깃 브랜치 전략

✅ Git Flow 간소화 버전 브랜치 전략 문서

1. 개요

본 프로젝트는 **Git Flow의 간소화 버전(Simplified Git Flow)** 을 사용하여 소스 코드 버전을 관리하고 팀 협업을 효율적으로 진행합니다. 이 방식은 전통적인 Git Flow의 복잡한 release, hotfix 브랜치를 생략하고, 핵심 브랜치인 `master`와 `develop`만을 중심으로 운영하는 구조입니다.

---

2. 브랜치 구성

브랜치 이름	역할
`master`	최종 제출 및 배포를 위한 안정된 코드 보관소
`develop`	기능 개발을 위한 공동 작업 브랜치. 모든 개발자가 이 브랜치를 기준으로 작업

---

3. 작업 흐름 (Workflow)

📌 3.1. 브랜치 생성 및 기능 개발

1. 모든 개발자는 `develop` 브랜치에서 최신 코드를 pull 한 뒤, 기능 개발을 시작합니다.
2. 기능을 추가할 때는 `develop`에서 직접 작업하거나 `feature/기능명` 브랜치를 생성해 작업합니다.
3. 기능 구현이 완료되면 `develop` 브랜치에 병합합니다.

📌 3.2. 통합 및 테스트

- 모든 기능이 `develop` 브랜치에 병합된 뒤, 전체 기능 통합 테스트를 수행합니다.
- 코드 충돌 해결, 기능 검증 등 품질 점검을 이 시점에 수행합니다.

📌 3.3. 마스터 브랜치로 병합 및 제출

- 최종적으로 `develop` 브랜치의 안정화된 코드를 `master` 브랜치로 병합합니다.
- `master` 브랜치는 외부 제출 또는 배포용으로만 사용되며, 직접 작업하지 않습니다.

4. 장점

- 복잡한 release/hotfix 없이도 명확한 브랜치 역할 구분이 가능
- `develop` 브랜치에서 통합 테스트를 통해 안정성 확보 가능
- 소규모 팀이나 단기 프로젝트에 적합한 실용적인 구조

5. 주의사항

- `master` 브랜치에서는 직접 작업하지 않으며, 반드시 `develop`을 통해 병합합니다.
- 브랜치 병합 전에는 항상 최신 코드를 pull 하고, 충돌 해결을 철저히 합니다.
- 기능 단위 커밋 메시지를 명확하게 작성하여 변경 이력을 관리합니다.

6. 그림 구조