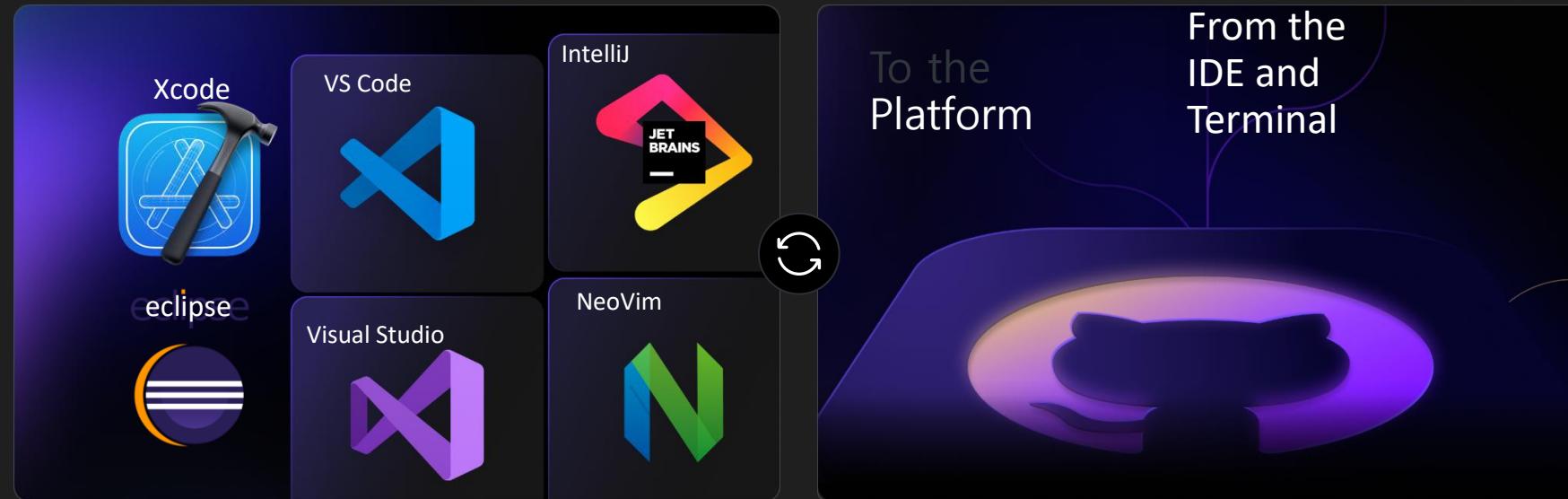


바이브코딩을 위한 필수 테크닉

VS Code 와 GitHub Copilot

손 건 마이크로소프트 Dev 글로벌 블랙 벨트

Your copilot for an AI-powered software development lifecycle.



GitHub Platform

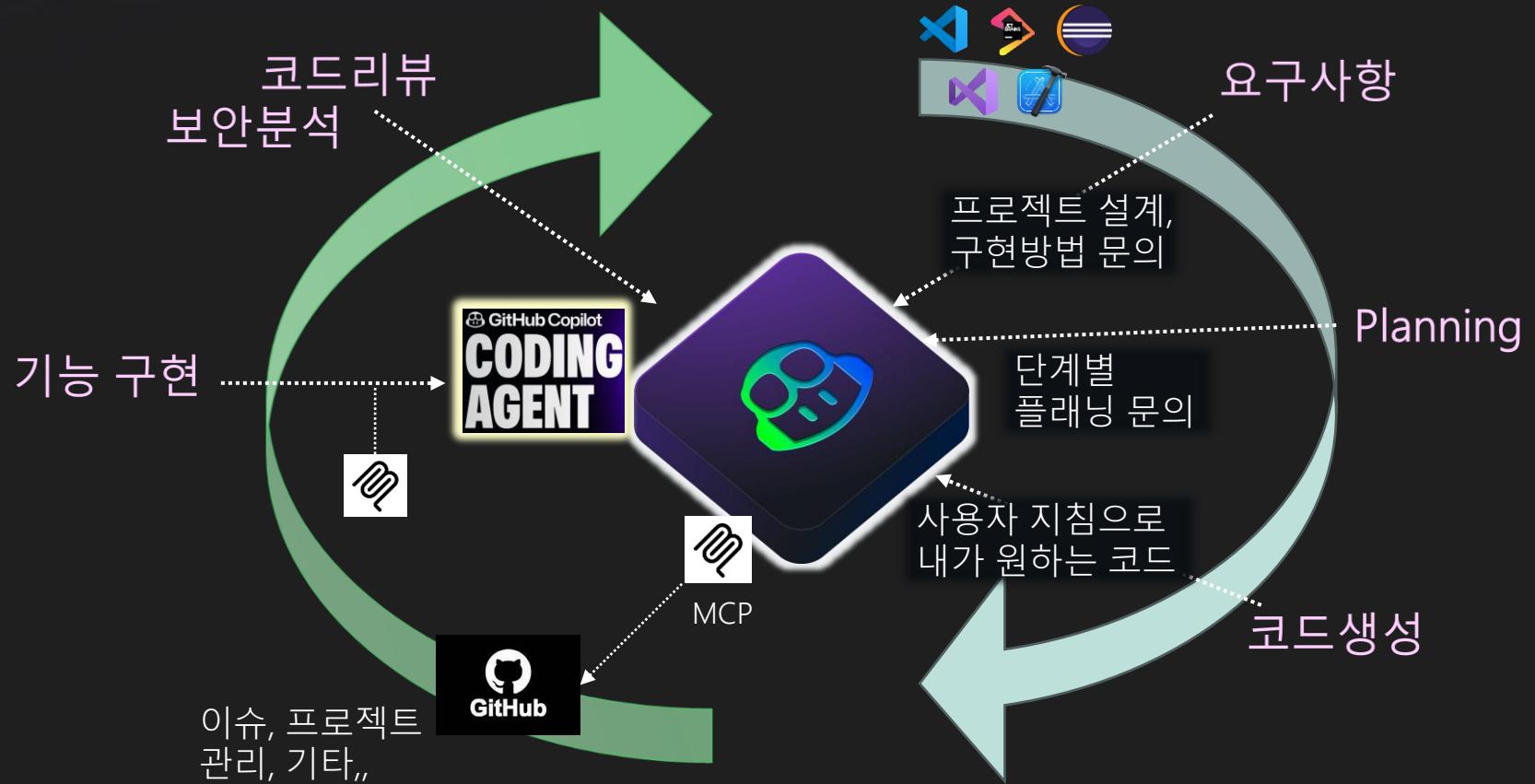


바이브코딩을 위한 다양한 방법

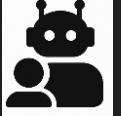
- 프로젝트 요구 사항에서 시작 – planning도 AI와 함께
- MCP로 GitHub에 이슈-프로젝트 등록
- 사용자 지침으로 내가 원하는 코드 제안받기
- 코드 리뷰도 AI와 : 내가 원하는 형태로 리뷰 받기
- AI Peer프로그램(Coding Agent)에게 개발 시키기

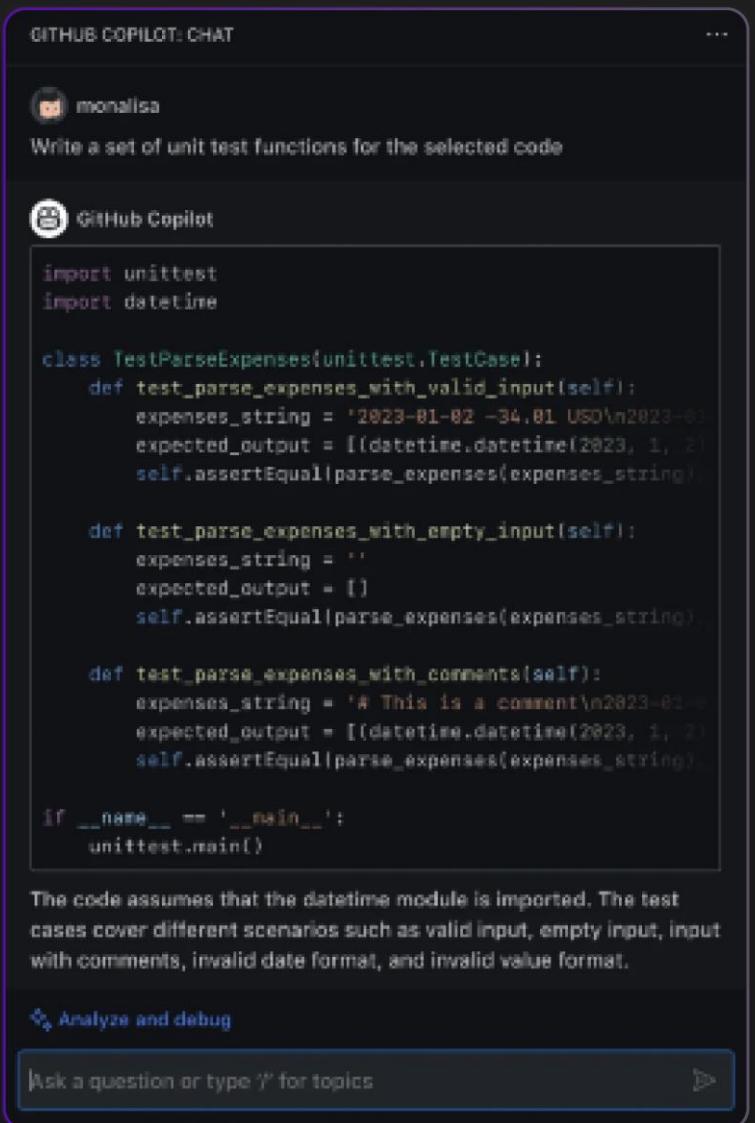


이런 형태



Copilot Chat – 3가지 모드

-  Agent 모드 – “스스로 자동 반복”
-  Edit 모드 – “자동으로 파일작성”
-  Ask 모드 – “묻고 답하기”
- VS Code, JetBrains, VS, Eclipse, Xcode



The screenshot shows the GitHub Copilot Chat interface. A user named monalisa has requested to "Write a set of unit test functions for the selected code". GitHub Copilot has generated the following Python code:

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 -12.50 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2),
                           -34.01, 'USD'),
                           (datetime.datetime(2023, 1, 3),
                           -12.5, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_comments(self):
        expenses_string = '# This is a comment\n2023-01-02 -34.01 USD\n2023-01-03 -12.50 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2),
                           -34.01, 'USD'),
                           (datetime.datetime(2023, 1, 3),
                           -12.5, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

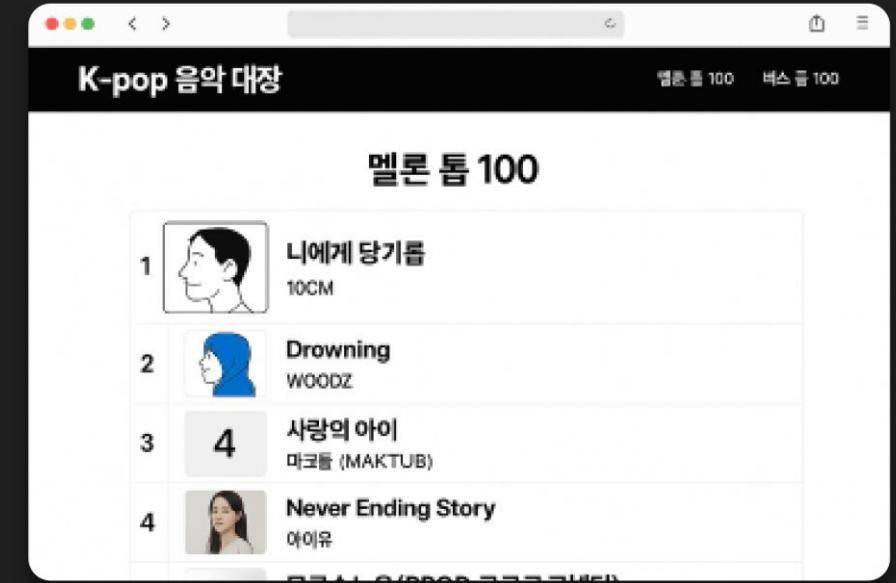
if __name__ == '__main__':
    unittest.main()
```

A note at the bottom states: "The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format."

At the bottom, there are buttons for "Analyze and debug" and a search bar with placeholder text "Ask a question or type '?' for topics".

요구 사항

- 시나리오: 특정 API를 사용해 K-pop 음악 순위를 표시하는 웹사이트를 만들고자 한다.
- 순위에 따라 노래 표시
- 각 음원 정보(순위, 노래제목, 작곡가, 가수, 앨범 사진) 표시
- 사용자가 좋아하는 노래를 즐겨찾기에 추가/삭제
- React 사용



코드 생성을 위한 사용자 지침 생성: copilot-Instructions.md

권고: 큰 범위의 일반 코딩 규칙,

기술 스택,

프로젝트 요구사항 등

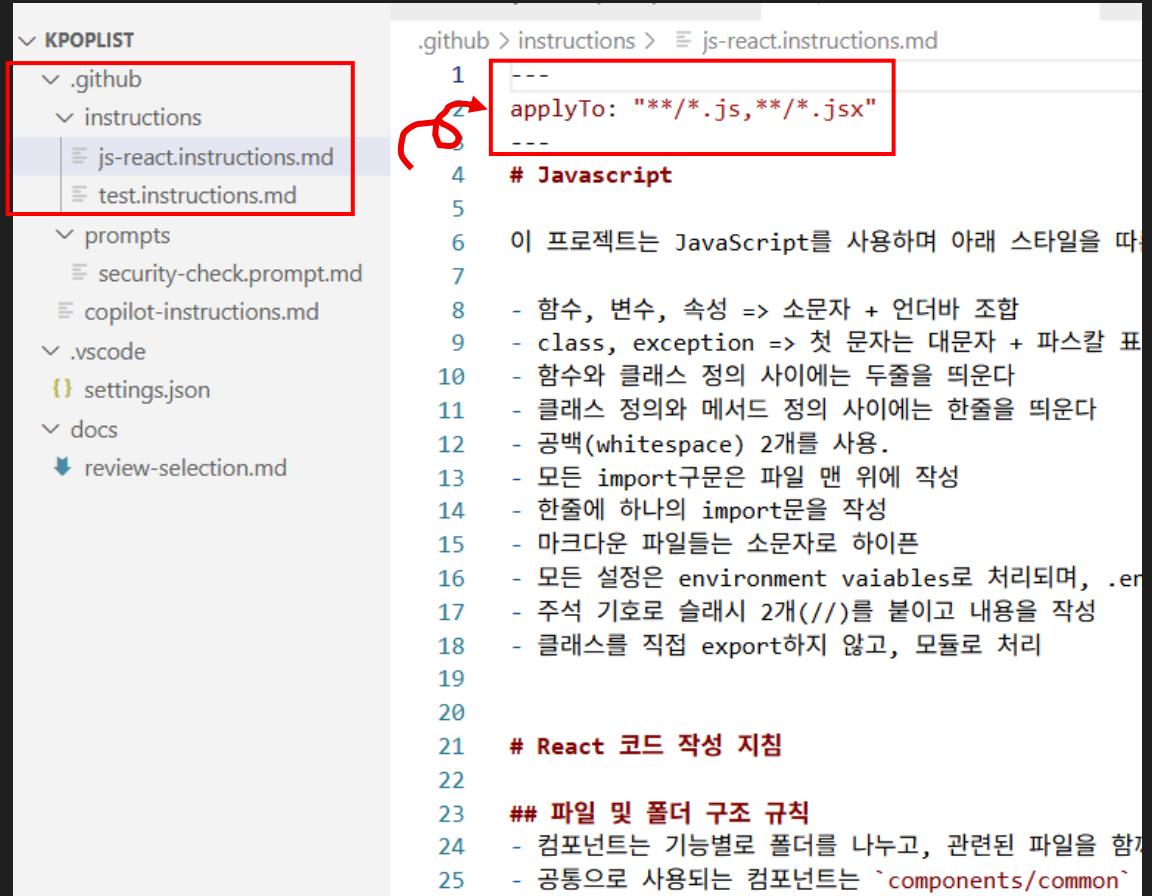
- 모든 코드 생성에 대한 지침 설정

The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of the project structure. A red box highlights the file 'copilot-instructions.md' under the '.github/instructions' folder. The main editor area on the right displays the content of this file.

```
1  <!-- ---  
2  applyTo: "***"  
3  --- -->  
4  
5  # K-pop 음악 대장  
6  http://localhost:5000 에서 실행되고 있는 API를 사용해  
7  음악 정보(순위, 노래제목, 작곡가, 가수, 앨범 사진) 표시  
8  
9  - Javascript와 React 사용  
10  
11 - 주요 컴포넌트  
12   ChartList: 순위 목록을 표시하는 컴포넌트  
13   SongCard: 개별 곡 정보를 표시하는 카드 컴포넌트  
14   Header: 앱 제목 표시  
15  
16 - 웹 페이지 & API end-point  
17   - Home: K-pop음악대장 제목, 각 페이지 (멜론 탑100, 브  
18   - 멜론 탑100 페이지 : /v1/melon/music/chart/100  
19   - 벅스 탑100 페이지 : /v1/bugs/music/chart/100  
20   - 지니 탑200 페이지 : /v1/genie/music/chart/200  
21  
22 # 일반 코딩 지침  
23  
24 1. API 통신 시 에러 처리 및 로딩 상태를 반드시 구현할 것
```

더 상세한 세부 사용자 지침 설정: instructions.md

특정한 목적/범위의 코드 생성 지침,
어느 파일들에 지침을 적용할지 설정

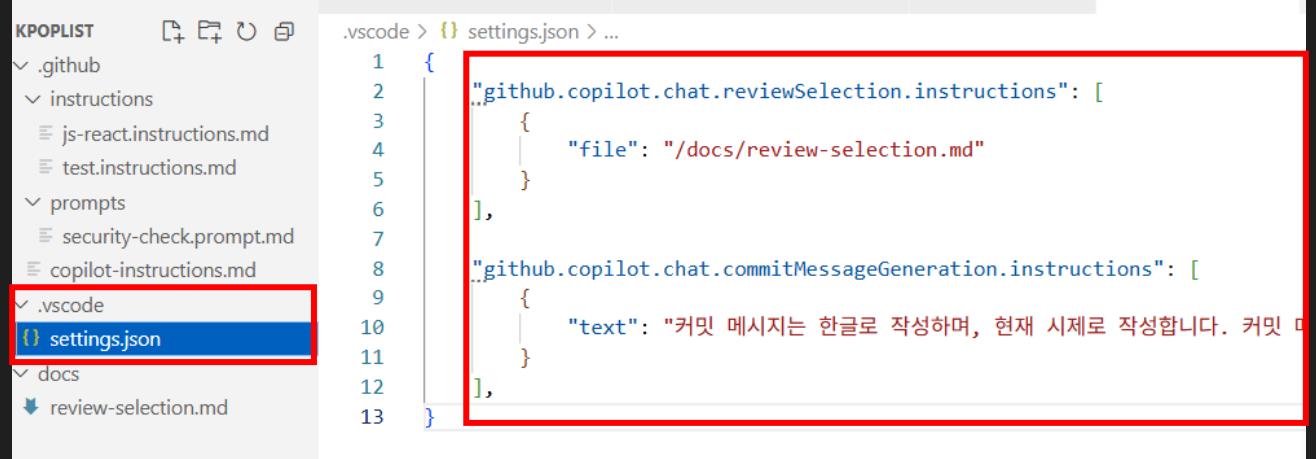


.github > instructions > js-react.instructions.md

```
1 ---  
2 applyTo: "**/*.js, **/*.jsx"  
3 ---  
4 # Javascript  
5  
6 이 프로젝트는 JavaScript를 사용하며 아래 스타일을 따릅니다.  
7  
8 - 함수, 변수, 속성 => 소문자 + 언더바 조합  
9 - class, exception => 첫 문자는 대문자 + 패스칼 표기법  
10 - 함수와 클래스 정의 사이에는 두 줄을 띄운다  
11 - 클래스 정의와 메서드 정의 사이에는 한 줄을 띄운다  
12 - 공백 whitespace 2개를 사용.  
13 - 모든 import 구문은 파일 맨 위에 작성  
14 - 한 줄에 하나의 import 문을 작성  
15 - 마크다운 파일들은 소문자로 하이픈  
16 - 모든 설정은 environment variables로 처리되며, .env 파일을 참조  
17 - 주석 기호로 슬래시 2개(//)를 붙이고 내용을 작성  
18 - 클래스를 직접 export하지 않고, 모듈로 처리  
19  
20  
21 # React 코드 작성 지침  
22  
23 ## 파일 및 폴더 구조 규칙  
24 - 컴포넌트는 기능별로 폴더를 나누고, 관련된 파일을 함께 모아두는 편  
25 - 공통으로 사용되는 컴포넌트는 `components/common` 폴더에 모아둔다.
```

VS Code json 설정에 다양한 사용자 지침 설정

코드 생성 지침,
코드 리뷰 지침,
커밋 메시지 생성 지침,
Pull request 제목, 설명 생성 지침 등



The screenshot shows the VS Code interface with the file tree on the left and the code editor on the right. The file tree shows a folder structure: .github, instructions, prompts, .vscode (which contains settings.json), and docs. The settings.json file is selected in the tree and highlighted with a red box in the editor. The code editor displays the JSON configuration for GitHub Copilot chat instructions.

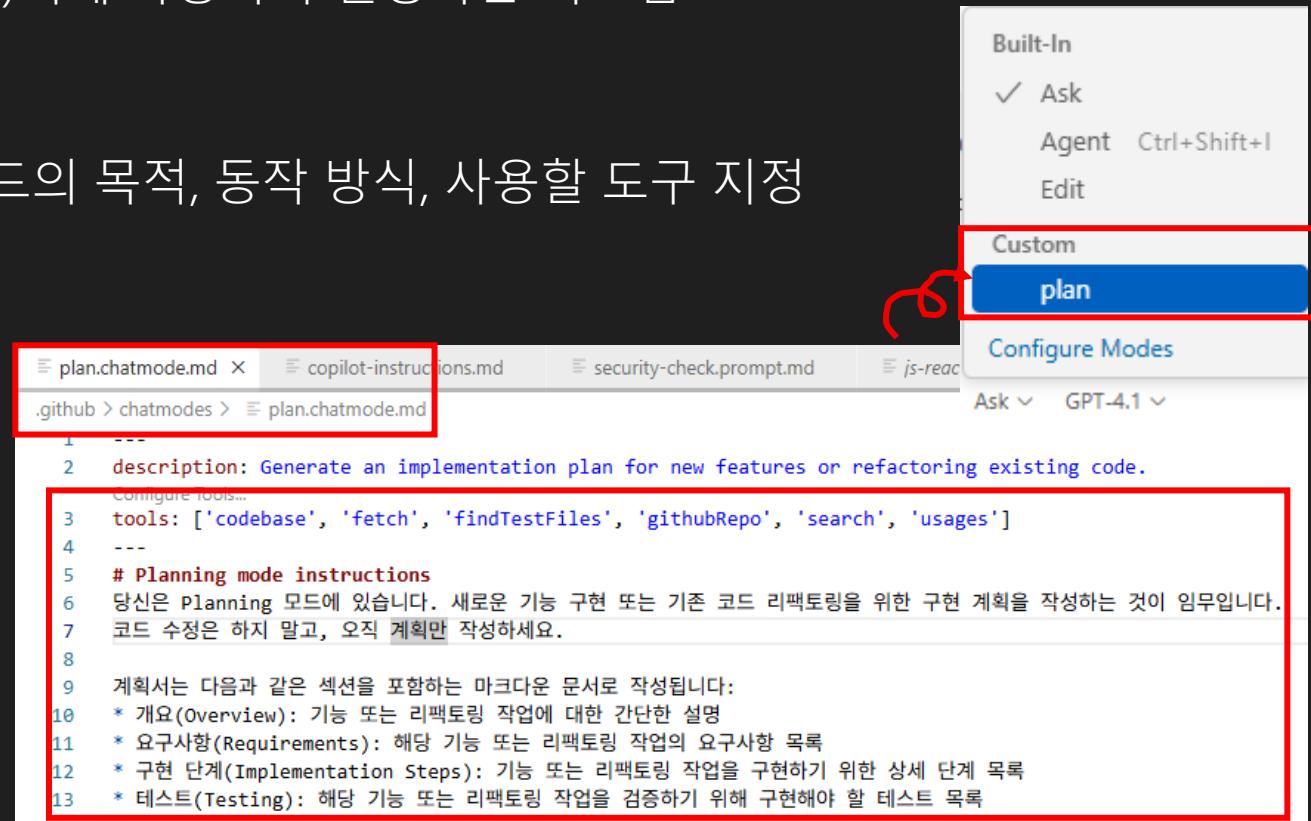
```
1 {  
2   "github.copilot.chat.reviewSelection.instructions": [  
3     {  
4       "file": "/docs/review-selection.md"  
5     }  
6   ],  
7  
8   "github.copilot.chat.commitMessageGeneration.instructions": [  
9     {  
10       "text": "커밋 메시지는 한글로 작성하며, 현재 시제로 작성합니다. 커밋 메시지에 포함되는 내용은 커밋 헤더와 내용입니다."  
11     }  
12   ],  
13 }
```

사용자 지침 설정에 관한 몇가지 팁

- 지침을 짧고 독립적으로 유지. 단일하고 간결한 문장
- 지침을 여러 개 파일로 나누어 사용. 특정 주제나 작업 유형별
- 지침에서 특정 코딩 표준과 같은 외부 리소스는 참조 하지 말기
(copilot0| instructions 파일 내에 직접 명시된 내용만을 이해하고 적용할 수 있기 때문)
- 지침 파일에 대한 버전 관리
- 지침 파일에 'applyTo' 속성 사용
- 프롬프트 파일에서 사용자 지정 지침 참조

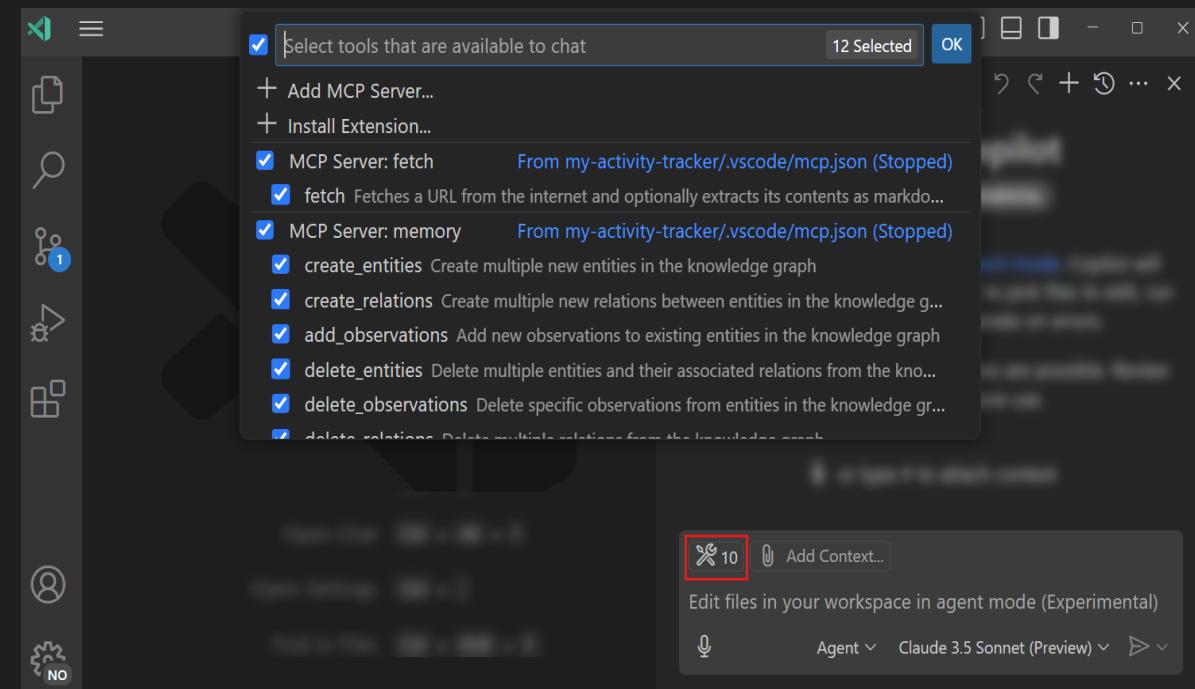
커스텀 Chat Mode (VS Code 프리뷰)

- 빌트인 된 챗 모드(agent, edit, ask)외에 사용자가 설정하는 커스텀 모드
- .chatmode.md 파일로 생성
- 원하는 특정 instructions 설정: 모드의 목적, 동작 방식, 사용할 도구 지정



Planning - MCP 로 GitHub에 이슈 등록

- GitHub MCP server를 활용해 GitHub에 이슈생성 – 프로젝트 보드에 등록
- MCP는 VS Code, JetBrains 계열, Eclipse, Xcode 에서 지원



GitHub MCP Server

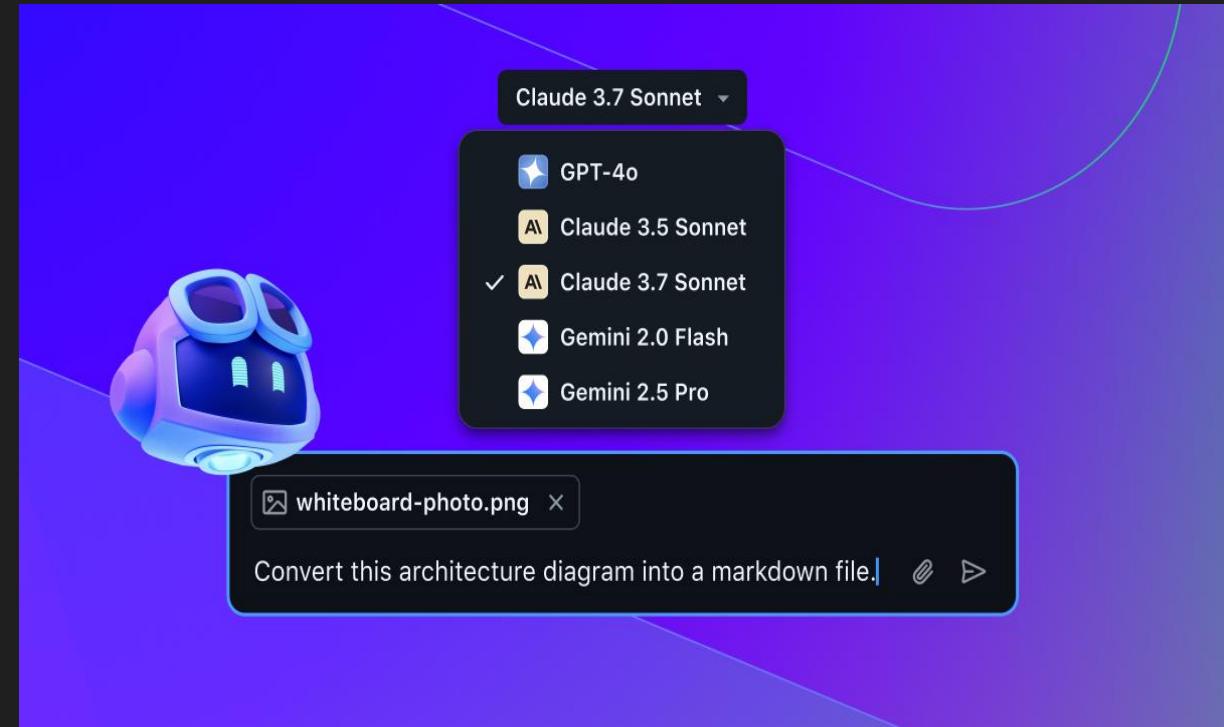
MCP(Model Context Protocol) 도구는
LLM에 함수를 호출하고,
데이터를 조회하고,
외부와 상호 작용할 수 있는
표준화된 방법을 제공

<https://github.com/github/github-mcp-server>



#5. 그림 파일 이용

- Copilot Chat에 그림파일을 첨부하여 코드 제안
- 에러 스크린샷을 첨부하고 Copilot으로부터 수정을 위한 코드를 제안 받아 문제 해결
- Claude Sonnet 3.5, Claude Sonnet 3.7, Gemini 2.0 Flash, Gemini 2.5 Pro, and GPT-4o models



VS Code 에서 Copilot 코드 리뷰 받기

코딩 중 리얼타임으로 AI에서 피드백을 제공

에디터에서 직접 개발 작업을 가속화하고 코드의 품질을 향상

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure for "MY-TYPESCRIPT-APP" with files like assets/icons, dist, and src.
- Code Editor:** The file "index.ts" is open, showing TypeScript code for a TodoList class.
- Code Review Comment (1 of 2):** A callout from GitHub Copilot suggests handling the case where the find method returns undefined by explicitly checking for it.
- Suggested change:** The suggested change is to replace the original if statement with one that handles the undefined case.
- Buttons:** "Apply and Go to Next" and "Discard and Go to Next" are visible at the bottom of the code editor.
- Bottom Status Bar:** Shows tabs for "main", "Timeline", and "TypeScript".

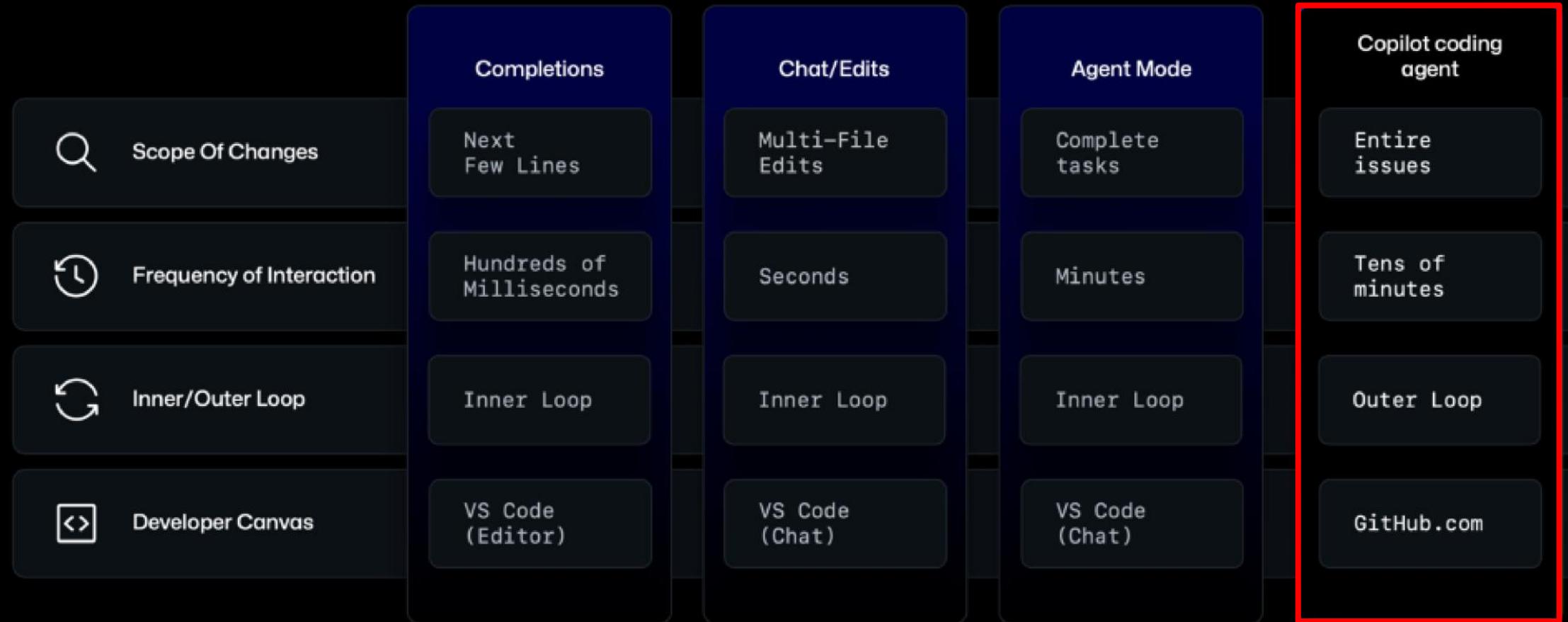


Coding Agent와 함께 개발

Copilot coding agent



GitHub Copilot / Code Generation

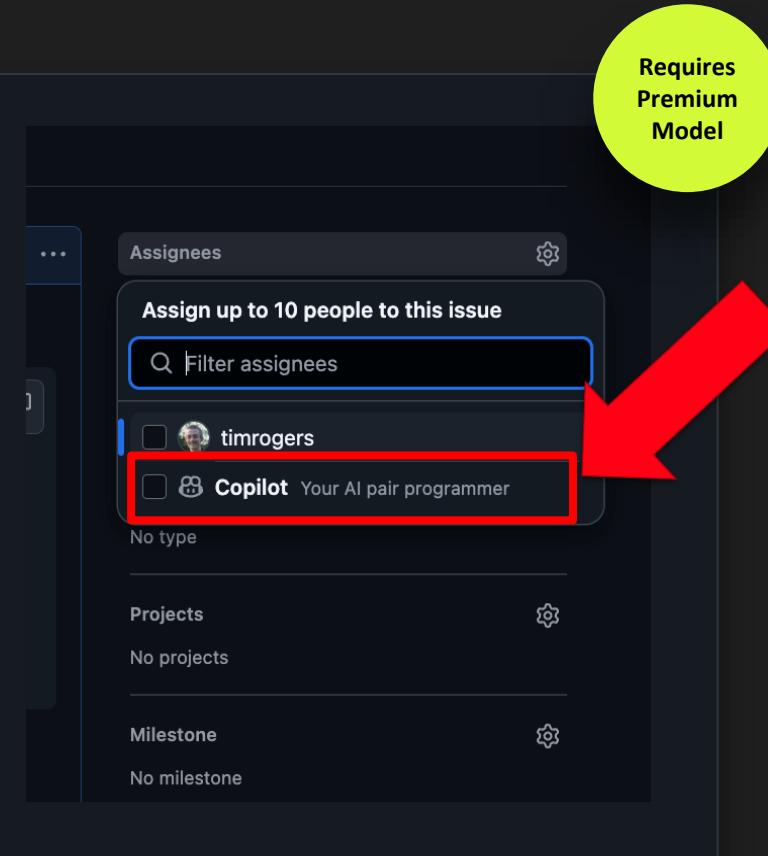


Copilot coding agent (preview)

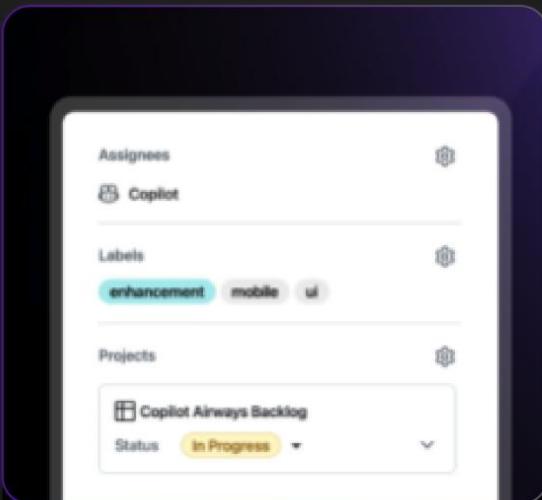
Copilot 코딩 에이전트를 사용하면
개발자는 Copilot에게 작업을 백그라운드에서
수행하도록 위임할 수 있어, 창의적이고 복잡하며
중요한 작업에 집중할 수 있습니다.

Copilot Chat을 통해,
혹은 이슈에서 **Copilot**을 assign하여
Copilot에게 작업을 지시하고,
풀 리퀘스트를 통해 진행 상황을 모니터링한
후, 풀 리퀘스트 리뷰를 통해 반복 작업을
수행합니다.

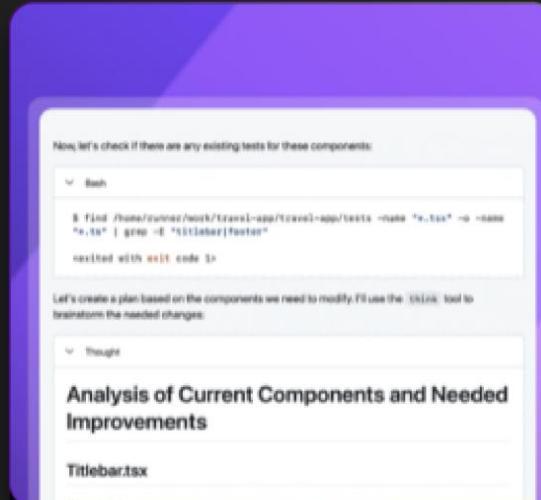
*Exclusive to Copilot Pro+ and Copilot Enterprise plans
as a Public Preview*



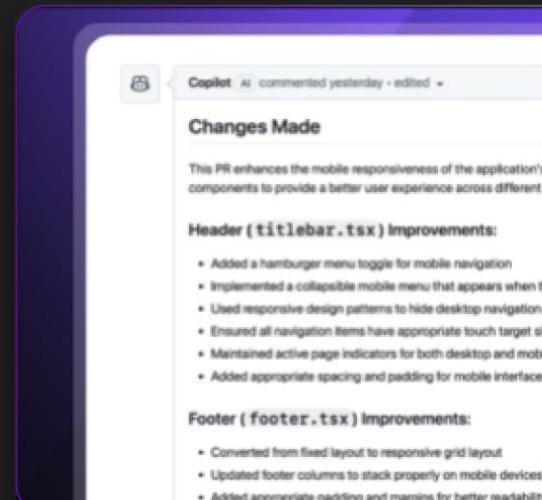
How it works



Copilot에게 작업을 할당하거나 Copilot Chat에서 요청하여 작업을 위임.



Copilot은 PR을 생성하고 검토를 요청합니다.



마치 사람이 생성한 PR처럼,
PR에 코멘트를 통해 Copilot에게
피드백을 제공하거나, 직접 코드를
확인하고 IDE에서 변경을 수행

VS code에서 Coding Agent에게 시키기



VS code에서 Coding Agent에게 시키기

[WIP] Header, ChartList, SongCard 컴포넌트 구현 (issue 3번) #11

Draft Copilot wants to merge 1 commit into main from copilot/fix-7ee96b18-659c-4d01-a1fe-d1f53fe260a6

Conversation 0 Commits 1 Checks 0 Files changed 0 +0 -0

Copilot AI commented now

Thanks for asking me to work on this. I will get started on it and keep this PR's description up to date as I form a plan and make progress.

Original description:

다음 세 가지 컴포넌트를 빌드하세요:

1. Header 컴포넌트: 앱 제목과 네비게이션 링크를 표시합니다.
2. ChartList 컴포넌트: 음악 순위 목록을 표시합니다.
3. SongCard 컴포넌트: 개별 곡 정보(순위, 제목, 작곡가, 가수, 앨범 사진)를 표시합니다.

이 컴포넌트들은 issue 3번의 요구사항을 충족해야 하며, 각 기능별로 React의 재사용 가능한 컴포넌트로 구현되어야 합니다.

Share your feedback on Copilot coding agent for the chance to win a \$200 gift card! Click [here](#) to start the survey.

Copilot is powered by AI, so mistakes are possible. Review before merging. Improve Copilot by leaving feedback using the 🔥 or 🚫 buttons.

Initial plan

Copilot assigned Copilot and son7211 now

Copilot started work on behalf of son7211 now

Assignees

- son7211
- Copilot

Reviewers

Suggestions

Request

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

In progress started now

Successfully merging this pull request may close these issues.

View session

The screenshot shows a GitHub pull request for issue #11. The title is '[WIP] Header, ChartList, SongCard 컴포넌트 구현 (issue 3번) #11'. A message from 'Copilot AI' is displayed, stating they will start working on the issue and keep the PR's description up to date. Below this, the original description is provided in Korean, detailing the three components to be built: Header, ChartList, and SongCard. The 'Assignees' section is highlighted with a red box, showing 'son7211' and 'Copilot' assigned to the task. The 'Development' section is also highlighted with a red box, showing 'In progress' status with 'started now'. The bottom right corner contains a note: 'Successfully merging this pull request may close these issues.'

VS code에서 Coding Agent에게 시키기

Code Issues 7 Pull requests 1 Actions Projects Wiki Security Insights Settings

← Back to pull request

[WIP] Header, ChartList, SongCard 컴포넌트 구현 (issue 3번) #11

Sessions

| | | |
|----------------------|----------|------------------|
| Initial implement... | 2m 53s | |
| Status | Duration | Premium Requests |
| In Progress | 2m 53s | 5 ⓘ |

I'll start by exploring the repository structure to understand the codebase and then implement the required React components according to the problem statement.

Call to github-mcp-server-get_issue

```
{  
  "id": 3185978421,  
  "number": 3,  
  "state": "open",  
  "locked": false,  
  "title": "Header, ChartList, SongCard 컴포넌트 구현"
```

Show more

Input

```
{  
  "owner": "johnjohncom",  
  "repo": "testkpopChart02",  
  "issue_number": 3  
}
```

View repository

```
.env  
.git  
.github  
.vscode  
docs
```

Coding Agent에게 맡길 수 있는 것들

[GitHub](#) 청구 팀이 [GitHub Copilot](#)의 코딩 에이전트를 사용하여 기술 부채를 지속적으로 소진하는 방법 - [GitHub Blog](#)

코드 테스트 커버리지 개선:

- 코드 베이스에서 테스트 커버리지 확장

종속성 바꾸기 :

- Mock up 라이브러리를 다른 라이브러리로 교체 라이브러리를 GitHub Copilot으로 교환하도록 이슈를 할당.

코드베이스 전반에 걸쳐 패턴 표준화:

- 코드베이스에서 오류를 반환하고 기록하는 여러 가지 방법 표준화.

프론트엔드 로딩 패턴 최적화:

- 모든 페이지 로드가 아니라 데이터가 요청될 때만 해당 API 호출을 수행하도록 애플리케이션을 변경하도록 요청

데드 코드 식별 및 제거:

- 프로젝트에서 사용하지 않는 기능, 오래된 엔드포인트 또는 오래된 구성이 있을 수 있는 곳을 안전하게 찾고 제거

GitHub에서 Copilot에게 자동 코드 리뷰 받기

Copilot 코드 리뷰는 버그,
잠재적인 성능 문제를 찾고
수정 사항을 제안하는 Copilot 에이전트에
기본 검토를 오프로드하는 데 도움이 됩니다.



Code review guide line 설정

The screenshot shows the GitHub Copilot settings page for a repository named 'kotlin-calculator'. The 'Settings' tab is selected. A red box highlights the 'Edit coding guideline' button and the 'Description' field. Another red box highlights the 'Sample' code block and the generated documentation.

Edit coding guideline Preview

Definition

Be clear and specific about what Copilot should look for.
[Learn more in the docs.](#)

Name *
kotlin

Description *

- 모든 함수이름은 'fun_' 으로 시작해야 한다
- 모든 함수는 doc 설명을 가져야 한다.

Sample [Generate from description](#)

```
1 private fun calculate(request: ArithmeticRequest): Double = when
2     (request.operation) {
3         "add" -> request.operand1 + request.operand2
4         "subtract" -> request.operand1 - request.operand2
5         "multiply" -> request.operand1 * request.operand2
6         "divide" -> request.operand1 / request.operand2
7         "power" -> Math.pow(request.operand1, request.operand2)
8         else -> throw IllegalArgumentException("Invalid operation: ${request.operation}")
9     }
10
11    fun handle(req: Request): Response {
12        val operation = operationLens(req)
13        val operand1 = operand1Lens(req).toDouble()
14        val operand2 = operand2Lens(req).toDouble()
15        val arithmeticRequest = ArithmeticRequest(operation, operand1,
16            operand2)
17        println("arithmeticRequest: $arithmeticRequest")
18        return try {
19            val result = calculate(arithmeticRequest)
20            Response(result)
21        } catch (e: Exception) {
22            Response("Error: ${e.message}")
23        }
24    }
25
26    companion object {
27        val operationLens: Lens<Request, String> = lens(
28            get = { req -> req.operation },
29            set = { req -> req.copy(operation = it) }
30        )
31        val operand1Lens: Lens<Request, Double> = lens(
32            get = { req -> req.operand1 },
33            set = { req -> req.copy(operand1 = it) }
34        )
35        val operand2Lens: Lens<Request, Double> = lens(
36            get = { req -> req.operand2 },
37            set = { req -> req.copy(operand2 = it) }
38        )
39    }
40}
```

Line 1
Function name must start with 'fun_'. Rename to match the required prefix. '''suggestion
private fun fun_calculate(request: ArithmeticRequest): Double = when (request.operation)
{
'''

Line 1
Function must include a doc description per coding guidelines.

Line 10
Function name must start with 'fun_'. Rename to match the required prefix.

Line 9
Function must include a doc description per coding guidelines. '''suggestion
Handles the incoming request and performs the specified arithmetic operation. * *
@param req The incoming request containing the operation and operands. * @return A
Response object containing the result of the arithmetic operation or an error message. */

VS Code에서 Prompt 재활용 : Prompt files

Copilot에게 재사용 가능한 프롬프트 instruction 파일(.md)을 제공

- 코드 생성: 컴포넌트, 테스트, migration등에 대한 재사용 프롬프트 제공 (예: React forms, API mocks)
- 도메인 전문지식 : 특정 도메인에 대한 전문적인 정보 제공 (예: 보안 프랙티스, 규정 준수확인 등)
- 팀 협업: 문서 패턴과 가이드라인 제공 (참조문서와 스펙제공)
- 온보딩: 복잡한 절차, 혹은 프로젝트의 특정한 패턴들에 대한 step-by-step 가이드

react-form.prompt.md

Your goal is to generate a new React form component.

Copy

Ask for the form name and fields if not provided.

Requirements for the form:

- * Use form design system components: [design-system/Form.md](../../docs/design-system/Form.md)
- * Use `react-hook-form` for form state management:
- * Always define TypeScript types for your form data
- * Prefer *uncontrolled* components using register
- * Use `defaultValues` to prevent unnecessary rerenders
- * Use `yup` for validation:
- * Create reusable validation schemas in separate files
- * Use TypeScript types to ensure type safety
- * Customize UX-friendly validation rules

security-api.prompt.md

Secure REST API review:

Copy

- * Ensure all endpoints are protected by authentication and authorization
- * Validate all user inputs and sanitize data
- * Implement rate limiting and throttling
- * Implement logging and monitoring for security events

이번 세션을 마치겠습니다
감사합니다