📘 Candy 팀 Git 협업 프로세스

1️⃣ 프로젝트 개요

프로젝트명: Candy
팀 구성: 4인 (리더 1명, 개발자 3명)
협업 방식: GitHub 기반 Feature Branch Workflow
목표: 안정적 코드 병합과 충돌 최소화를 통한 효율적 협업

2️⃣ 브랜치 구조 및 역할

브랜치	용도	담당
`main`	실제 배포용	팀 리더
`develop`	통합 개발 / QA용	전 팀원
`feature/*`	기능 개발용	각 담당자
`fix/*`	버그 수정용	관련 담당자
`hotfix/*`	긴급 수정 (배포 후)	팀 리더 또는 담당자

3️⃣ Candy 팀 역할 예시

이름	역할	담당 브랜치 예시
A	팀 리더 / 배포 담당	`develop`, `main`
B	로그인 / 회원가입	`feature/login`, `feature/register`
C	상품 / 장바구니	`feature/product`, `feature/cart`
D	결제 / 리뷰	`feature/payment`, `feature/review`

4 협업 진행 절차

단계	담당	명령어 / 행동	설명
1. 초기 세팅	리더	`main`, `develop` 생성	기본 브랜치 구성
2. 프로젝트 클론	전 팀원	`git clone <repo>`	원격 저장소 복제
3. 최신 develop 반영	전 팀원	`git checkout develop` → `git pull origin develop`	최신 코드 반영
4. 기능 브랜치 생성	각자	`git checkout -b feature/login develop`	기능 단위 브랜치 생성
5. 개발 및 커밋	각자	`git add . && git commit -m "feat: 로그인 기능 추가"`	기능 단위 커밋
6. 중간 동기화	각자	`git pull origin develop`	협업자 코드 반영
7. 푸시	각자	`git push origin feature/login`	원격 브랜치 업로드
8. PR 생성	각자	GitHub → "Compare & pull request"	base: develop / compare: feature/login
9. 코드리뷰 & 병합	리더	GitHub에서 Merge Pull Request	리뷰 후 merge
10. 로컬 최신화	전 팀원	`git checkout develop` → `git pull origin develop`	최신 코드 반영
11. 브랜치 정리	각자	`git branch -d feature/login` → `git push origin --delete feature/login`	정리
12. 다음 기능 시작	각자	`git checkout -b feature/cart develop`	새 작업 시작

5 커밋 메시지 규칙 (Conventional Commits)

Prefix	의미	예시
`feat:`	새 기능 추가	`feat: 장바구니 상품 추가 기능 구현`
`fix:`	버그 수정	`fix: 로그인 중 토큰 만료 오류 수정`
`refactor:`	코드 리팩토링	`refactor: 결제 모듈 구조 개선`
`docs:`	문서 수정	`docs: README 배포 절차 추가`
`style:`	코드 스타일 / UI 변경	`style: 상품 카드 hover 효과 추가`
`chore:`	빌드 / 설정 변경	`chore: ESLint 설정 업데이트`

6 PR 작성 규칙 (Pull Request)

PR 제목 예시

```csharp
[FEAT] 로그인 기능 추가
```

PR 본문 예시

```diff
- JWT 기반 로그인 기능 구현
- Axios 인터셉터 추가
- 로그인 페이지 레이아웃 개선
```

리뷰어 지정: 팀 리더 혹은 담당 파트 개발자

Merge 기준: 리뷰 승인 후 → `develop`으로 merge

---

7 충돌 발생 시 처리 절차

1 충돌 발생 시 GitHub에서 "Conflict" 표시

2 로컬에서 해결:

```bash
git checkout feature/login
git pull origin develop
# 충돌 파일 수정 후
git add .
git commit -m "fix: resolve merge conflict with develop"
git push origin feature/login
```

3 PR 다시 업데이트 → Merge 진행

---

8 브랜치 정리 절차

병합 완료 후

bash 코드 복사

```
git checkout develop
git pull origin develop
git branch -d feature/login
git push origin --delete feature/login
```

---

9 최종 정리 요약표

시점	명령어	목적
브랜치 생성 전	`git pull origin develop`	최신 코드 기반 시작
작업 중간	`git pull origin develop`	협업자 코드 반영
PR 직전	`git pull origin develop`	충돌 예방
병합 후	`git pull origin develop`	최신 상태 유지