Claude를 이용한 E2E 테스트 코드 자동 작성

대규모 서비스 개발 및 운영을 위한 실전 엔지니어링 워크숍

목차

1. AI 기반 테스트 자동화의 필요성

현재 E2E 테스트의 문제점

- 테스트 시나리오 작성과 유지보수에 많은 시간과 인력 소요
- 복잡한 비즈니스 로직에 대한 테스트 케이스 누락 가능성
- 변경사항 발생 시 테스트 코드 수정 부담
- 테스트 코드의 품질 편차 발생

AI가 해결할 수 있는 부분

- 자연어 기반 테스트 시나리오 자동 생성
- 테스트 케이스 커버리지 향상
- 코드 변경에 따른 테스트 코드 자동 업데이트
- 일관된 품질의 테스트 코드 생성

2. Claude 소개 및 특징

소개

- Anthropic에서 개발한 **AI 언어 모델**

특징

- 뛰어난 자연어 이해 및 생성 능력
- **코드 생성과 분석에 특화**된 기능
- **맥락 이해**를 통한 정확한 요구사항 파악
- 대화형 인터페이스를 통한 손쉬운 상호작용

2. Claude 소개 및 특징

다른 AI 모델과의 차별점

- **더 긴 컨텍스트 윈도우** 지원
- **정확한 코드 생성** 능력
- 높은 일관성과 신뢰성
- 강력한 추론 능력

테스트 자동화에 적합한 이유

- 테스트 시나리오의 **자연어 이해 능력**
  - 품질 높은 테스트 코드 생성 가능
- **기존 테스트 코드 분석 및 개선 제안** 가능
  - 지속적인 테스트 품질 향상 가능

3. Claude를 활용한 E2E 테스트 자동화 프로세스

전체 워크플로우 개요

1. 요구사항 분석 및 테스트 범위 정의
2. 테스트 시나리오 작성
3. Claude를 통한 테스트 코드 생성
4. 코드 검증 및 수정
5. 테스트 실행 및 결과 분석
6. 피드백 반영 및 개선

3. Claude를 활용한 E2E 테스트 자동화 프로세스

테스트 시나리오 작성 단계

- 사용자 스토리 기반 시나리오 작성
- 테스트 케이스 설계
- 예상 결과 정의

테스트 코드 생성 단계

- Claude에 프롬프트 전달
- 코드 생성 및 검토
- 필요시 반복 생성

4. 테스트 시나리오 정의

- 테스트 자동 생성을 위해 **명확한 시나리오** 필요
- 사용자 스토리 기반으로 작성
- **입력 값**과 **기대 결과**

테스트 시나리오 구성 요소

- 전제조건이 무엇인지
- 각 단계별로 어떤 행동을 수행해야 하는지
- 어떤 결과를 기대하는지
- 어떤 예외 상황이 발생할 수 있는지

4. 테스트 시나리오 정의

예시: 로그인 기능 시나리오

- 정상적인 자격 증명으로 로그인
- 잘못된 비밀번호로 로그인 시도
- 존재하지 않는 이메일로 로그인 시도

4. 테스트 시나리오 정의

정상적인 자격 증명으로 로그인

전제 조건:

- 로그인 페이지에 접속: https://example.com/login

테스트 동작:

- 이메일 입력 필드에 'user@example.com' 입력
- 비밀번호 입력 필드에 'correct123!' 입력
- 로그인 버튼 클릭

기대 결과:

- 로그인 성공
- 대시보드 페이지로 이동 (/dashboard)

4. 테스트 시나리오 정의

잘못된 비밀번호로 로그인 시도

전제 조건:

- 로그인 페이지에 접속: https://example.com/login

테스트 동작:

- 이메일 입력 필드에 'user@example.com' 입력
- 비밀번호 입력 필드에 'wrong456!' 입력
- 로그인 버튼 클릭

기대 결과:

- 로그인 실패
- "비밀번호가 올바르지 않습니다" 오류 메시지 표시
- 로그인 페이지 유지

4. 테스트 시나리오 정의

존재하지 않는 이메일로 로그인 시도

전제 조건:

- 로그인 페이지에 접속: https://example.com/login

테스트 동작:

- 이메일 입력 필드에 'nonexistent@example.com' 입력
- 비밀번호 입력 필드에 'anypassword' 입력
- 로그인 버튼 클릭

기대 결과:

- 로그인 실패
- "등록되지 않은 이메일입니다" 오류 메시지 표시
- 로그인 페이지 유지

5. Claude를 이용한 테스트 코드 생성

Claude를 이용한 테스트 코드 생성 방법

- 테스트 대상 **시스템의 구조와 사용 기술 스택, 테스트 환경 정보**를 명시
- **Given-When-Then 형식**으로 테스트 시나리오를 상세히 기술
- 테스트 프레임워크나 의존성 정보 같은 특별한 고려사항을 추가

```
시스템:
테스트  시나리오:
프레임워크:

[상황1]
Given:
When:
Then:

[상황2]
Given:
When:
Then:
```

5. Claude를 이용한 테스트 코드 생성

프롬프트 예시 – 로그인 기능 테스트 코드 생성 (1/2)

```
다음 내용으로 E2E 테스트 코드를 TypeScript로 간략하게 작성해주세요.

시스템: React 애플리케이션
테스트 시나리오: 로그인 기능 테스트
프레임워크: Playwright

[정상적인 자격 증명으로 로그인]
Given:
- 로그인 페이지에 접속: /login
When:
- 이메일 입력 필드에 'user@example.com' 입력
- 비밀번호 입력 필드에 'correct123!' 입력
- 로그인 버튼 클릭
Then:
- 로그인 성공
```

(계속)

5. Claude를 이용한 테스트 코드 생성

프롬프트 예시 – 로그인 기능 테스트 코드 생성 (2/2)

```
- 대시보드 페이지로 이동 (/dashboard)

[잘못된 비밀번호로 로그인 시도]
Given:
- 로그인 페이지에 접속: /login
When:
- 이메일 입력 필드에 'user@example.com' 입력
- 비밀번호 입력 필드에 'wrong456!' 입력
- 로그인 버튼 클릭
Then:
- 로그인 실패
- "비밀번호가 올바르지 않습니다" 오류 메시지 표시
- 로그인 페이지 유지
```

5. Claude를 이용한 테스트 코드 생성

Claude로부터 코드 생성 결과

```
import { test, expect } from "@playwright/test";

test("올바른 자격 증명으로 로그인", async ({ page }) => {
  await page.goto("https://example.com/login");
  await page.fill("#email", "user@example.com");
  await page.fill("#password", "correctpassword");
  await page.click('button[type="submit"]');
  await expect(page).toHaveURL("https://example.com/dashboard");
});

// ...
```

6. 생성된 코드의 검증 및 수정

코드 생성 및 검토 프로세스

- 앞서 설계한 프롬프트를 기반으로 초기 코드를 생성
- 생성된 코드의 품질을 검토
- 필요한 경우 세부적인 조정을 요청
- 최종적으로 코드를 확정

6. 생성된 코드의 검증 및 수정

검증 방법

- Claude가 생성한 코드를 **직접 실행**하여 검증
- **실제 애플리케이션**과의 **요소 선택자** 일치 여부 확인
- 필요한 경우 **코드 수정**

일반적인 수정 사항

- **선택자(selector)** 수정
- **경로(URL)** 조정
- **메시지 내용** 검증

7. 한계점 및 고려사항

- Claude의 **이해도 한계**로 인해 정확하지 않은 코드 생성 가능성
- **보안 이슈**: 민감한 정보는 프롬프트에서 제외
- **유지보수성** 및 **확장성** 고려 필요

해결 방안

- **정확하지 않은 코드 생성 가능성**
  - 생성된 코드의 철저한 리뷰 필요
- **보안 강화**
  - 민감 정보를 위한 별도의 환경변수 관리
  - 테스트 데이터 익명화 처리
- **유지보수성 개선**
  - 모듈화된 테스트 구조 설계
  - 명확한 테스트 문서화 규칙 수립

8. 추가팁 – 소스 코드 인식을 위한 효과적인 프롬프트 작성법

전체 코드베이스 전달 방법

```
"다음은 현재 프로젝트의 전체 소스 코드입니다. 각 파일은 '---' 구분자로 구분됩니다:

파일: src/components/Login.tsx
---
[소스 코드]
---

파일: src/api/auth.ts
---
[소스 코드]
---

이 코드베이스에 대한 E2E 테스트 코드를 생성해주세요."
```

8. 추가팁 – 효과적인 컨텍스트 제공 방법

프로젝트 구조 설명

```
"프로젝트 구조:
- src/
  - components/
  - api/
  - utils/
  - tests/

주요 기능:
1. 사용자 인증
2. 데이터 조회
3. 실시간 업데이트

테스트 요구사항:
[요구사항 목록]"
```

8. 추가팁 – Cursor와 Windsurf를 활용한 테스트 코드 자동화

Cursor IDE

- **AI 기반** 코드 자동 완성 및 생성
- GitHub Copilot 통합
- **실시간 코드 분석** 및 제안

Windsurf

- **자연어 기반** 테스트 케이스 생성
- **테스트 시나리오** 자동 변환
- **다양한 프레임워크** 지원

감사합니다!

Q&A

질문이 있으시면 자유롭게 해주세요.