Minecraft RAG Chatbot

Minecraft Mini Mate

intel 인공지능 앱크리에이터 양성과정

CONTENTS

1. 팀/프로젝트 소개
2. 문제 및 요구 사항 정의
3. 주요 기능
4. 페르소나 및 사용자 시나리오
5. 수익 모델
6. 프로젝트 구성
7. 협업 방식 소개
8. 개발 과정
9. 회고

01. 팀/프로젝트 소개

팀 소개

PM

고석현

프론트엔드

김태호

데이터 엔지니어

위형빈

AI 엔지니어

윤주완

데이터 엔지니어

차유원

프로젝트 소개

Minecraft Mini Mate

“ 마크 하다 막힐 때, 누구나 쉽게 쓰는 마인크래프트 AI 친구!! ”

게임 마인크래프트 플레이 중 실시간 자연어 입력 기반으로 현재 상황에 대한 해결책,
할 수 있는 것 등의 정보를 얻는 **“slm + rag”** 시스템 기반의 데스크탑 앱

02. 문제 및 요구 사항 정의

문제 정의

게임 정보 검색의 비효율성

게임 내 정보검색

- 내가 가진 아이템으로 무엇을 할 수 있는가? - 일부만 제공
- 내가 하려는 일에 어떤 아이템이 필요한가? - 알 수 없음

게임 내 툴팁만으로는 레시피·용도 파악 어려움

- 결국, 검색 필요

문제 정의

게임 정보 검색의 비정확성

1. 인터넷 검색

마인크래프트는 대규모·소규모 업데이트가
연중 수차례 진행
→ 이미 알고 있던 레시피·생태계가 바뀌거나
완전히 새로운 메커니즘이 추가

블로그 등 온라인 자료는
게시 시점에 고정되어 있어,
최신 버전 기준으로,
더 이상 사실이 아닌 정보가 빈번한 문제.
입문자·복귀 유저는 무엇이 최신 정답인지 스
스로 판별하기 어려움.

문제 정의

게임 정보 검색의 비정확성

2. LLM 활용

대부분의 **LLM**은 사전 학습 시점 이후 데이터가 반영되지 않음
→ 최신 패치 내용이 누락되어 과거 버전 기준 답변을 제시할 수 있음.
또한 광범위한 지식 기반으로 설계되어 도메인(마인크래프트) 특화 정확도가 낮음

시계 조합법을 **Chat GPT**에 물어볼 경우
→ 잘못된 정보 대답

🛠 제작 방식 (제작대 사용)

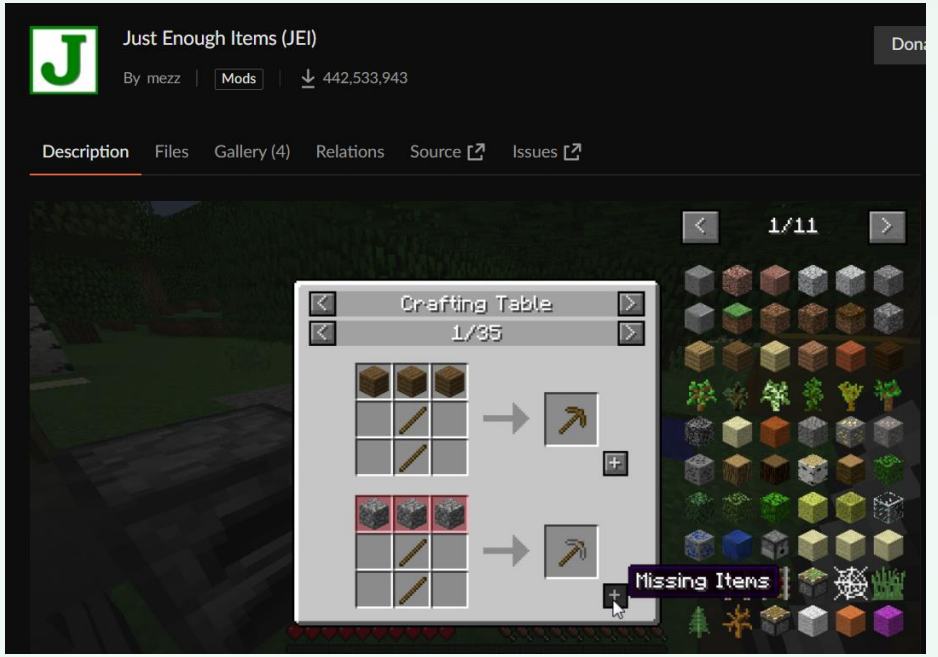
```
css

[ 금 ]   [ 레드스톤 ]   [ 금 ]
[     ]   [ 금 ]       [     ]
[     ]   [     ]      [ 금 ]
```

문제 정의 - 수익성

게임 정보 관련 기존 앱과의 비교

마인크래프트 조합법 관련 사이드 앱의 다운로드 수는 **4억**

- 해당 앱의 경우 조합법만 제공
- 해당 앱을 설치하기 위해서는 까다로운 전문 지식 필요



요구사항 정의

01

실시간성

- 너무 느리면 검색이 더 빠름

02

정확성

- 잘못된 정보는 전달하지 않도록

03

유저 친화성

- 직관적인 UI 로 쉽게 사용 가능하도록

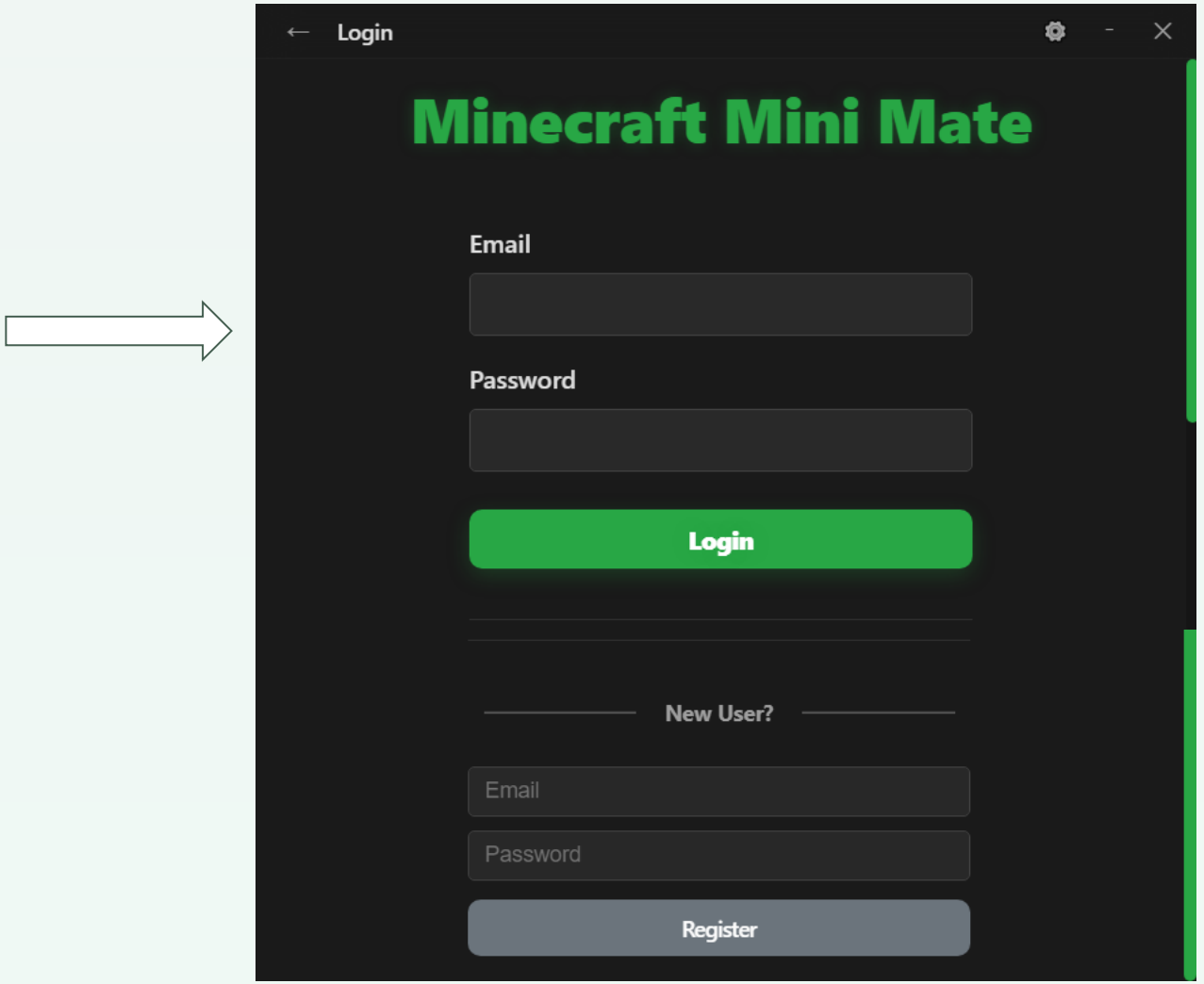
04

최대한 가볍게

- 게임 플레이와 동시에 사용 가능하도록

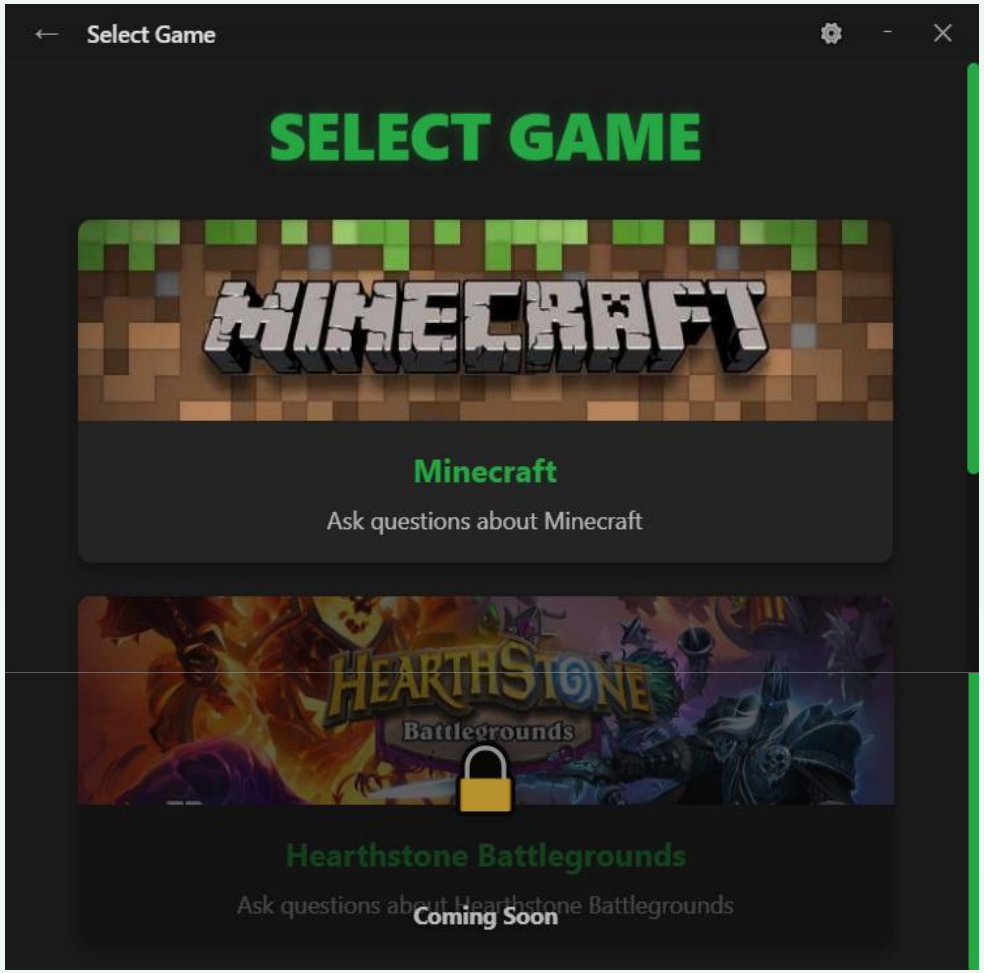
03. 주요 기능

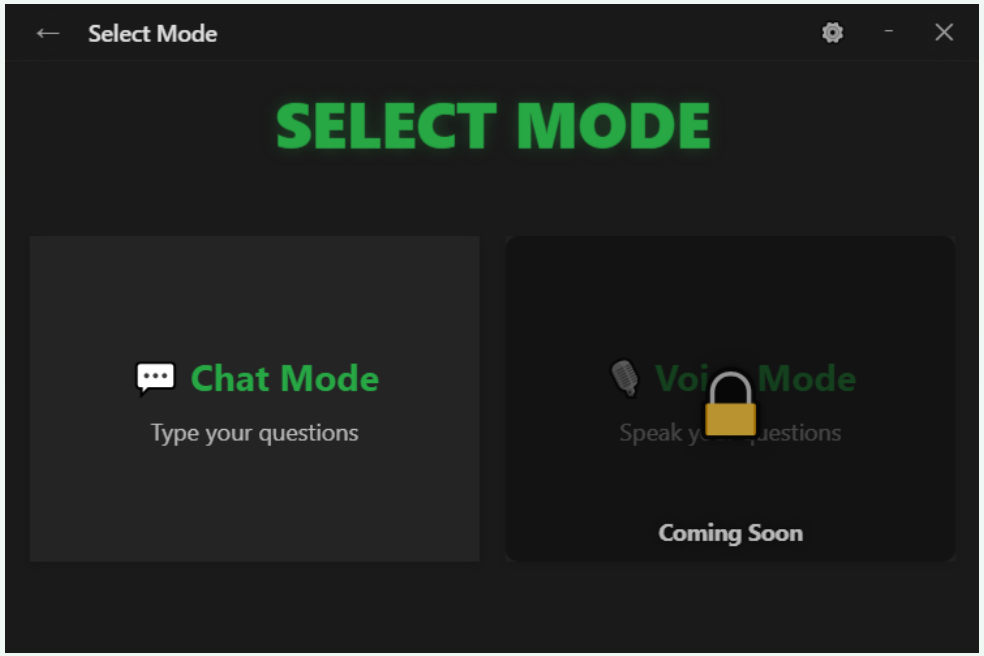
주요 기능



초기화면 -> 로그인 및 회원가입 화면

주요 기능



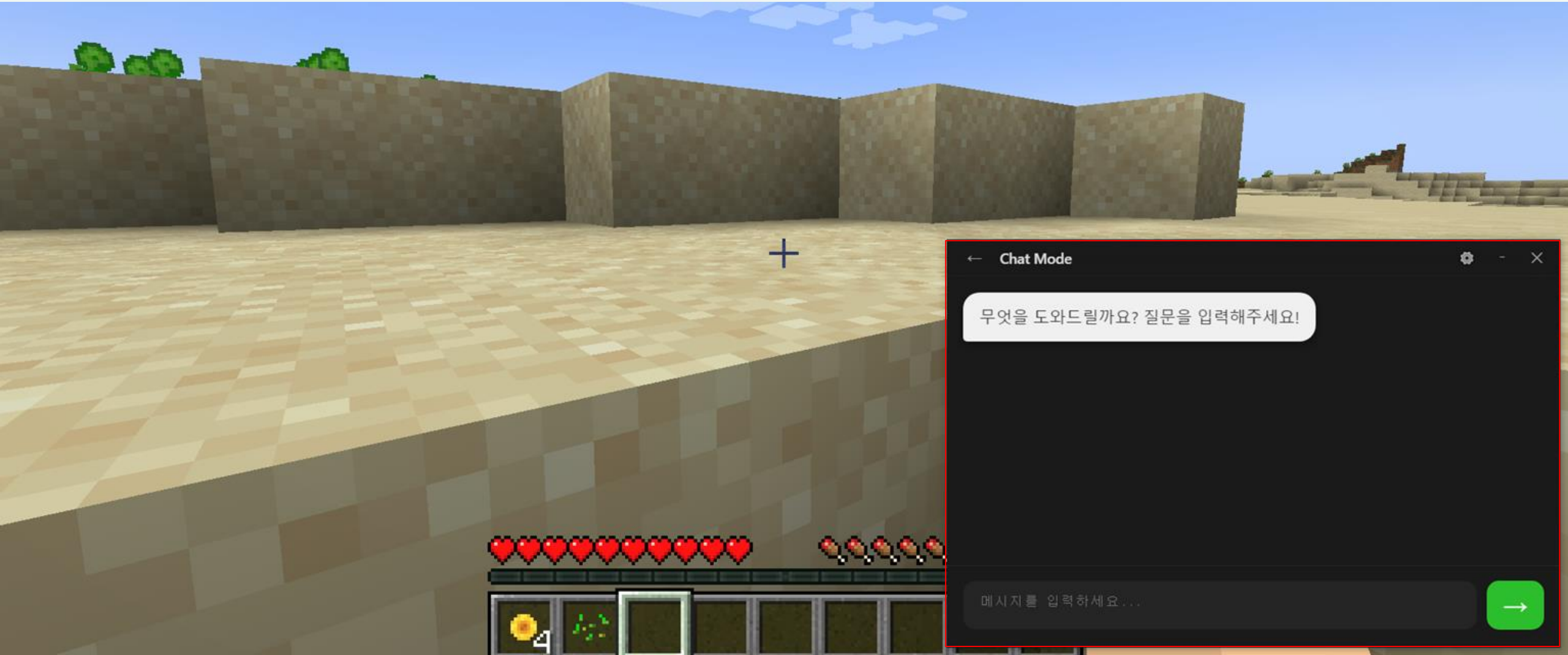


게임 선택 화면

모드 선택 화면 - **chat / voice**

주요 기능

Chat Mode – 마인크래프트 게임 플레이 중 동시에 한 화면에서 사용 가능

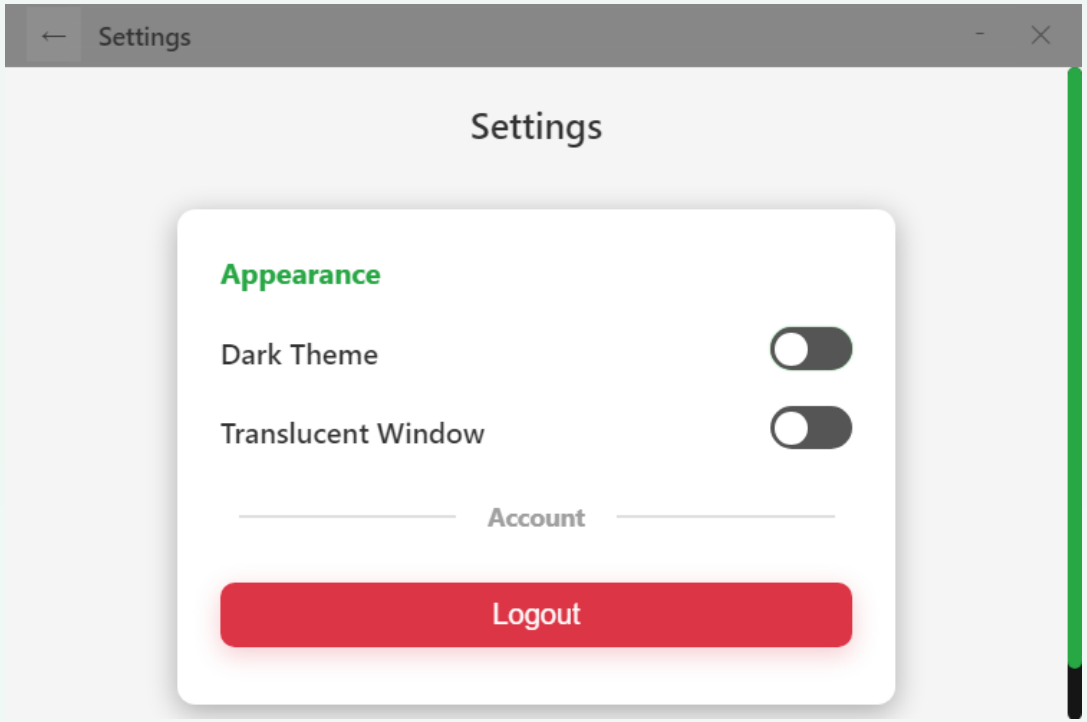


주요 기능

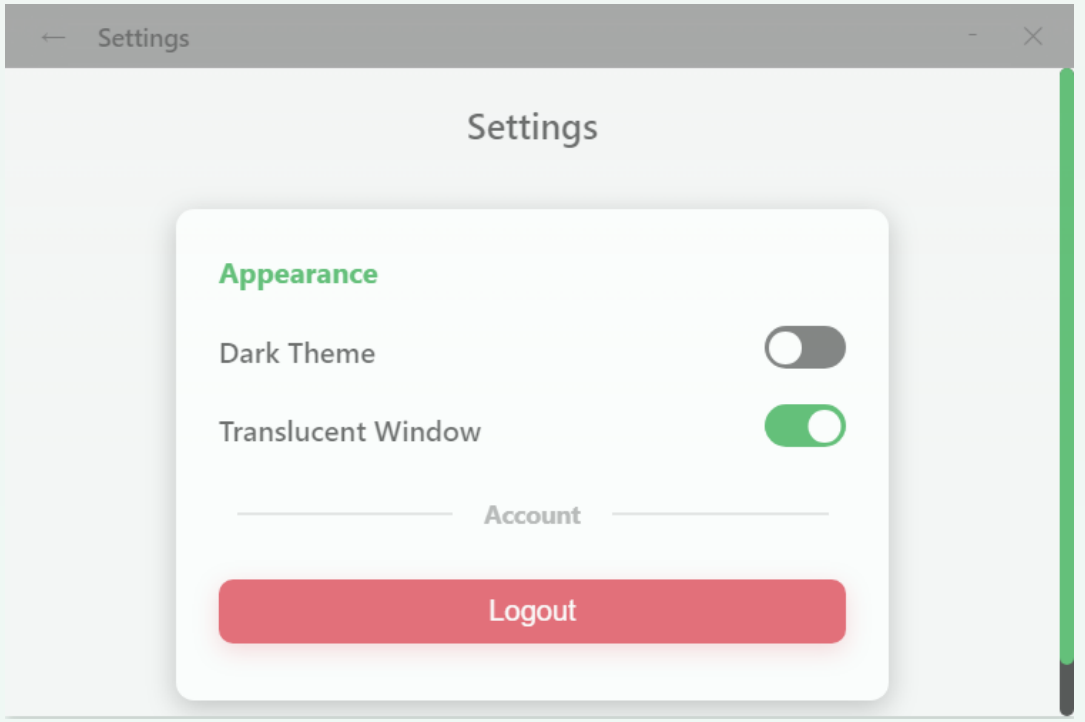
자연어 질문 입력 – 아이템 조합법, 레시피 등 반환

주요 기능

설정 **settings** 화면



라이트 모드



반투명 모드

04. 페르소나 및 사용자 시나리오

페르소나

이름 : 최성민

나이 : 22세

직업 : 대학생, 카페 알바

마인크래프트 게임 총 플레이 타임 10시간 정도의 초보자.

MBTI : ENFP

외향(E): 친구들과 전략을 공유하고, 커뮤니티 토론을 즐김
직관(N): 게임 내 다양한 환경과 아이템 조합을 실험해 보고 싶어 함
감정(F): 긍정적인 피드백·칭찬에 동기 부여를 크게 받음
인식(P): 계획보다는 상황에 맞춰 즉흥적으로 대응하는 것을 선호

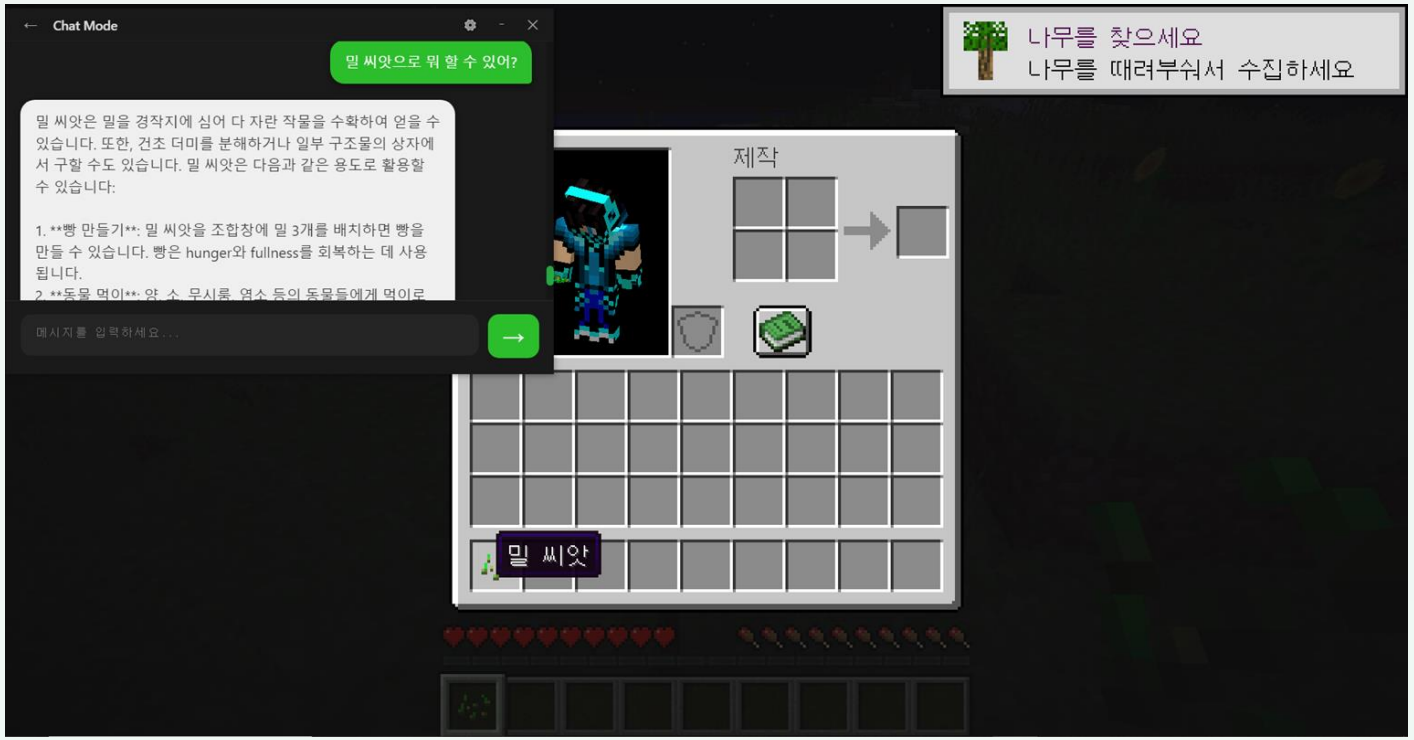
Pain Point

- 공식 위키·블로그 글이 장황해 핵심만 뽑기 어려움
- 게임 중 두 번 이상 앱 전환하면 몰입도 저하
- 잘못 제작 시 아이템 낭비 → 좌절감

Needs

- 핵심 정보만 뽑아주는 간단한 요약으로 제작 스트레스 없이 빠르게 시작하고 싶음
- 제작 실패로 인한 아이템 낭비를 줄이기 위한 정확한 단계별 가이드 필요
- 게임 중 전환 없이 바로 확인 가능한 직관적인 인터페이스의 도우미 앱 필요

사용자 시나리오



플레이 중 밀 씨앗을 발견한 유저. 밀 씨앗으로 무엇을 할 수 있는지 궁금해진 유저.

→ **Minecraft Mini Mate** 앱을 게임플레이 화면에 같이 띄워놓고 질문 후 답변 얻음.

05. 수익 모델

수익 잠재력

JEI의 1% 전환 시 월 최대 100만 달러 이상의 광고 매출 확보 가능!

경쟁 앱인 Just Enough Item (JEI)의 다운로드 수: 4억 회

1% 전환 시: 400만 명의 사용자

광고 단가(eCPM) 가정: $5 (예시)

월 광고 노출 수: 1,000회/사용자

월 예상 광고 수익: 400만 사용자 × 1,000 노출 × $5 eCPM / 1,000 = $2,000,000

수익성 가정 & 계산

MAU (Monthly Active Users) : 400만 명

(1% 전환 가정)

- DAU (Daily Active Users): 133만 명 (MAU / 30일)
- 하루당 광고 노출 수: 1,000회
- eCPM (Effective Cost Per Mille): $5 (가정)
- 월 예상 매출: DAU × 하루당 광고 노출 수 × eCPM × 30일 / 1,000 = $2,000,000

06. 프로젝트 구성

시스템 아키텍처

GUI

01

Translator

GoogleTranslator

02

Embedder

all-mpnet-base-v2

03

FAISS

04

sLM

EXAONE-3.5-2.4B-Instruct

05

시스템 흐름

앱 실행 - > 게임 선택 - > 질문 입력

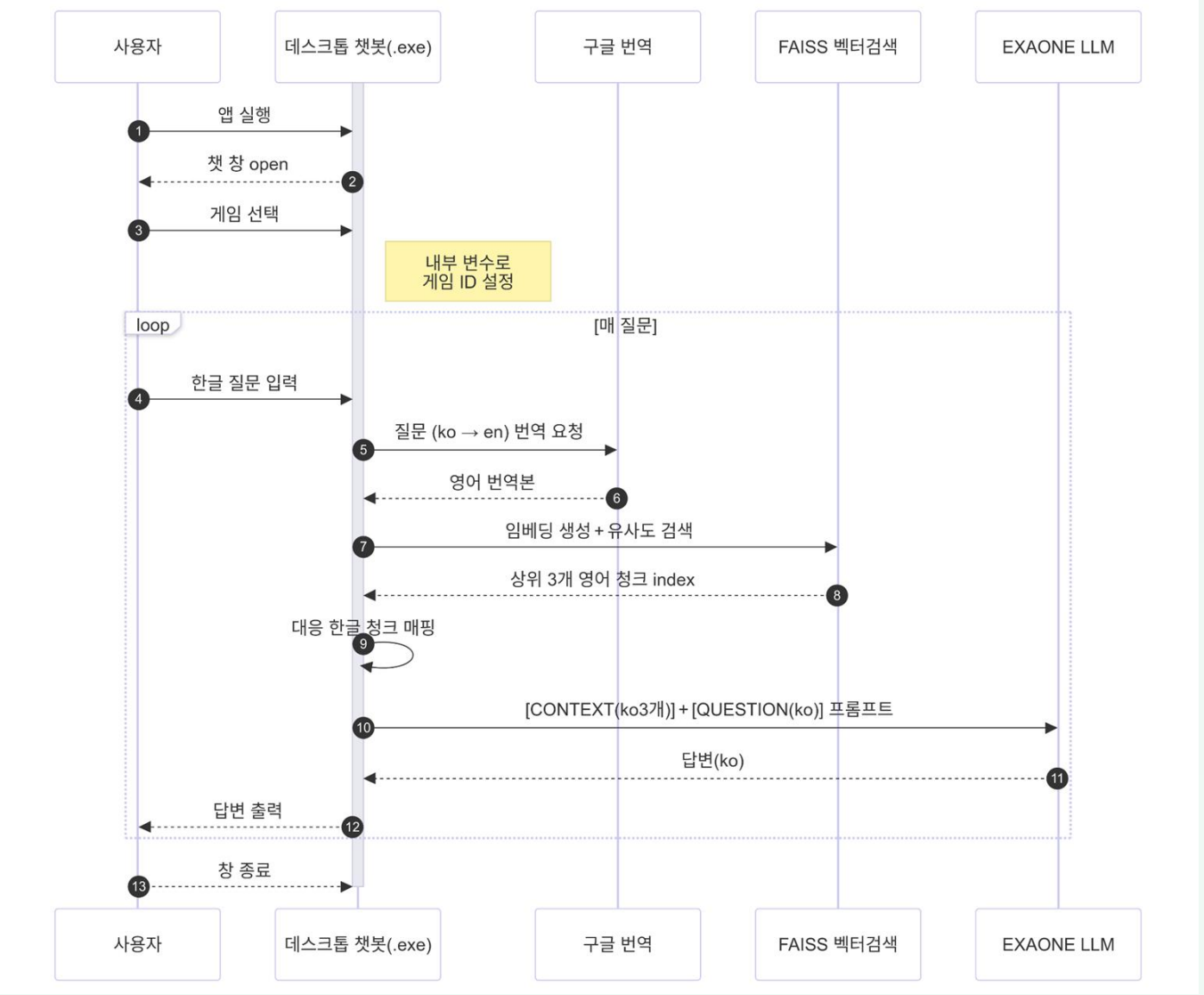
Google 번역
(ko→en)

FAISS 상위 **3** 컨텍스트

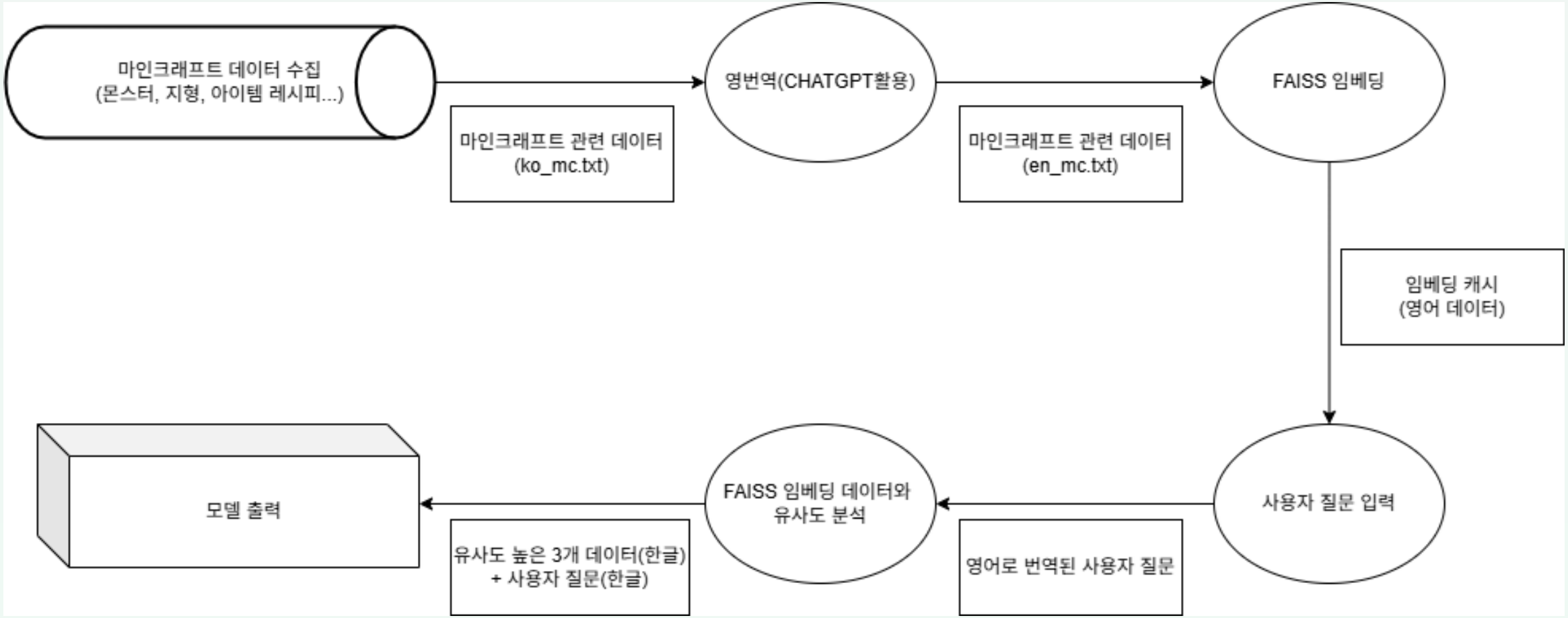
EXAONE sLM 답변

GUI 출력

시퀀스 다이어그램


사용자
데스크톱 챗봇(.exe)
구글 번역
FAISS 벡터검색
EXAONE LLM
앱 실행
챗 창 open
게임 선택
내부 변수로
게임 ID 설정
loop
[매 질문]
한글 질문 입력
질문 (ko → en) 번역 요청
영어 번역본
임베딩 생성 + 유사도 검색
상위 3개 영어 청크 index
대응 한글 청크 매핑
[CONTEXT(ko3개)] + [QUESTION(ko)] 프롬프트
답변(ko)
답변 출력
창 종료
사용자
데스크톱 챗봇(.exe)
구글 번역
FAISS 벡터검색
EXAONE LLM

데이터 플로우 다이어그램



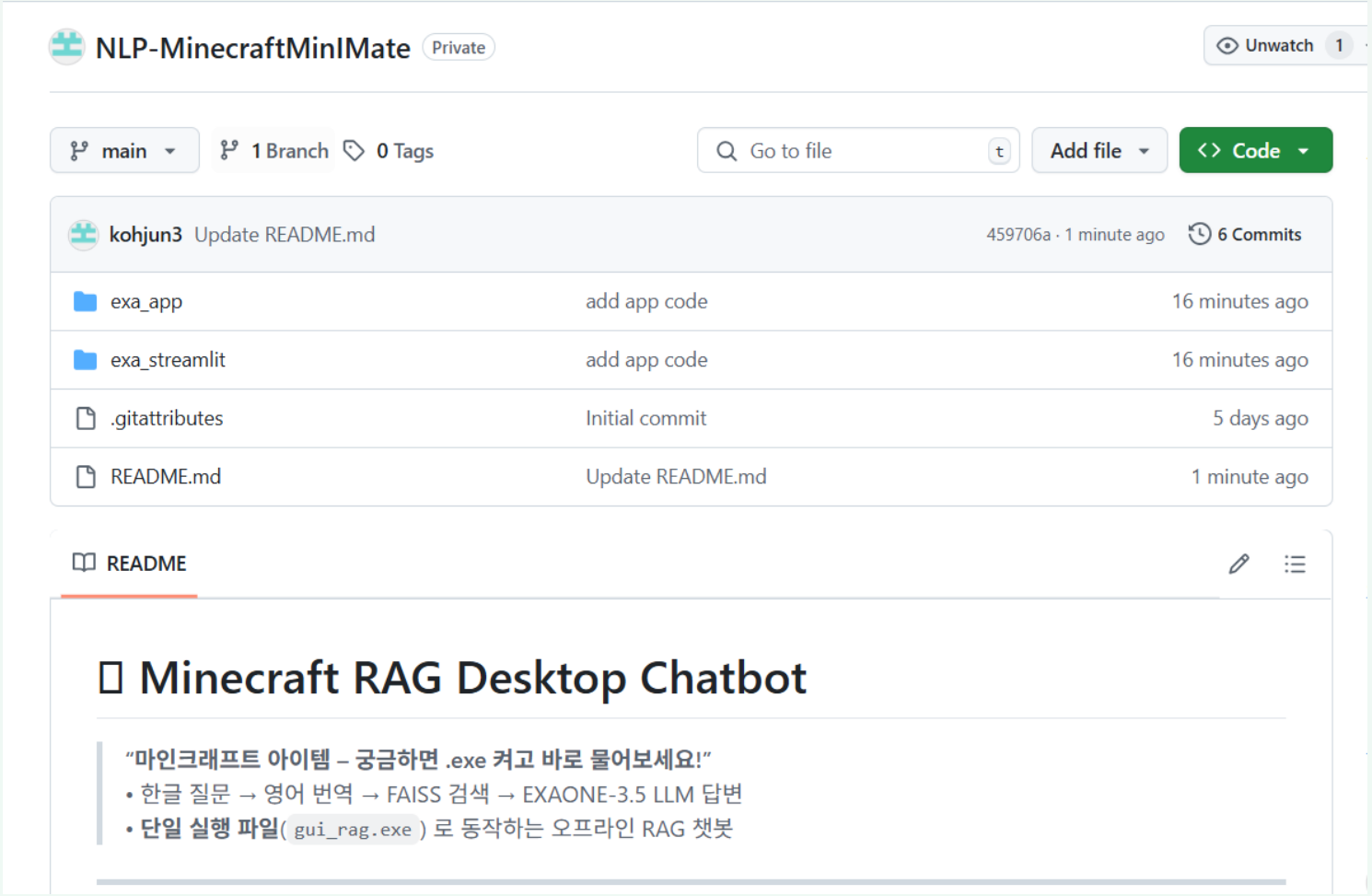
07. 협업 방식 소개

협업 방식 - Github

Github 기반 코드 관리

- **sLM** 테스트용 **Streamlit** 코드

- 실제 최종 결과물 **app** 코드



협업 방식 - **Slack**

Slack 기반 소통

- 실시간 대화
- 데이터 공유
- 진행 상황 공유

08. 개발 과정

데이터 수집

4.1.1. 평원

완만한 언덕이 주를 이루는 평탄한 생물 군계이다. 시야가 탁 트여 있기 때문에 용암과 같은 특이한 지형을 찾기가 쉽다. 대개 Y=64 부근에 평탄한 지형으로 형성되지만 때로는 목초지처럼 고원을 형성하기도 한다.

잔디 블록으로 이루어진 지표면 위에 키 작은 잔디, 키 큰 잔디, 덤불이 자라 있으며, 낮은 확률로 참나무가 한 그루씩 자라 있다. 생성되는 꽃으로는 민들레와 양귀비뿐만 아니라 선애기별꽃, 데이지, 수레국화도 있다. 돼지, 양, 소, 닭, 말, 당나귀가 생성되며, 마을과 약탈자 전초기지가 생성된다.

"item_name": "평원",
"description": "완만한 언덕이 주를 이루는 평탄한 생물 군계이다. 시야가 탁 트여 있기 때문에 용암과 같은 특이한 지형을 찾기가 쉽다. 대개 Y=64 부근에 평탄한 지형으로 형성되지만 때로는 목초지처럼 고원을 형성하기도 한다.
잔디 블록으로 이루어진 지표면 위에 키 작은 잔디, 키 큰 잔디, 덤불이 자라 있으며, 낮은 확률로 참나무가 한 그루씩 자라 있다. 생성되는 꽃으로는 민들레와 양귀비뿐만 아니라 선애기별꽃, 데이지, 수레국화도 있다. 돼지, 양, 소, 닭, 말, 당나귀가 생성되며, 마을과 약탈자 전초기지가 생성된다.",
"tags": ["평야, 온대"],

"item_name": "해바라기 평원",
"description": "평원의 변종으로, 해바라기가 생성되는 유일한 생물 군계이다.",
"tags": ["평야, 온대"],

마인크래프트 위키에서 몬스터, 지역, 아이템 조합법, 아이템 사용처 등의 정보를

tabular data 형태(이름, 설명, **tag**)로 정리하여 **txt**파일로 작성

데이터 수집


```
"item_name": "Plains",
"description": "A flat biome mainly composed of gentle hills. Due to its wide open view, it is easy to spot unusual terrain features like lava pools. It usually forms flat terrain around Y=64, but sometimes forms plateaus like meadows.₩nShort grass, tall grass, and bushes grow on the grass block surface, and oak trees may occasionally grow individually. Flowers that can spawn include not only dandelions and poppies, but also azure bluets, oxeye daisies, and cornflowers. Pigs, sheep, cows, chickens, horses, and donkeys spawn here, along with villages and pillager outposts.",
"tags": ["Plains", "Temperate"]


"item_name": "Sunflower Plains",
"description": "A variant of the plains biome, and the only biome where sunflowers generate.",
"tags": ["Plains", "Temperate"]
```


얻은 데이터를 영어로 번역

- 기존에는 한국어 데이터를 그대로 사용했지만 한국어 임베딩 모델의 한계로 인해 '질문 – 답변' 매칭의 결과가 만족스럽지 못했음
- 영어로 번역 후 '질문 – 답변' 매칭의 결과가 눈에 띄게 만족스러워짐

sLM & 임베딩모델 설정 및 양자화

```
EMB_MODEL_NAME       = "sentence-transformers/all-mpnet-base-v2"
CHAT_NAME            = "LGAI-EXAONE/EXAONE-3.5-2.4B-Instruct"
DEVICE_GPU           = torch.device("cuda" if torch.cuda.is_available() else "cpu")
DEVICE_CPU           = torch.device("cpu")
CACHE_DIR            = ".exa_bge_rerank_cache"
os.makedirs(CACHE_DIR, exist_ok=True)
translator = GoogleTranslator(source='ko', target='en')
```

sLM : EXAONE-3.5-2.4B-Instruct

Embedding : sentence-transformers / all-mpnet-base-v2

```
def load_llm_only():
    bnb = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_use_double_quant=True,
        bnb_4bit_quant_type='nf4',
        bnb_4bit_compute_dtype=torch.bfloat16,
        llm_int8_enable_fp32_cpu_offload=True
    )
    tok_c = AutoTokenizer.from_pretrained(CHAT_NAME, trust_remote_code=True)
    mdl_c = AutoModelForCausalLM.from_pretrained(
        CHAT_NAME, trust_remote_code=True,
        quantization_config=bnb, device_map={'':0}
    )
    return tok_c, mdl_c
```

sLM 모델 4bit 양자화

- 메모리 1/4, 속도는 비슷, 정밀도 손실 최소화

사용자 질문 번역

```python
# ──────────────────────────── exa_translate() 대신 쓸 함수 ────────────────────────────
def translate_ko2en(text: str) -> str:
    """
    deep_translator 기반 한→영 번역.
    실패하면 원문 반환.
    """
    try:
        return translator.translate(text).strip()
    except Exception as e:
        st.warning(f"번역 오류: {e}")
        return text
```

사용자 질문 영어 번역

- 영어로 번역된 데이터와 유사성 비교를 위해 사용자 질문도 영어로 변역
- 번역도 EXAONE을 사용하는 것을 고려, 시간적 손실이 커 질문 번역은 단순하게 구글 번역 기반으로 번역

태그 부여

```
query_en = translate_ko2en(query_ko)
TAG_LIST = [
    "Crafting", "Cooking", "Smelting/Fuel", "Enchanting Resource", "Potion",
    "Diamond Acquisition", "Iron Ore Acquisition", "Gold Ore Acquisition", "Coal Acquisition",
    "Redstone Dust Acquisition", "Lapis Lazuli Acquisition", "Emerald Acquisition",
    "Overworld", "Nether", "End", "Plains", "Desert", "Forest", "Ocean", "Cave",
    "Mountain/Hills", "Underground", "Melee Combat", "Ranged Combat", "Projectile",
    "Mob Drop", "Animal Drop", "Hostile", "Passive", "Farming", "Breeding",
    "Animal Feed", "Fishing", "Bone Meal Production", "Redstone Signal Generator",
    "Redstone Signal Transmission", "Redstone Signal Comparison",
    "Redstone Signal Amplification and Delay", "Automatic Item Collection and Transport",
    "Inventory Management", "Building", "Decoration", "Light Source",
    "Teleportation", "Villager Trading"
]
```

번역된 질문에 대해 간단하게 태그 부여

- 존재하는 태그 중 최소 2개에서 4개의 태그를 부여하도록 설정
- 이를 통해 짧거나 이해하기 어려운 모호한 질문에 대해 부착된 태그를 함께 사용해 유사도 비교, 더 좋은 결과 획득

최종 입력 데이터 생성

```
q_vec = st.session_state.embedder_cpu.embed_query(query_en_with_tags).reshape(1, -1)
D,I=st.session_state.en_index.search(np.array(q_vec,dtype='float32'),k=min(50,st.session_state.en_index.ntotal))
TOP_CTX = 3
ko_contexts = []
for block_idx in I[0][:TOP_CTX]:
    if 0 <= block_idx < len(st.session_state.ko_blocks):
        ko_contexts.append(st.session_state.ko_blocks[block_idx])
```

- 영어 질문+태그를 임베딩
- **en_index.search()**로 유사도(내적) 상위 **k**개의 영어 블록 인덱스 **I**와 점수 **D** 획득
- **I[0][:TOP_CTX]** → 영어·한글 파일에서 같은 블록 인덱스 사용
- **ko_contexts**에 저장한 한글 청크를 **"[CONTEXT]"** 아래에 삽입
- 사용자의 한국어 질문 **query_ko**를 **"[QUESTION]"**에 붙여 최종 프롬프트를 구성.

Electron App

- 기술 스택

분야	사용 기술
프레임워크	Electron.js – 데스크탑 앱 개발 (HTML + JS + Node.js)
프론트엔드 구성	HTML, CSS (Tailwind 일부 커스텀), JavaScript
상태 저장	Electron preload.js → `window.electronAPI`, `localStorage`
API 통신	FastAPI 서버 (`rag_server.py`)와 RESTful 방식으로 통신 (`fetch` API 사용)
보조 라이브러리	Supabase.js (인증), TTS/STT 지원 예정
UI 동작 구성	각 페이지는 독립적인 `.html` 파일로 구성: `login.html`, `index.html`, `mode-select.html`, `chat-mode.html`, `voice-mode.html` 등

- 앱 구조 요약

```
electron-app/
├── assets/                 # 아이콘, 배경 이미지 등
├── src/                    # 클라이언트 로직 (JS)
│   ├── navigation.js       # 페이지 이동
│   ├── apiClient.js        # API 호출 정리
│   ├── supabaseClient.js   # Supabase 연결
│   └── ...
├── styles.css              # 공통 CSS
├── main.js                 # Electron 진입점 (main 프로세스)
├── preload.js              # Electron - 웹간 안전한 bridge
├── *.html                  # 페이지들
├── start.bat               # Electron 앱 실행
├── serverstart.bat         # FastAPI 서버 실행
└── src/rag_server.py       # FastAPI 백엔드
```

09. 회고

주요 성과

- 실시간으로 자연어를 활용한 마인크래프트 정보 획득 앱 개발 성공
- 한국어 기반 임베딩 모델의 성능이 아직 부족하다는 것을 깨달음
- 현 상황에서는 가능하면 영어로 데이터를 번역 후 영어 임베딩 모델 활용이 좋다는 것을 알게됨
- 유사도 비교를 더 명확히 하기위해 **sLM**을 활용한 태깅 작업의 필요성을 알게됨

한계 및 개선 방안

- 마인크래프트에 대한 데이터 추가 필요 (특정 상황별 수행할 일)
- 데이터 추가에 따른 태그 추가 필요
- 다국어 기능 지원(영어 등)
- 더 많은 게임 지원

회고

PM 고석현 “데이터와 sLM만 잘 찾으면 바로 될 줄 알았는데 생각보다 많은 고민과 세부 요소에 대한 이해가 필요하다는것을 깨달았습니다.”

Frontend 김태호 “sLM 을 사용하여 유의미한 앱을 만들어 볼 수 있는 귀중한 경험이었고, 버전 충돌, 모델 용량 문제를 겪으면서 현업에서 마주칠 이슈들을 생생히 경험할 수 있었습니다.”

Data Engineer 위형빈 “나무위키의 방대한 정보를 체계적으로 정제하고, EXAONE 3.5와 연동해 실시간 답변을 구현하는 과정을 통해 데이터 엔지니어링과 모델링의 시너지를 경험했습니다.”

AI Engineer 윤주완 “게임 도메인에서 프로젝트를 진행하며, 실제 게임 이용자의 입장에서 직접 플레이하며 결과물을 테스트하는 과정을 겪었는데, 이러한 사용자 관점에서의 검증 과정은 매우 유의미하고 즐거운 경험이었습니다.”

Data Engineer 차유원 “마지막 프로젝트를 성공적으로 완성해서 기쁘네요. 유능한 팀원들 덕분에 유종의 미를 거두었어요.”

Thanks for your attention