목표 및 솔루션 개요

2

3

PART 02

- ③ 솔루션 구조 및 주요 기능
- ④ 코드 구성 및 주요 함수 설명

4

5

PART 03

- ⑤ 프롬프트 설계 및 LLM 응답 예시
- ⑥ 실제 코드 예제 및 리뷰 결과 시연

6

PART 1 * 프로젝트 배경 및 문제 정의

동양미래대학교

1

코드 리뷰의 중요성

- 코드 리뷰는 소프트웨어 개발 과정에서 품질을 보장하는 핵심 단계
- 버그의 사전 발견, 코드의 성능과 가독성의 개선
- 생산성 향상, 안정성 보장

기존 코드 리뷰 문제점

- 시간과 비용 소모
- 초보 개발자의 학습 기회 부족
- 리뷰 품질의 일관성 부족

프로젝트 목표

- 코드 리뷰 자동화
- 초보 개발자를 위한 실질적 학습 지원
- 리뷰 품질 개선과 일관성 유지

2

3

4

5

6

PART 1 * 프로젝트 목표



동양미래대학교

1

자동화된
코드 리뷰 제공



코드의 오류, 스타일, 성능, 리팩토링 등 자동 피드백

2

LLM 기반 자동 리뷰로 일관성과 효율성 향상

3

실질적인
학습 도우미 역할



코드의 문제점을 인식하고 개선하는 학습 기회 제공

4

이해하기 쉬운 피드백으로 코드 작성능력 향상

5

코드 품질 개선 및
유지보수성 향상



코드의 가독성과 효율성 상승 및 유지보수성 향상에 기여

6

코드 리뷰 시간 단축, 프로젝트 전체 생산성 향상

PART 1 * 솔루션 개요

동양미래대학교

1

OpenAI API 기반
LLM 활용



GPT 모델을 사용하여 입력된 코드에 대한 자동 리뷰를 제공

2

코드의 구조와 의미를 분석, 다양한 측면에서 피드백을 제공

3

다각적 리뷰 제공



코드 오류, 스타일 가이드 준수, 성능 최적화, 리팩토링 제안

4

코드의 질을 여러 측면에서 개선할 수 있도록 지원

5

사용자 피드백 기반
품질 개선



사용자 피드백을 통해 모델이 제공하는 리뷰 품질을 점검

6

피드백 루프를 통해 LLM의 리뷰 성능을 지속적으로 개선

PART 2* 프로젝트 기능

동양미래대학교

1

OpenAI API 기반 코드 분석

사용자 피드백 수집 및 품질 개선

2



3

OpenAI API의 GPT 모델을
사용하여 코드를 분석

사용자가 코드 리뷰 결과에 대한 피드백을
제출할 수 있도록 하여 리뷰 품질을
점검하고 개선

4

5

리팩토링과 테스트 케이스 제안을 포함한
포괄적인 피드백을 제공

리뷰 정확성과 만족도를 지속적으로 향상

6



1

2

주요기능

3

코드 스타일 및 컨벤션 체크

성능 최적화 및 리팩토링 제안

케이스 생성 및 엣지 케이스 제안

4

- 코드가 특정 스타일 가이드(예: Python의 PEP 8)에 맞는지 확인하고, 스타일 개선 사항을 제안합니다.

- 코드의 성능을 분석하여 시간과 공간 복잡도를 줄일 수 있는 최적화 방법을 제안합니다.

- 사용자가 작성한 코드의 기능을 확인할 수 있도록 자동으로 테스트 케이스를 생성합니다.

5

- 코드 가독성과 일관성을 높여 팀 프로젝트 코드 품질 유지에 기여합니다.

- 필요한 중복 코드를 제거하거나, 효율적인 알고리즘으로 리팩토링할 수 있는 구체적인 방법을 제공합니다.

- 일반적인 입력 외에도 엣지 케이스를 제안하여 코드의 견고성을 검증할 수 있도록 지원합니다.

6

```
[ ] 1 import openai
2
3 # API 키 설정
4 KEY = 'sk-proj-bZ0nTpZ4hjKa6v9Gasqip2Neu82IPM_IAI1htojE0TztBy_wJsR5du8_PT9_08nhTuj i-MZ1FzT3B1bkFJXx9t8A3Ql vv_f6Y46Adxpt80_g000ce7Y9RmyXChJtdgq5-Rl z0SrADdxz09MHIDM_B6Rf f i oA'
5 openai.api_key = KEY
6
```

```
[ ] 1 def code_review_helper(code_snippet):
2     """
3     사용자 코드에 대한 다각적인 분석과 최적화 제안을 제공합니다.
4     - 코드 오류, 스타일, 성능, 리팩토링, 테스트 케이스 생성, 복잡도 분석 포함.
5     """
6     # 사용자 코드에 대한 설명과 최적화 힌트를 요청하는 프롬프트 생성
7     messages = [
8         {"role": "system", "content": ""}
9         You are a coding assistant specializing in code review.
10        You will provide feedback categorized into 'Errors', 'Style and Convention', 'Performance Optimization',
11        'Refactoring Suggestions', and 'Test Cases'.
12        Also, provide a time and space complexity analysis for the given code.
13        Please reply into korean
14        """},
15        {"role": "user", "content": f"Here's the code snippet:\n{code_snippet}\nPlease provide an analysis."}
16    ]
17
18    # OpenAI API를 사용하여 응답 생성
19    response = openai.ChatCompletion.create(
20        model="gpt-3.5-turbo",
21        messages=messages
22    )
23
24    # 응답 메시지 반환
25    return response.choices[0].message.content
26
```

```
[ ] 1 def feedback_collection(response):
2     """
3     사용자 피드백 수집하여 개선할 수 있는 방안 확인.
4     """
5     user_feedback = input("\nDo you find this feedback helpful? (yes/no): ")
6     if user_feedback.lower() == "no":
7         suggestions = input("Please provide suggestions for improvement: ")
8         print("\nThank you! Your feedback will be used to improve the review quality.")
9     else:
10        print("\nGlad to know the feedback was helpful!")
11
```



```
[ ] 1 user_code = """
2 def find_max(numbers):
3     max_number = 0 # 올바른지 않은 초기값
4     for number in numbers:
5         if number > max_number:
6             max_number = number
7     return max_number
8 """
9
```

```
[ ] 1 # 코드 리뷰 도우미 호출
2 review_result = code_review_helper(user_code)
3
4 # 결과 출력
5 print("Code Review Result:")
6 print(review_result)
7
8 # 사용자 피드백 수집
9 feedback_collection(review_result)
10
```



Code Review Result:

이 코드의 분석은 다음과 같습니다:

- 오류:
 - max_number의 초기값이 0으로 설정되어 있어, 주어진 숫자 리스트에 음수가 포함되어 있을 때 올바른 최대값을 찾지 못할 수 있습니다. 최소값으로 초기화하는 것이 더 안전합니다.
- 스타일과 규칙:
 - 함수와 변수의 명명은 명확하고 이해하기 쉽게 잘되었습니다.
 - 코드의 들여쓰기가 일관되고 가독성이 좋습니다.
- 성능 최적화:
 - 주어진 코드는 이미 최적화되어 있으며, 주어진 숫자 리스트의 길이에 선형적으로 실행됩니다.
- 리팩터링 제안:
 - 초기값을 최소값으로 설정하는 것이 더 안전하며, 이 경우 max_number = float('-inf')와 같이 할당할 수 있습니다.
- 테스트 케이스:
 - 함수에 다양한 종류의 숫자 리스트를 사용하여 테스트하는 것이 좋습니다. 음수, 중복값, 빈 리스트 등에 대한 테스트가 필요합니다.

시간 복잡도: $O(n)$ - 주어진 숫자 리스트의 모든 요소를 한 번씩만 확인합니다.

공간 복잡도: $O(1)$ - 고정된 공간을 사용하므로 입력 크기에 관계없이 일정한 양의 공간을 사용합니다.

Do you find this feedback helpful? (yes/no): yes

Glad to know the feedback was helpful!



1

2

3

4

5

6



프로젝트 관련 궁금한 점을
자유롭게 질문해주세요!



Ask me Questions