W06
오류 해결과 통합 환경

AI와 함께 버그 해결하기

코드 단위 테스트에서는 문제 없이 돌아가는데

유닛 테스트는 통과했는데, 실제로 돌려보니 안 된다?흔한 상황들:

- "로그인 테스트는 통과했는데, 실제 로그인하면 404 에러가 난다"
- "내 컴퓨터에서는 되는데, 배포하면 안 된다"
- "어제까지 됐는데, 오늘 갑자기 안 된다"

왜 이런 일이 생기는가?

- 유닛 테스트는 "부품 하나"만 검사한다
- 부품이 다 정상이어도, 조립하면 문제가 생길 수 있다
- 테스트 환경과 실제 환경이 다르다

"테스트 통과 ≠ 문제 해결"

에러 발생만 해결한다고 끝나는 것이 아니다

에러 해결 후 확인해야 할 것들:

- 원래 의도한 대로 동작하는가?
- 다른 기능이 망가지지 않았는가?
- 같은 문제가 다시 발생하지 않을 것인가?

흔한 실수:

- AI가 에러를 없애려고 해당 기능 자체를 삭제
- 에러 메시지만 숨기고 실제 문제는 남겨둠
- 한 군데 고치다가 다른 데를 망가뜨림

버그의 발생 원인

- 문법 오류 (Syntax Error)
- 런타임 오류 (Runtime Error)
- 논리 오류 (Logic Error)
- 통합 오류 (Integration Error)

문법 오류 (Syntax Error)

- 프로그래밍 언어의 규칙(문법)을 지키지 않아 발생하는 오류(오타나 괄호 누락 등)
- 대부분 코드를 실행하기 전(컴파일 단계)에 발견되므로 수정이 쉬움
- 프로그램 자체가 실행되지 않음
- 예시: ; (세미콜론) 누락, Print를 Prnt로 오타

문법 오류의 빠른 해결

AI는 실행하면서 빠르게 문법 오류를 해결한다

- 하지만 문법 오류만 해결되면 실행 가능하므로 문제가 해결되었다고 착각

문법 오류 해결의 함정:

- AI가 오타를 고치면서 변수명을 바꿔버림
- 누락된 괄호를 추가하면서 코드 구조를 변경
- 에러를 없애려고 해당 코드 블록을 통째로 삭제

"실행된다 ≠ 의도대로 동작한다"

질문 1. 트러블 슈팅 점검

"지금 문제를 해결하는 과정에서 이 수정이 원래 의도한 기능을 유지하는지 확인해줘"

"지금 변경으로 영향을 받는 다른 부분들에 대해서도 점검해줘"

논리 오류 (Logic Error)

- 프로그램은 정상적으로 실행되고 에러 메시지도 없음
- 결과가 개발자의 의도와 다르게 나오는 경우
- 컴퓨터는 시킨 대로 했을 뿐이라서, 원인을 찾기 가장 어렵고 위험한 버그
- 무한 루프, 잘못된 수식, 조건문의 부등호 방향 등.

정상동작이 무엇인지 확인하며 해결해야 한다

왜 논리 오류가 어려운가?

- 에러 메시지가 없다 → AI에게 줄 힌트가 없다
- 컴퓨터는 시킨 대로 했을 뿐
- "이게 맞는 건지 틀린 건지" AI도 판단할 수 없다

해결 방법: 작성한 요구사항을 적극 활용한다

- Given: 이 상태에서
- When: 이 행동을 하면
- Then: 이 결과가 나와야 한다 ← 이것이 "정답"

"정상 동작의 기준을 먼저 정의해야, AI도 틀린 걸 찾을 수 있다"

질문 2. 잘못된 로직 점검

“docs 폴더의 XXX 파일을 참고해 지금 오류를 해결하며 수정한 것이 원래 기능 의도대로 만든 것인지 확인해줘"

“XXXX 입력을 넣으면 YYY 결과가 나오는것이 자연스러운데 지금 그렇게 나오지 않아 어떤 로직 설계가 잘못 되어있는지 확인해줘"

런타임 오류 (Runtime Error)

- 런타임 오류: 코드는 맞는데, 실행 중에 터지는 오류

문법적으로는 문제없지만, 실행해보면 발생:

- 0으로 나누기 (계산 오류)
- 없는 파일 열기 (파일 접근 오류)
- 네트워크 연결 실패 (외부 서비스 오류)
- 메모리 부족 (리소스 오류)

왜 미리 발견하기 어려운가?

- "실행해봐야" 발생하는 오류
- 특정 조건에서만 발생 (예: 데이터가 비어있을 때)
- 개발 환경에서는 안 나다가 실제 배포 후 실행할때만 발생

런타임 오류를 확인하기 위한 개발자 도구

런타임 오류는 "실행 중"에 발생하므로, 실행 환경에서 확인해야 한다

개발자 도구의 역할:

- 실행 중 발생하는 에러 메시지 확인
- 네트워크 요청/응답 확인
- 변수 값 추적 (디버깅)

왜 개발자 도구를 알아야 하는가?

- AI에게 "안 돼요"라고 하면 AI도 모른다
- 개발자 도구에서 에러 메시지를 복사해서 AI에게 전달
- "어디서 문제가 생겼는지" 정확한 정보를 줄 수 있다

"개발자 도구 = AI에게 줄 정확한 에러 정보를 찾는 곳"

웹에서 개발자 도구에 접근하기

브라우저 개발자 도구 열기:

- Windows: F12 또는 Ctrl + Shift + I
- Mac: Cmd + Option + I
- 또는: 마우스 우클릭 → "검사(Inspect)"

주요 탭과 용도:

- Console: 에러 메시지 확인 (빨간 글씨)
- Network: API 호출 실패 확인 (빨간 줄)
- Elements: 화면 요소 확인

AI에게 전달할 때:

- Console 탭 열기
- 빨간색 에러 메시지 전체 복사
- "이 에러가 발생했어. 원인을 찾아줘" + 에러 메시지 붙여넣기

앱에서 개발자 도구에 접근하기

앱은 패키징을 만든 후 실행하기 때문에 개발자 도구에서 바로 오류를 가져오기 어렵다.

현실적인 대안:

- 배포 플랫폼의 로그 확인 (Vercel, Netlify 등)
- 에러 발생 시 화면 캡처
- 에러가 재현되는 상황을 정확히 기록

AI에게 전달할 때:

- "이 화면에서 [X] 버튼을 누르면 에러가 난다"
- "어제까지는 됐는데 오늘 안 된다"
- 가능하면 서버 로그나 배포 플랫폼의 로그 함께 전달

질문 3. 개발자 도구로 런타임 에러 해결하기

“지금 프로그램이 열렸는데 제대로 실행되지 않아. 런타임에서 발생한 에러를 확인하기 위해 개발자 도구를 어떻게 사용해야 하는지 알려줘"

“해당 개발자 도구에서 어느 위치의 오류를 너에게 알려주면 원인을 파악할 수 있는지 설명해줘"

에러 로깅

에러 로깅: 문제가 생겼을 때 기록을 남기도록 코드에 미리 작성해두는 것

- 에러가 발생했는데 개발자 도구에 아무것도 안 나온다..?
- 코드에 에러를 기록하는 부분이 없으면 로그도 없다
- 로그가 없으면 "어디서 문제가 생겼는지" 알 수 없다

문제가 생겼는데 정보가 없을 때, 콘솔에 의도적으로 정보를 로깅

- console.log("로그인 시도:", email)
- console.error("로그인 실패:", error)
- 기능을 만들 때 미리 요청해두면 나중에 문제 추적이 쉽다

"로그를 남겨야, 나중에 문제를 찾을 수 있다"

질문 4. 에러 로깅 관리하기

"이 함수에서 에러가 발생하면 로그를 남기도록 해줘"

"API 호출이 실패했을 때 어떤 에러인지 콘솔에 출력해줘"

"에러 발생 시 어떤 데이터가 들어왔는지 기록해줘"

“문제가 해결되었으면 작성한 에러 로그 코드를 정리해줘"

통합 오류(Integration Error)

통합 오류: 각 부품은 정상인데, 연결하면 문제가 생기는 오류

- 프론트엔드와 백엔드의 데이터 형식이 다름
- API 주소가 개발용과 배포용이 다름
- 서로 다른 버전의 라이브러리 사용
- 인증 토큰이 제대로 전달되지 않음

통합 오류의 특징:

- 에러 메시지가 모호한 경우가 많다
- "양쪽 다 확인해야" 원인을 찾을 수 있다

"통합 테스트 = 부품을 조립해서 함께 돌려보는 것"

통합 오류를 확인하기 위한 환경 구성

왜 유닛 테스트에서 못 잡는가?

- 유닛 테스트는 "혼자서" 테스트
- 연결 상태는 "함께" 테스트해야 발견
- "내 코드는 맞는데, 상대 코드와 안 맞는다"

통합 테스트를 하려면 "연결된 환경"이 필요하다

- 프론트 ↔ 백엔드: 로컬에서 두 서버 동시 실행
- 앱 ↔ API: Mock 서버 또는 실제 서버
- 서비스 ↔ DB: 로컬 DB 또는 클라우드 DB

통합 테스트의 체크 리스트

환경 확인:

- 프론트엔드 서버가 실행 중인가?
- 백엔드 서버가 실행 중인가?
- 데이터베이스가 연결되어 있는가?
- API 주소가 올바른가? (localhost vs 실제 서버)

데이터 확인:

- 테스트용 데이터가 준비되어 있는가?
- 데이터 형식이 양쪽에서 일치하는가?

인증 확인:

- 로그인이 필요한 기능이면 토큰이 있는가?
- API 키가 올바르게 설정되어 있는가?

질문 5. 통합 테스트 실행하기

"프론트와 백엔드를 동시에 실행해서 전체 시스템을 통합테스트 할 수 있는 전략을 수립해줘"

"로컬에서 통합 테스트할 수 있는 환경을 구성하고 실제 테스트를 진행해줘"

오늘 해결해야 할 문제

1. 실제 통합 테스트가 가능한 단계까지의 마일스톤을 확보하고 통합 테스트가 가능한 단계까지 개발을 진행합니다.
2. 해당 단계까지 개발을 진행하면서 발생하는 문법 오류, 논리 오류, 런타임 오류, 통합 오류 등을 발견하고 해결합니다.
3. 문제가 발생하는 과정에서 필요한 조치(요구사항 점검, 에러 로깅, 통합 환경 구성 등)를 적절하게 수행합니다.

Q&A