# 프로젝트 개요

* **프로젝트명**: 어떤 OTT가 나에게 적합한가?
* **한 줄 소개**: 사용자가 보고 싶은 콘텐츠를 입력하면, 어떤 OTT에서 제공하는지와 화질/음질/요금제를 비교하고 개인 선호도에 맞춰 추천을 제시하는 **OTT 비교·추천 플랫폼**.
* **레퍼런스 맥락**: 키노라이츠/왓챠피디아식 “어디서 볼 수 있나” + 화질/음질/요금제 비교 + 선호 기반 추천.

## 1. 목표 & 범위

### 1.1 목표

1. 콘텐츠별 **플랫폼 가용성(Availability)** 조회
2. 플랫폼/플랜별 **영상·음성 품질** 및 **구독료** 비교
3. 사용자 **선호·예산·광고 허용 여부** 등을 반영한 **추천 점수** 산출

### 1.2 범위(최초 버전)

* 지역: KR 기준(단일 리전 가정)
* 제공 형태: **구독(SVOD)** 중심 (선택적으로 **대여/구매** 확장)
* 뷰: **콘솔 UI(MVC)** → 이후 웹(스프링/프론트)로 확장 가능
* 데이터: JSON/CSV 정적 시드 + In-Memory Repository

## 2. 주요 사용자 시나리오(Use Cases)

* **UC-1: 콘텐츠 검색**
  + 사용자가 제목을 입력 → 동일/유사 타이틀 후보 제시 → 콘텐츠 선택
* **UC-2: 플랫폼 비교**
  + 선택한 콘텐츠 기준으로 **플랫폼 × 플랜** 테이블 출력 (화질/음질/요금/동시접속/광고)
* **UC-3: 추천**
  + 사용자 선호(예: 4K 선호, Dolby 음향 선호, 예산 12,000원, 광고 비허용) 입력 → 점수 상위 N개 추천
* **UC-4: 정렬/필터**
  + 가격 오름차순, 화질 상한, 광고 여부 등 필터링

## 3. 기능 요구사항 (Functional Requirements)

1. **검색**: 제목/연도/유형(Movie/Series) 키워드로 콘텐츠 조회
2. **가용성 조회**: 콘텐츠 → (플랫폼, 플랜) 조합별 제공 여부
3. **비교표 출력**: 해상도, HDR, 오디오 채널, 광고 유무, 월 구독료, 동시접속 수
4. **추천 점수**: 사용자 선호 + 제약(예산/광고) 반영, 상위 N 정렬
5. **데이터 관리**: In-Memory 로딩(초기 시드 JSON/CSV) + Repository 패턴

### 3.1 비기능 요구사항 (NFR)

* 간결한 콘솔 UX(3초 내 결과)
* 테스트 용이성(서비스/스코어러 단위 테스트)
* 확장성: 지역/플랫폼/플랜/제공 형태 추가 용이

## 4. 데이터 모델

### 4.1 Enum

* Quality: {SD, HD, FULL\_HD, UHD\_4K}
* Hdr: {NONE, HDR10, DOLBY\_VISION}
* Audio: {STEREO, \_5\_1, DOLBY\_ATMOS}
* OfferType: {SUBSCRIPTION, RENTAL, PURCHASE}

### 4.2 기본 엔티티

* **Content**
  + id: long, title: String, year: int, type: ContentType(MOVIE|SERIES), genres: Set<Genre>
* **Platform**
  + id: long, name: String, region: String, plans: List<Plan>
* **Plan** (플랫폼의 요금제)
  + id: long, platformId: long, name: String
  + monthlyFee: int(KRW), maxQuality: Quality, hdr: Hdr, audio: Audio
  + concurrentStreams: int, ads: boolean
* **Availability** (어떤 콘텐츠가 어떤 조건으로 제공되는지)
  + contentId: long, platformId: long, offerType: OfferType (초기엔 SUBSCRIPTION)
  + *(옵션)* planIds: Set<long> (특정 플랜에서만 가능 시)

### 4.3 사용자 선호

* **UserPreference**
  + budgetKRW: int, preferQuality: Quality, preferHdr: Hdr, preferAudio: Audio
  + allowAds: boolean
  + weights: PreferenceWeights { price, quality, audio, ads } (0~1, 합=1 권장)

## 5. 아키텍처 (MVC)

* **Model**: entity/enum + repository + service(도메인 로직)
* **View**: ConsoleView (검색/비교/추천 메뉴)
* **Controller**: SearchController, CompareController, RecommendController

### 5.1 레이어 책임

* Repository: In-Memory CRUD, 시드 데이터 로딩(JSON/CSV)
* Service: 검색/매칭/스코어링/정렬 비즈니스 규칙
* Controller: 입력 유효성, 서비스 호출, View DTO 구성
* View: 테이블 렌더링(콘솔), 정렬/필터 응답

## 6. 클래스 다이어그램 (Mermaid)

classDiagram  
 direction LR  
  
 class Content {  
 +long id  
 +String title  
 +int year  
 +ContentType type  
 +Set~Genre~ genres  
 }  
  
 class Platform {  
 +long id  
 +String name  
 +String region  
 +List~Plan~ plans  
 }  
  
 class Plan {  
 +long id  
 +long platformId  
 +String name  
 +int monthlyFee  
 +Quality maxQuality  
 +Hdr hdr  
 +Audio audio  
 +int concurrentStreams  
 +boolean ads  
 }  
  
 class Availability {  
 +long contentId  
 +long platformId  
 +OfferType offerType  
 +Set~long~ planIds  
 }  
  
 class UserPreference {  
 +int budgetKRW  
 +Quality preferQuality  
 +Hdr preferHdr  
 +Audio preferAudio  
 +boolean allowAds  
 +PreferenceWeights weights  
 }  
  
 class PreferenceWeights {  
 +double price  
 +double quality  
 +double audio  
 +double ads  
 }  
  
 Content "1" --> "\*" Availability  
 Platform "1" --> "\*" Availability  
 Platform "1" --> "\*" Plan  
  
 class ContentRepository  
 class PlatformRepository  
 class AvailabilityRepository  
  
 class SearchService  
 class MatchingService  
 class ScoringService  
 class RecommendationService  
  
 ContentRepository ..> Content  
 PlatformRepository ..> Platform  
 AvailabilityRepository ..> Availability  
  
 SearchService --> ContentRepository  
 MatchingService --> AvailabilityRepository  
 MatchingService --> PlatformRepository  
 ScoringService --> UserPreference  
 RecommendationService --> MatchingService  
 RecommendationService --> ScoringService

## 7. 주요 서비스 & 알고리즘

### 7.1 검색(SearchService)

* 입력: query:String
* 처리: 제목 정규화/토큰화, 부분일치 + 연도/유형 필터
* 출력: List<Content> (유사도 점수로 정렬 가능)

### 7.2 매칭(MatchingService)

* 입력: contentId
* 처리: Availability에서 해당 콘텐츠 제공 플랫폼 조회 → Platform과 Plan 조인
* 출력: List<PlatformPlanMatch>
  + {platform, plan, offerType}

### 7.3 스코어링(ScoringService)

* 입력: {platform, plan}, UserPreference
* **예시 점수 공식 (0~100)**
  + priceScore = max(0, 1 - (fee / budget)) \* 100
    - (예산 초과 시 0, 예산 이내일수록 높음)
  + qualityScore = qMatch(maxQuality, preferQuality)
  + audioScore = aMatch(audio, preferAudio)
  + adsScore = allowAds ? 100 : (plan.ads ? 0 : 100)
  + finalScore = priceScore\*w.price + qualityScore\*w.quality + audioScore\*w.audio + adsScore\*w.ads
* qMatch/aMatch는 동일=100, 한 단계 낮음=70, 두 단계 낮음=40, 그 이하=0 등 **계단 함수**로 단순화

### 7.4 추천(RecommendationService)

* 처리: 각 (platform, plan)의 finalScore 계산 → 상위 N 반환
* 출력: List<Recommendation> (점수/이유/핵심 차이 요약 포함)

## 8. 콘솔 View 플로우(초안)

1. 콘텐츠 검색 → 결과 목록 → 번호 선택
2. **비교표 출력**(플랫폼×플랜)
3. **선호 설정** 입력(예산/품질/음향/광고/가중치)
4. **추천 결과**(상위 N) + 정렬/필터 옵션

## 9. 시드 데이터 포맷 예시

### 9.1 platforms.json

[  
 {"id":1, "name":"Netflix", "region":"KR"},  
 {"id":2, "name":"Disney+", "region":"KR"},  
 {"id":3, "name":"Wavve", "region":"KR"}  
]

### 9.2 plans.json

[  
 {"id":11, "platformId":1, "name":"Standard", "monthlyFee":13500,  
 "maxQuality":"FULL\_HD", "hdr":"NONE", "audio":"\_5\_1", "concurrentStreams":2, "ads":false},  
 {"id":12, "platformId":1, "name":"Premium", "monthlyFee":17000,  
 "maxQuality":"UHD\_4K", "hdr":"DOLBY\_VISION", "audio":"DOLBY\_ATMOS", "concurrentStreams":4, "ads":false}  
]

### 9.3 contents.json

[  
 {"id":101, "title":"Interstellar", "year":2014, "type":"MOVIE", "genres":["SCIFI","DRAMA"]},  
 {"id":102, "title":"The Bear", "year":2022, "type":"SERIES", "genres":["DRAMA"]}  
]

### 9.4 availability.json

[  
 {"contentId":101, "platformId":1, "offerType":"SUBSCRIPTION", "planIds":[11,12]},  
 {"contentId":101, "platformId":2, "offerType":"SUBSCRIPTION", "planIds":[21]},  
 {"contentId":102, "platformId":2, "offerType":"SUBSCRIPTION", "planIds":[21]}  
]

## 10. 개발 계획(스프린트 초안)

* **S1**: 엔티티/리포지토리/시드 로더 + 콘솔 검색/선택
* **S2**: 매칭/비교표 + 선호 입력 폼
* **S3**: 스코어링/추천 + 정렬/필터
* **S4**: 단위테스트 + 리팩토링(전략/팩토리 도입)
* **S5(선택)**: Spring Boot REST + React Table UI

## 11. 확장 아이디어

* 지역/언어/자막·더빙 옵션, 오프라인 다운로드, 콘텐츠 회차별 화질 차등
* 대여/구매(EST/TVOD) 가격 비교, 번들(통신사/카드) 할인 규칙
* 크롤러/외부 API 연동(실시간 가용성)
* 개인화: 시청 이력 기반 플랫폼 선호 학습

## 12. 위험요인 & 대응

* **데이터 신뢰성**: 시드 갱신 자동화(스크립트), 수동 검증 체크리스트
* **스코어 편향**: 가중치 기본값 공개, 슬라이더로 조정 가능
* **법적 이슈**: 가격/상표 표기 가이드 정리, 출처 표시

## 13. 시퀀스 다이어그램(검색→비교→추천)

sequenceDiagram  
 actor U as User  
 participant V as ConsoleView  
 participant C1 as SearchController  
 participant S1 as SearchService  
 participant C2 as CompareController  
 participant S2 as MatchingService  
 participant C3 as RecommendController  
 participant S3 as Scoring/Recommendation  
  
 U->>V: 콘텐츠 제목 입력  
 V->>C1: search(query)  
 C1->>S1: findByTitle(query)  
 S1-->>C1: List<Content>  
 C1-->>V: 후보 목록  
 U->>V: 콘텐츠 선택  
 V->>C2: compare(contentId)  
 C2->>S2: listAvailability(contentId)  
 S2-->>C2: List<PlatformPlanMatch>  
 C2-->>V: 비교표 데이터  
 U->>V: 선호/예산 입력  
 V->>C3: recommend(contentId, prefs)  
 C3->>S3: scoreAndRank(matches, prefs)  
 S3-->>C3: 추천 리스트  
 C3-->>V: 상위 N 결과

## 14. 패키지 구조 초안(Java, MVC)

ott/compare (루트)  
 ├─ model/ (entity, enum, dto)  
 ├─ repository/ (in-memory + loader)  
 ├─ service/ (search, matching, scoring, recommend)  
 ├─ controller/ (search/compare/recommend)  
 └─ view/ (ConsoleView)

## 15. 콘솔 출력 예시(요약)

[검색 결과: "Interstellar"]  
1) Interstellar (2014, Movie)  
2) Interstellar: The IMAX Experience (2014, Movie)  
> 선택: 1  
  
[플랫폼/플랜 비교]  
플랫폼 플랜 화질 HDR 오디오 동시접속 광고 가격  
Netflix Premium 4K DolbyVision DolbyAtmos 4 N 17,000  
Disney+ Standard 4K HDR10 5.1ch 4 N 9,900  
  
[선호 입력] 예산 12,000원 / 선호품질 4K / HDR DolbyVision / 오디오 Atmos / 광고 허용 N  
[추천]  
1) Disney+ Standard (점수 88.5)  
2) Netflix Premium (점수 74.0)

## 16. 체크리스트 (과제 제출 관점)

* MVC 설계 명확성
* 클래스 다이어그램 & 시퀀스 다이어그램
* 기능 요구/비기능 요구
* 데이터 모델 및 시드 포맷
* 알고리즘(스코어링) 정의
* 확장/리스크까지 정리

### 부록: 점수 함수 의사코드

score(match, pref):  
 fee = match.plan.monthlyFee  
 priceScore = max(0, 1 - fee/pref.budgetKRW) \* 100  
 qualityScore = stepMatch(match.plan.maxQuality, pref.preferQuality)  
 audioScore = stepMatch(match.plan.audio, pref.preferAudio)  
 adsScore = (pref.allowAds ? 100 : (match.plan.ads ? 0 : 100))  
 with w = pref.weights  
 return priceScore\*w.price + qualityScore\*w.quality + audioScore\*w.audio + adsScore\*w.ads