Project YEOUL - 슬라이드별 발표 대본 **(**축약본**)**

슬라이드 **1:** 표지

안녕하십니까. 멀티 에이전트 AI 플랫폼, '프로젝트 여울'의 발표를 맡은 [발표자명]입니다. 저희 팀 가람이 준비한 인터랙티브 시나리오 엔진에 대해 설명드리겠습니다.

슬라이드 **2:** 기존 **AI** 서사의 한계

먼저 기존 생성형 AI 서사 플랫폼의 치명적인 한계를 말씀드리겠습니다. 첫째, 서사 붕괴입니다. 맥락 상실과 장기 기억력 부재로 일관성이 결여되어 몰입도가 파괴됩니다. 둘째, 환각 현상입니다. 대화가 길어질수록 죽은 NPC가 부활하거나 존재하지 않는 아이템을 언급하는 등 세계관 규칙을 무시합니다.

슬라이드 **3:** 왜 **'**여울**'**인가**?**

저희는 멀티 에이전트 조율 엔진으로 이 문제를 해결했습니다. Narrator Agent가 상황을 묘사하고, Validator Agent가 오류를 검수하며, World Manager가 절대적 진실을 관리합니다. 이들을 Orchestration 시스템이 유기적으로 연결하여 AI의 한계를 극복하고 완벽한 몰입감을 제공합니다.

슬라이드 **4:** 프로젝트 개요

플레이어의 선택이 실시간 데이터로 치환되어 흐르는 인터랙티브 플랫폼입니다. '상상을 서사로, 선택을 현실로'라는 슬로건 아래, 단순한 AI 대화가 아닌 모든 행동이 세계관을 바꾸는 진정한 참여형 서사를 지향합니다.

슬라이드 **5:** 타겟 유저

세 가지 핵심 유저층을 타겟합니다. 플레이어에게는 WorldState 엔진이 보장하는 엄격한 규칙 속 무한 자유도를 제공합니다. 빌더에게는 클릭 몇 번으로 복잡한 세계관을 구축할 수

있는 도구를 제공합니다. 작가에게는 AI와 협업하여 막힘없이 인터랙티브 스토리를 창작할 수 있는 환경을 제공합니다.

슬라이드 **6:** 프로젝트 목표

정성적으로는 AI를 넘어선 완벽한 몰입감을 목표로 합니다. 정량적으로는 세 가지 지표를 설정했습니다. 턴당 3초 이내 응답 속도, Validator를 통한 서사 오류 0% 지향, WorldState 엔진을 통한 데이터 정합성 100%입니다.

슬라이드 **7:** 목차

총 8개 파트로 구성됩니다. 파트 1-2는 프로젝트 배경과 개발 계획, 파트 3-6은 데이터 설계, 멀티 에이전트, 엔진 로직, 시스템 아키텍처, 파트 7-8은 서비스 시연과 향후 계획입니다.

슬라이드 **8:** 팀원 소개

간단히 팀원 소개 드리겠습니다. 김세영 팀장은 아키텍처 기획과 시나리오 빌더, CBT 운영을 담당했습니다. 안재현 팀원은 LangGraph 엔진 설계와 SSE 스트리밍, 인프라 관리를 총괄했습니다. 정진웅 팀원은 OAuth 인증 시스템과 프론트엔드 최적화, 랭킹 시스템을 개발했습니다.

슬라이드 **9: [Intermission]** 개발 계획

이제 개발 계획 파트로 넘어갑니다. 상상 속 아이디어를 실제 서비스로 구현한 과정과 LangGraph, FastAPI 선택 배경, 협업 프로세스를 설명드리겠습니다.

슬라이드 **10:** 전체 프로젝트 로드맵

네 단계로 진행했습니다. 12월 16-20일 멀티 에이전트 아키텍처 설계, 12월 20일-1월 20일 LangGraph 엔진과 빌더 개발, 이후 SSE 스트리밍과 시각화 고도화, 현재 Railway 클라우드 배포 및 CI/CD 구축을 완료했습니다.

슬라이드 **11:** 상세 일정 및 마일스톤

Jira 간트차트로 총 46일간 관리했습니다. 기획/설계, 에이전트 개발, 엔진 고도화, 운영/안정화의 4개 마일스톤을 차질 없이 진행했으며 현재 CBT와 최종 튜닝 단계입니다.

슬라이드 **12:** 기술 스택

LangGraph로 순환형 워크플로우를 구현하고, FastAPI와 sse-starlette로 비동기 스트리밍을 구축했습니다. 프론트는 HTMX와 Jinja2로 경량화하고, 데이터는 PostgreSQL과 Qdrant, MinIO로 하이브리드 구성했습니다. Railway와 GitHub Actions로 CI/CD를 완성했습니다.

다만, 유저 경험을 위해 시나리오 빌드 단계에서는 React를 적용하였습니다.

슬라이드 **13:** 기술 선정 사유 **1 - FastAPI**

먼저 FastAPI는 세 가지 이유로 선택했습니다. LangChain, OpenAI SDK와의 완벽한 호환성, LLM 대기 시간 동안의 비동기 처리 성능, 그리고 SSE 실시간 스트리밍을 가볍고 빠르게 구현할 수 있는 최적의 환경이었기 때문입니다.

슬라이드 **14:** 기술 선정 사유 **2 - LangGraph**

더불어 LangGraph를 사용하지 않는 단순 체인 구조로는 턴제 TRPG의 복잡한 상태 관리가 불가능했습니다. 상태 기반 루프와 조건부 분기를 정교하게 제어할 수 있는 LangGraph로 이야기의 회귀와 분기를 완벽하게 오케스트레이션할 수 있었습니다.

슬라이드 **15:** 협업 프로세스

Jira로 마일스톤 중심 티켓 관리, Git으로 기능별 브랜치와 코드 리뷰, Notion으로 개발 로그와 API 명세 공유를 통해 팀원 간 기술 간극을 제거하고 품질을 유지했습니다.

슬라이드 **16: [Intermission]** 데이터 및 상태 설계

데이터 및 상태 설계 파트입니다. 비정형 서사를 정형 데이터로 변환하는 방법과 AI 환각을 억제하는 WorldState 엔진을 설명드리겠습니다.

슬라이드 **17:** 데이터 설계 개요

이용 가능한 데이터로 만드는 핵심은 비정형 텍스트를 정형 JSON으로 제어하는 것입니다. 모든 서사 요소를 규격화하여 AI가 정확히 처리하게 하는 Strict Schema Strategy를 도입했습니다. 예를 들어 NPC 적대 여부는 AI가 아닌 시스템이 결정하고, 아이템 소지에 따라 서사가 분기됩니다.

슬라이드 **18: Scenario Schema** 설계

먼저 Pydantic으로 씬, NPC, 아이템을 엄격하게 정의했습니다. Root 모델 중심의 계층적 구조화, HP와 골드 등 실시간 수치 관리, Condition-Action-Effect 인과관계 규격화로 타입 안정성을 확보했습니다.

슬라이드 **19: WorldState** 엔진 설계

AI는 창의적 서사를 생성하지만, 모든 수치 변화는 WorldState 엔진이 검증합니다. HP, 골드, 위치는 코드 기반 규칙 엔진이 관리하며, "텍스트는 AI가 쓰지만 생사는 코드가 결정한다"는 원칙으로 환각을 차단하고 일관된 플레이를 보장합니다.

슬라이드 **20:** 데이터 영속화 전략

PostgreSQL JSONB로 정형/비정형 통합 관리, 모든 행동 직후 실시간 스냅샷으로 0% 유실 실현, 로드 시 스키마 검증으로 데이터 조작 방지를 구현했습니다. 언제든 과거 상태를 완벽 복원할 수 있습니다.

슬라이드 **21: RAG** 지식베이스 구축

Qdrant 벡터 DB로 방대한 세계관 데이터를 관리합니다. 현재 상황에 필요한 지식만 실시간 추출하여 LLM에 주입하는 RAG 파이프라인으로 컨텍스트 효율을 극대화하고 AI 정확도를 높였습니다.

슬라이드 **22:** 데이터 변환 로직

builder_agent.py가 자연어 설정을 Scenario Graph로 변환합니다. AI가 단계별로 데이터를 정제하고 논리 구조를 조정하여 창작자 의도를 시스템에 정밀 반영하는 핵심적인 가교 역할을 합니다.

슬라이드 **23:** 데이터 동기화 프로세스

프론트엔드, 백엔드, AI 엔진이 enrich_world_state 함수를 통해 실시간 동기화됩니다. DB에 저장된 데이터와 메모리 런타임 상태를 항상 일치시켜 선택이 즉각 반영되는 정교한 환경을 제공합니다.

슬라이드 **24:** 서사 컨텍스트 주입

정형 JSON 데이터가 시스템 프롬프트로 변환되어 AI에 주입됩니다. AI는 이를 '절대적 진실'로 인식하여 어떤 상황에도 설정 오류 없이 일관된 서사를 전개하며 완벽한 몰입감을 제공합니다.

슬라이드 **25: [Intermission]** 멀티 에이전트 시스템 설계

멀티 에이전트 시스템 파트입니다. 관심사 분리 원칙에 따른 전문가 에이전트 팀의 유기적 협업 구조를 살펴보겠습니다.

슬라이드 **26:** 에이전트 아키텍처 개요

SoC 원칙으로 전문가 팀을 구성했습니다. Orchestrator가 전체를 지휘하고, 서사 생성자, NPC, Validator가 각자 전문 영역에 집중하며 협력합니다. 분업 구조로 성능을 극대화하고 완성도 높은 결과물을 산출합니다.

[소프트웨어 공학에서 관심사 분리(Separation of Concerns, SoC) 원칙은 컴퓨터 프로그램을 구별된 부분으로 나누어 각 부문이 개개의 기능이나 관심사(Concern)만을 전담하도록 하는 디자인 원칙]

슬라이드 **27: Orchestrator** 에이전트

game_engine.py 메인 그래프가 이 역할을 합니다. 사용자 입력을 분석하고 최적 에이전트에 업무 할당, 각 에이전트 결과를 취합하여 전달, 시스템 상태를 유지하며 통신을 조율하는 컨트롤 타워입니다.

슬라이드 **28: NPC & Narrator** 에이전트

Narrator는 전지적 시점에서 상황을 묘사하고, NPC는 페르소나에 맞춰 대사를 생성합니다. LLM 출력이 JSON으로 엄격히 분리되어 화면에서 서사와 대사가 명확히 구분되며 깊은 몰입감을 제공합니다.

슬라이드 **29: Validator** 에이전트

서사 무결성의 파수꾼입니다. 생성된 서사가 세계관이나 상태값과 충돌하는지 실시간 판별하며, 오류 발견 시 즉시 재생성 루프를 가동해 설정 붕괴를 차단합니다.

슬라이드 **30:** 프롬프트 엔지니어링

Chain of Thought로 논리적 비약을 방지하고, 프롬프트 안의 예시를 통해 모델이 **in-context reasoning**, 즉 실무에서 말하는 few-shot learning으로 구체적 예시를 제공했습니다. AI가 맥락을 정확히 파악하여 플레이어가 이질감 없이 정교한 시나리오를 경험하게 합니다.

슬라이드 **31:** 에이전트 간 협업 프로세스

create_game_graph로 각 에이전트가 독립 노드로 작동합니다. intent_parser가 입력 분석 후 Conditional Edge로 최적 경로 결정, rule_engine이나 npc_actor가 처리하고 narrator가 통합하여 일관된 서사를 제공합니다.

슬라이드 **32:** 환각 억제 전략

환각 억제 전략 총 세 가지입니다. Schema Validation으로 AI 출력을 규격화된 객체로 강제 변환, WorldState Grounding으로 절대적 수치 데이터 기반 검증, Loop Verification으로 오류 시 즉시 재수행 및 보정을 수행합니다.

슬라이드 **33&34:** 품질 평가 지표 및 유저 평가

품질 평가 지표입니다. 보시는 바와 같이 Validator 도입 후 오류가 23건에서 8건으로 감소해 90% 정합성이라는 안정성을 빠르게 확보했습니다. //

유저 평가에서 78% 만족도를 얻었으며, 개선사항 피드백을 바탕으로 지속 개선하고 있습니다.

슬라이드 **35: [Intermission]** 시스템 로직 및 내러티브 엔진

시스템 로직 및 내러티브 엔진 파트입니다. 설계가 실제로 어떻게 작동하는지 구체적 메커니즘을 살펴보겠습니다.

슬라이드 **36: LangGraph** 기반 워크플로우

Start에서 intent_parser로 입력을 분석하고, Conditional Edge로 transition/battle은 rule_engine, chat/action은 npc_actor로 라우팅됩니다. PlayerState 객체가 전역 상태를 관리하며 복잡한 상태 전이를 안정적으로 처리합니다.

슬라이드 **37: Intent Parser**

예시로 "칼을 휘두른다"라는 자연어 문장을 "action: Attack"으로 변환합니다. 계층형 파서로 단순 규칙과 LLM 추론을 결합하여 모호한 자연어를 정교한 시스템 액션으로 치환합니다.

슬라이드 **38:** 게임 규칙 엔진

판정 로직은 AI가 아닌 파이썬 코드로 처리됩니다. 데미지 계산은 random 모듈과 수치 연산으로 수행되어 환각 없는 공정한 게임을 실현합니다.

슬라이드 **39: World State Manager**

HP가 0이면 'Dead' 상태로 변경하는 등의 로직은 state.py의 하드 코딩된 규칙으로 작동합니다. AI가 게임 근간 규칙을 흔들지 못하게 합니다.

슬라이드 **40:** 시나리오 브랜칭

유저가 작성한 문장에 따라 실시간으로 분기합니다. 그래프 엔진이 월드 상태와 조건에 맞는 Next Scene을 동적으로 연결하며 Conditional Edge로 무한한 서사 확장을 지원합니다.

슬라이드 **41:** 실시간 스트리밍

routes/game.py로 LLM 텍스트를 한 글자씩 실시간 전송합니다. 전체 완성을 기다리지 않고 생동감 있는 진행을 경험할 수 있습니다.

슬라이드 **42:** 엔진 최적화

시나리오와 LLM 캐싱 전략으로 호출 비용과 지연을 줄였습니다. 메모리 캐싱으로 응답 속도를 향상시켜 쾌적한 플레이 환경을 제공합니다.

슬라이드 **43: [Intermission]** 시스템 아키텍처

시스템 아키텍처 파트입니다. 안정적 서비스를 위한 Docker 기반 클라우드 인프라를 살펴보겠습니다.

슬라이드 **44:** 전체 서비스 인프라 구성도

FastAPI 백엔드를 중심으로 PostgreSQL은 메타데이터, Redis는 세션 캐싱, Qdrant는 벡터 검색, MinIO는 파일 저장을 담당하는 분산 아키텍처를 Railway에 구축했습니다.

슬라이드 **45:** 백엔드 아키텍처

FastAPI 중심으로 인증, 게임 로직, 데이터 접속이 레이어드 아키텍처로 분리됩니다. 비동기 구조로 대량 요청을 안정적으로 처리합니다.

슬라이드 **46:** 프론트엔드 아키텍처

Jinja2 템플릿과 React를 활용하여 빠른 UI를 구축했습니다. 동기/비동기 인터페이스를 구조화하고, 픽셀 아트 스타일과 다크 모드로 TRPG 특유의 몰입감 있는 분위기를 완성했습니다.

슬라이드 **47:** 인프라 배포

GitHub 코드 푸시 시 자동으로 빌드 및 배포되는 CI/CD 파이프라인을 구축했습니다. Railway PaaS 플랫폼으로 서버 관리 없이 완전 관리형 서비스 및 안정적 운영 환경을 확보했습니다.

슬라이드 **48:** 데이터베이스 구성

PostgreSQL에 정형 메타데이터를, Qdrant에 비정형 벡터 지식을 저장하는 하이브리드 구조입니다. 데이터 특성별로 최적화하여 성능을 높이고, init_db.py로 스키마를 자동 관리합니다.

슬라이드 **49:** 지능형 가이드 챗봇

키워드 매칭과 Qdrant 벡터 검색을 결합한 하이브리드 RAG 파이프라인으로 빠른 응답 속도를 확보했습니다. AI 장애 시 Fallback 로직으로 자동 전환되어 중단 없는 고신뢰성 가이드 서비스를 제공합니다.

슬라이드 **50:** 보안 및 인증

Cloudflare로 DDoS 방어, 환경 변수로 민감 정보 격리, OAuth 2.0과 토큰 기반 세션으로 비밀번호 유출을 차단하는 3단계 보안을 구축했습니다.

슬라이드 **51: [Intermission]** 서비스 구현

서비스 구현 파트입니다. 실제 사용자가 경험하는 인터페이스와 서사 흐름을 보여드리겠습니다.

슬라이드 **52:** 실시간 서비스 시연

지금부터 라이브 시연을 통해 에이전트와 엔진의 실제 작동을 보여드리겠습니다. 한 줄 아이디어가 풍성한 세계관으로 변하고, AI 마스터가 실시간으로 서사를 지휘하는 과정을 확인하시기 바랍니다.

[실제 서비스 시연 진행]

슬라이드 **53: [Intermission]** 트러블슈팅 및 향후 계획

마지막 파트입니다. 기술적 난제 극복 과정과 향후 고도화 계획을 말씀드리겠습니다.

슬라이드 **54:** 트러블슈팅 **1**

여섯 가지 문제를 해결했습니다. 서버 타임아웃은 비동기와 스트리밍으로, AI 환각은 WorldState를 상위 개념으로 정의하여, 보안 문제는 CloudFlare와 OAuth 2.0으로 해결했습니다.

슬라이드 **55:** 트러블슈팅 **2**

협업 충돌은 커밋 문구 규칙으로, 부족한 상상력은 CBT를 통한 피드백 수집으로, 제한적인 운영 로그는 디버깅 시스템 강화로 해결했습니다.

슬라이드 **56:** 프로젝트 성과 요약

세 가지 성과를 달성했습니다. RAG, WorldState, SSE로 서사 유지력과 무결성을 확보했고, 텍스트만으로 시나리오를 생성하는 환경을 구축했으며, 콘텐츠 플랫폼으로서의 가능성을 확인했습니다.

슬라이드 **57:** 한계점 분석

고성능 LLM의 API 비용 문제, 기초적인 물리 엔진 로직, 데이터 품질에 따른 AI 답변 편차가 한계로 남아있습니다. 이를 성장의 이정표로 삼아 개선하겠습니다.

슬라이드 **58:** 서비스 고도화 **1 -** 멀티플레이 확장

멀티 유저 세션 도입, AI 주도의 실시간 턴 관리, 파티 협동 로직으로 여러 명이 함께 플레이하는 역동적인 서사 환경을 구축할 예정입니다.

슬라이드 **59:** 서비스 고도화 **2 -** 멀티미디어 확장

상황별 BGM 자동 생성, 캐릭터별 TTS 음성 합성, 시청각 결합 멀티미디어 환경으로 오감을 자극하는 압도적 몰입감을 완성할 계획입니다.

슬라이드 **60:** 최종 결론 및 **Q&A**

프로젝트 여울은 AI와 인간이 함께 이야기를 만드는 차세대 콘텐츠 플랫폼으로 성장할 것입니다. 누구나 창작의 즐거움을 누리고 고유한 서사를 향유할 수 있는 세상을 꿈꿉니다.

긴 발표 경청해 주셔서 감사합니다. 이제 질의응답 시간을 갖겠습니다.

[발표 종료**]**