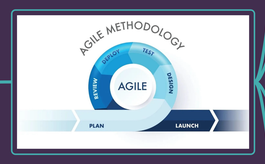
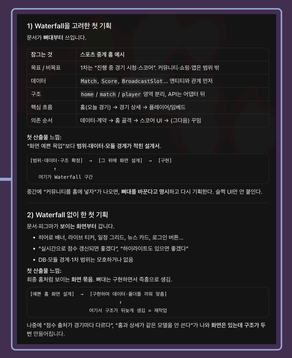
Harness
(= 저는 이렇게 합니다)

- 필요시 하네스 작동
  - 필요하지 않다면 그냥 AI와 뚝딱뚝딱
  - 내 프로그램 개발 과정에 따른 하네스 변경은 언제든 가능하다!
  - 푸시를 기준으로 하네스 변동 사항 체크!
- 프로그램 생성에 대한 커다란 맥락 사전 제공
  - 거시적 단계 (=설계)
    - WaterFall Process
      -
        - 폭포수는 거대하고 강력하다
        - 한 번 터진 폭포는 언제나 한 방향으로 흐르며 멈추거나 되돌아가지 않는다.
          - 실수하면 안되는 커다란 프로젝트
            - feat. NASA
      -

  - 미시적 단계 (=구현 & 그 이후)
    - Agile Process
      - 

        - 앱 개발에 주로 쓰이는 방식!
        - 일단 배포! 그리고 피드백 받기
        - 그다음 최대한 빨리 수정해서 다시 배포!
        - 이 과정을 무한 반복!
          - 도를스 같죠?😊
- 1. 설계
  - PLAN MODE Fable 혹은 상위 모델
    - waterfall Process를 요청
      - 이러면 프로젝트의 규모 세팅을 맥락으로 세팅
      - '뒤로 돌아갈 수 없다' 라는 규칙에 의해 보다 섬세한 설계를 시도함. 특히 엣지케이스를 많이 점검!
        - 

          1) Waterfall을 고려한 첫 기획

          문서가 **훨씬**부터 선입니다.

참고는 것	스포츠 중계 홈 예시
목표 / 비목표	'Fito 스'만을 중 경기 시뮬/스코어', 커뮤니티 소셜 영은 범위 밖
데이터	Match, Score, BroadcastStat… 엔티티와 관계 먼저
구조	home / match / player 명확 분리, API는 이걸터 뒤
핵심 흐름	홈(오늘 경기) → 경기 상세 → 플레이어/팀페드
의존 순서	데이터 계약 → 홈 골격 → 스코어 UI → (그다음) 무엇

          첫 산출물 느낌:
          "화면 먼저 목업"보다 범위·데이터·오들 관계가 확한 설계서.

          ```
          [범위·데이터·구조 확정]  →  [그 위에 화면 설계]  →  [구현]
                                  ↑
                              여기가 Waterfall 구간
          ```

          중간에 "커뮤니티를 홈에 넣자"가 나오면, **뒤로를 바꾼다고** 명시하고 다시 기획한다. 슬쩍 UI만 안 붙인다.

          2) Waterfall 없이 한 첫 기획

          문서 피그마가 보이는 **화면부터** 갑니다.

          - 하이로 배너, 라이브 티커, 일정 그리드, 뉴스 카드, 로그인 버튼…
          - "실시간으로 점수 갱신되면 좋겠다", "해이라이트도 있으면 좋겠다"
          - DB 모델 안개 가득 채워진는 오중하거나 없음

          첫 산출물 느낌:
          최종 결과물 보이는 **화면 목표**, **뒤로**는 구현하면서 즉흥으로 생김.

          ```
          [대충 홈 화면 설계]  →  [구현하며 데이터·흐름을 거꾸로 맞춤]
                                  ↑
                              여기서 구조가 뒤늦게 생김 = 재작업
          ```

          나중에 "점수 좀 바가 갱가면다 다르다", "홈과 상세가 같은 오열을 안 쓴다"가 나와 **화면은 있는데 구조가 두번** 만들어집니다.
- 2. 구현
  - 성능개선
    - 알고리즘 선택시키기
      - 자료를 어떻게 정렬시킬 것인가?
      - 자료를 어떻게 탐색할 것인가?
        - 빠른 알고리즘 vs 확실한 알고리즘 vs 안전한 알고리즘
      - 자료를 어떻게 저장시킬 것인가?
      - 데이터를 어떻게 저장시킬 것인가?
        - 데이터 테이블
          -
            - 실제 필요한 데이터 저장소보다 더 많은 공간이 낭비됨!
        - 노드를 링크시키기
          - 지금보시는 마인드맵. 필요한 만큼만 용량 쓰기!
          - 데이터를 관리할 때 동적 메모리 할당 원리를 고려한 알고리즘을 사용해줘
  - 기능별로 모듈화를 반드시 하도록!!!
  - 에이전트모드 혹은 하위모델
    - Agile Process 작동
      - 디버그 문서 생성
        - 에이전트용
        - 사람용!! 매우중요 ❗
- 3. 리팩토링
  - 에이전트모드 혹은 하위모델
    - Agile Process 작동
      - 디버그 문서 참조
- 4. 자산화
  - 작업 단위로 기록 남기기
  - 시간(날짜) 단위로 기록 남기기

- AI 활용과정
  - 첫 번째
    - 스프레드시트에 선택과목 조사
      - 앱스 스크립트
  - 두 번째
    - 제미나이 **Deep Research** 를 활용한 관심분야 깊게 살펴보기
      - 프로그램 다시 공부
      - 파이썬
      - AI 윤리
  - 세 번째
    - **RAG**를 활용한 교과 내용 재구성 작업
      - 정보
      - 데이터과학
        - 💙
      - 인공지능 기초
      - 정보과학
  - ~ 현재
    - 웹 제작
      - **SMART LMS**
      - **Prompt Card**
    - 앱 제작
      - AR Apps
      - **AI Mascot**
- 보안성 검토는 선택적!
  - Y/N