Functional Specification

**Open Source Software Development Contest**

**Team Name : 4-Nyang DAN (사냥단)**

**Project Name : NOQUESTION**

임석범, 박유빈, 변윤성, 정하은

Dankook Univ.

Mobile System Engineering

**목 차**

**1. 문서 개요 (Purpose & Audience) 1**

**2. 용어 정의 (Glossary) 3**

**3. 시스템 개요 (System Overview) 6**

3-1. 전체 아키텍처 6

3-2. 모듈별 역할 6

3-3. 처리 흐름 (End-to-End 시퀀스) 6

**4. 기능 요구사항 (Functional Requirements) 6**

4-1. Video Capture & Adaptive Sampling 6

4-2. OCR + PII Detection 6

4-3. Image Explanation LLM 6

4-4. Vector DB & RAG 6

4-5. Prediction LLM 6

4-6. Answering LLM 6

4-7. UI/UX 6

**5. 비기능 요구사항 (Non-Functional Requirements) 6**

5-1. 성능 (Performance) 6

5-2. 보안 & 개인정보 (Security & Privacy) 6

5-3. 확장성 & 가용성 (Scalability & Availability) 6

5-4. 로깅 & 모니터링 6

**6. 비기능 요구사항 (Non-Functional Requirements) 6**

6-1. REST API 스펙 6

6-2. 이벤트 처리 6

**7. 테스트 전략 (Testing Strategy) 6**

7-1. 단위 테스트 6

7-2. 통합 테스트 6

7-3. E2E 테스트 6

**8. 배포 & 버전 관리 (Deployment & Versioning) 6**

**9. 추후 개선 항목 (Future Enhancements) 6**

**1. 문서 개요 (Purpose & Audience)**

본 문서는 **NoQuestion** 프로젝트의 전반적인 구조, 요구사항, 기능 정의를 명확히 기술하기 위해 작성되었다. NoQuestion은 **사용자의 작업 맥락을 이해하고, 미래 행동을 예측하며, 선제적으로 질문을 추천하는 지능형 보조 시스템**이다.

기존 LLM 기반 어시스턴트는 주로 다음과 같은 한계를 가진다:

* **현재 작업 중심:** 사용자가 요청한 시점의 화면이나 텍스트에만 의존하여 응답을 제공한다.
* **과거 문맥 부족:** 사용자의 과거 작업 이력이나 맥락을 충분히 고려하지 못한다.
* **미래 예측 부재:** 앞으로 사용자가 수행할 작업을 예측하지 못하며, 이에 따라 필요한 정보나 가이드를 제공하지 않는다.

**NoQuestion**은 이러한 문제를 해결하고, 사용자 경험을 한 단계 발전시키는 것을 목표로 한다.

본 시스템은 다음과 같은 핵심 기능을 제공한다:

1. **작업 이력 기반 이해:** 영상 기반 작업 기록을 분석하고 Vector DB에 저장하여, 과거 맥락을 포함한 질의응답 제공
2. **RAG(Retrieval-Augmented Generation) 통합:** LLM과 Vector DB를 결합하여, 과거 데이터와 현재 상태를 모두 활용한 고도화된 응답 생성
3. **미래 행동 예측:** 현재 작업 환경을 기반으로 사용자가 다음에 수행할 가능성이 높은 작업을 예측
4. **질문 추천 시스템:** 예측된 작업에 따라, 사용자가 가질 가능성이 높은 질문을 3가지 이상 제안하여 사전 준비 가능
5. **PII(개인정보) 보호 강화:** 영상에서 추출한 텍스트 내 민감 정보를 탐지하고 안전하게 처리

본 문서는 다음과 같은 이해관계자를 대상으로 한다:

* **제품 관리자 (PM)**: 프로젝트 목표와 범위를 이해하고 기능 요구사항을 확인
* **AI 및 백엔드 개발자**: OCR, PII 탐지, Vector DB 연동, RAG 기반 서비스 구현
* **AI/ML 엔지니어**: 이미지 설명 LLM, 예측 LLM, Answering LLM 설계 및 최적화
* **프론트엔드 개발자**: UI/UX 구현 (질문 추천 UI, 사이드바 챗 인터페이스)
* **보안 담당자**: PII 마스킹, 데이터 암호화, 접근 제어 정책 검토
* **QA 및 운영 수행자**: 기능 테스트, 성능 측정, 모니터링, 배포 검증

**2. 용어 정리 (Glossary)**

|  |  |
| --- | --- |
|  | **정의** |
| **NoQuestion** | 본 프로젝트의 이름. 사용자 작업 맥락을 이해하고, 미래 행동을 예측하며, 예상 질문을 사전에 제시하는 지능형 보조 시스템. |
| **Agent 앱** | Electron + React 기반 데스크톱 앱. 사용자 PC에서 화면 캡처, 로컬 OCR/PII 필터링, 서버 업로드 수행. |
| **AI 분석 서버** | FastAPI 기반. BLIP-2, PaddleOCR, SBERT, GPT 모델을 활용해 이미지 설명, PII 탐지, 질문 예측 및 응답을 처리. |
| **백엔드 서버** | Spring Boot 기반. 이미지 수신, Adaptive Sampling, 세션 관리, AI 서버 연동 담당. |
| **Electron** | 데스크톱 애플리케이션을 웹 기술(React, Node.js)로 구현 가능하게 하는 프레임워크. |
| **Image Explanation LLM** | |  | | --- | |  |  |  | | --- | | 입력 이미지(사용자 화면)를 분석하고 해당 화면의 맥락을 텍스트로 설명하는 역할을 하는 LLM. | |
| **Prediction LLM** | 현재 화면과 과거 맥락(Vector DB 기반)을 바탕으로 사용자의 **다음 행동** 및 **예상 질문 Top-3**를 예측하는 모델. |
| **Answering LLM** | |  | | --- | |  |  |  | | --- | | 사용자의 실제 질문에 대해, RAG(Retrieval-Augmented Generation) 기반으로 응답을 생성하는 모델. | |
| **RAG** | Retrieval-Augmented Generation, 외부 지식 소스를 검색 후, LLM이 생성 과정에서 해당 정보를 활용하는 방식. 본 시스템에서는 Vector DB에서 검색된 맥락을 기반으로 응답을 제공. |
| **Vector DB** | 텍스트를 임베딩(벡터화)하여 저장·검색하는 데이터베이스. 과거 사용자 작업의 설명(LLM Output)을 저장하고, 유사도 기반 검색을 제공. |
| **Adaptive Sampling** | 사용자 PC 화면을 영상 스트림으로 캡처한 후, 특정 조건(입력 이벤트, Scene Change, Motion, 주기적)을 기준으로 효율적으로 프레임을 추출하는 방법. |
| **Scene Change Detection** | 영상에서 장면 전환이 발생하는 시점을 감지하는 기술. 변화가 큰 프레임만 선택하여 분석 비용을 줄임. |
| **Motion-Based Sampling** | 화면 내 움직임의 크기를 분석하여, 변화가 많은 구간에서 더 많은 이미지를 추출하는 방식. |
| **OCR** | Optical Character Recognition, 이미지 내 텍스트를 인식하고 추출하는 기술. 본 프로젝트에서는 UI 요소나 메시지의 텍스트 분석에 사용. |
| **PII** | Personally Identifiable Information, 개인 식별이 가능한 정보(비밀번호, 주민등록번호, 신용카드 번호 등). OCR 후 텍스트에서 탐지하며, 마스킹 처리 적용. |
| **질문 추천** | Question Prediction, 사용자의 미래 작업을 기반으로, 예상 질문 3개를 미리 제안하는 기능. |
| **UI Overlay** | 화면 좌측 상단에 표시되는 작은 영역. 예상 행동과 질문을 실시간 표시하며, 클릭 시 채팅으로 연결. |
| **Sidebar Chat** | 화면 오른쪽에 위치하는 인터페이스. 사용자가 직접 질문하거나 예상 질문을 클릭해 LLM 응답을 확인. |
| **Confidence Score** | |  | | --- | |  |  |  | | --- | | Prediction LLM이 예측한 행동/질문의 신뢰도를 나타내는 값(0~1). 기준값 이상일 때만 UI에 노출.  (아직 명확하지는 않음.) | |

표 1. 용어 정리

**3. 시스템 개요 (System Overview)**

**3-1. 전체 아키텍처**

**(수정 전 버전)**

본 시스템은 사용자 화면에서 캡처한 영상을 기반으로**, Adaptive Sampling → OCR + PII Detection → Image Explanation LLM → Vector DB 저장 → Prediction LLM → Question Recommendation → Answering LLM** 순으로 처리된다.

아래 그림은 **NoQuestion AI 시스템의 구조**를 보여준다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**(수정 버전) : 수정 전 버전이랑 잘 합쳐야 될듯**

**NoQuestion 시스템은** **Agent 앱(Electron + React)**, **백엔드 서버(Spring Boot)**, **AI 분석 서버(FastAPI)** 로 구성되며, 각 모듈은 다음 역할을 담당한다:

* **Agent 앱**: 사용자 PC 화면 캡처, 로컬 OCR & PII 필터링, 이미지 업로드, UI 제공
* **백엔드 서버**: Adaptive Sampling, 이미지 저장, 세션 관리, AI 분석 서버 연동
* **AI 분석 서버**: 이미지 분석, 질문 예측, 응답 생성

아키텍처는 다음과 같다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**3-2. 모듈별 역할**

|  |  |
| --- | --- |
| **모듈** | **주요 역할** |
| **Agent 앱 (Electron + React)** | - 사용자 로그인 및 인증 (JWT)  - 화면 캡처 (주기 + 이벤트)  - 로컬 OCR 및 민감정보(PII) 필터링  - 백엔드 API로 이미지 업로드  - UI 제공 (질문 추천 + 응답) |
| **백엔드 서버 (Spring Boot)** | - Adaptive Sampling 로직 적용  - 이미지 및 메타데이터 저장 (S3, PostgreSQL)  - 세션 관리 (사용자별 작업 히스토리)  - AI 분석 서버와 연동 |
| **AI 분석 서버 (FastAPI)** | - 이미지 의미 분석 (GPT) - OCR + PII 탐지 (PaddleOCR) - 벡터화(Open AI, Embedding) 후 Vector DB 저장 - 질문 예측 (Prediction LLM) - 질문 응답 생성 (Answering LLM, RAG 기반) |

**3-3. 처리 흐름 (End-to-End 시퀀스)**

**NoQuestion**의 전체 동작 과정은 다음과 같다:

1. **앱 실행 및 로그인**
   * 사용자가 **Agent 앱** 실행 → JWT 인증 기반 로그인 완료
2. **화면 캡처 시작**
   * 사용자가 "모니터링 시작" 버튼 클릭
3. **로컬 OCR + PII 필터링**
   * 민감정보가 탐지되면 마스킹 처리 후 백엔드로 전달
4. **이미지 업로드 (Spring Boot)**
   * API: /upload-screenshot 호출
5. **Adaptive Sampling**
   * 백엔드에서 이미지 변화율/텍스트 변화율 기준으로 저장 여부 판단
6. **이미지 저장 및 메타데이터 기록**
   * AWS S3 저장, PostgreSQL에 세션 데이터 저장
7. **AI 분석 요청 (백엔드 → FastAPI)**
   * 백엔드가 AI 분석 서버로 REST 요청 전송
8. **AI 분석 단계 (FastAPI)**
   * BLIP-2로 이미지 설명 생성
   * OCR + PII 필터링
   * 임베딩 벡터화 후 FAISS 저장
   * 질문 예측 (Top-3), 질문 응답 생성
9. **백엔드 → Agent 앱 결과 전달**
   * 예측 질문 3개 + Confidence Score 포함
10. **UI 표시 (Agent 앱)**
    * 좌측 상단 Overlay: 예상 질문 Top-3 표시
    * 우측 Sidebar Chat: 질문 클릭 시 응답 표시

**4. 기능 요구사항 (Functional Requirements)**

**4‑1. Agent 앱 (Electron + React)**

**목적 : 사용자 PC에서 화면을 캡처하고, 로컬 OCR + PII 필터링 수행 후 백엔드 서버로 이미지 전송 및 UI 제공.**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **기능** | **설명** | **조건** |
| **FR-AG-1** | 로그인 및 인증 | 이메일/비밀번호 로그인 → JWT 발급 | HTTPS 적용 |
| **FR-AG-2** | 화면 캡처 | 주기 5초 또는 입력 이벤트 시 즉시 수행 | CPU ≤ 10% |
| **FR-AG-3** | 로컬 OCR 수행 | PaddleOCR 혹은 Tesseract 변형 사용, 문자별 좌표 위치 포함 | 정확도 ≥ 90% |
| **FR-AG-4** | PII 필터링 | 주민번호, 카드번호, 비밀번호 등 정규식 기반 탐지 → 마스킹 | 탐지율 ≥ 95% |
| **FR-AG-5** | 이미지 업로드 | Upload-screenshot,API 호출, 마스킹 된 이미지 전송 | 이미지 ≤ 5 MB |
| **FR-AG-6** | 설정 UI | 캡처 주기, 학습 토큰 입력, Dark/Light 모드 설정 제공 | 반응형 UI |

**4‑2. 백엔드 서버 (Spring Boot)**

**목적: Adaptive Sampling 적용, 이미지 저장, 메타데이터 관리, AI 분석 서버 연동.**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **기능** | **설명** | **조건** |
| **FR-BE-1** | Adaptive Sampling 판단 | 이전 이미지 대비 텍스트 변화율 ≥ 20% 또는 SSIM ≤ 0.75 시 저장 | 처리 ≤ 300 ms |
| **FR-BE-2** | 이미지 저장 | DB사용, 저장 key = userID + timestamp  (Local DB) | 암호화 at rest |
| **FR-BE-3** | 메타데이터 저장 | PostgreSQL에 userID, timestamp, 이미지 해시 등 저장 | 정합성 확보 |
| **FR-BE-4** | AI 분석 요청 | FastAPI /analyze 호출, 이미지 URL 포함 | Timeout ≤ 10 초 |
| **FR-BE-5** | 질문/응답 기록 | 예상 질문 Top‑3 및 GPT 응답 저장 | 로그 보존 세션 종료시 까지 |

**4‑3. AI 분석 서버 (FastAPI + OpenAI API)**

**목적: 이미지 설명, 텍스트 추출, 임베딩, 질문 예측 및 응답 생성 등 AI 처리를 수행.**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **기능** | **설명** | **조건** |
| **FR-AI-1** | 이미지 설명 (OpenAI Vision) | GPT-4 Vision 혹은 GPT-4o모델 활용. 이미지에서 화면 상황 설명 | 응답 ≤ 2 초 |
| **FR-AI-2** | OCR + 텍스트 위치 추출 | PaddleOCR 또는 유사 엔진 사용, 텍스트와 좌표 함께 반환 | 정확도 ≥ 90% |
| **FR-AI-3** | PII 마스킹 확인 | OCR 결과 텍스트에 PII 포함 여부 검사, 민감정보 flag 처리 | 탐지율 ≥ 99% |
| **FR-AI-4** | 무익 텍스트 제거 | UI 설명 제외 불필요 문구 제거(ex. 브라우저 로딩 메시지) | 노이즈 ≤ 5% |
| **FR-AI-5** | 텍스트 임베딩 (OpenAI API) | text-embedding-3-large 사용 (조정 가능), 임베딩 크기 3072차원 또는 축소 | 유사도 정밀도 우수 OpenAI |
| **FR-AI-6** | 벡터 저장 (Vector DB) | FAISS 또는 유사 저장소 사용, 최신 순 Top‑K=8 검색 지원 | 검색 latency ≤ 200ms |
| **FR-AI-7** | 질문 예측 (Prediction LLM) | GPT-4o 모델 기반, 현재 설명 + 유사 컨텍스트 사용, Top‑3 질문 생성 | Confidence ≥ 0.7,  응답 ≤ 2 s |
| **FR-AI-8** | 질문 응답 생성 (Answering) | GPT-4o API 사용, RAG 기반 응답 생성 및 리소스 출처 포함 | 초기 응답 ≤ 2 s,  최종 ≤ 15 s |

**4‑4. UI/UX (Agent 앱 내 Electron + React)**

**목적: 사용자에게 타임라인, 추천 질문, GPT 응답, 설정을 직관적으로 제공.**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **기능** | **설명** | **조건** |
| **FR-UI-1** | 타임라인 뷰 | 캡처된 이미지와 설명이 시간 순으로 정렬되어 보여짐 | Lazy loading 적용 |
| **FR-UI-2** | 추천 질문 Overlay | 예상 질문 Top‑3와 Confidence 표시, 클릭 시 Sidebar로 전달 | 실시간 표시 |
| **FR-UI-3** | Sidebar Chat 인터페이스 | 클릭 또는 직접 질문 입력 → GPT 응답 Markdown 형태로 렌더링 | 코드/표 지원 |
| **FR-UI-4** | 세션 요약 보고서 | 하루/세션 기준 요약: 추천 행동, 주요 캡처 상황 요약 제공 | PDF 저장 가능 |
| **FR-UI-5** | 설정창 | 캡처 주기, API 키 입력, UI 모드 전환 제공 | 반응형 및 접근성 UI 고려 |

**5. 비기능 요구사항 (Non-Functional Requirements)**

본 장에서는 NoQuestion 시스템의 비기능적 요구사항을 정의한다. 여기에는 **성능**, **보안 & 개인정보**, **확장성 & 가용성**, **로깅 & 모니터링**을 포함한다.

**5‑1. 성능 (Performance)**

시스템은 실시간성 요구를 만족해야 하며, 주요 성능 목표는 다음과 같다:

|  |  |
| --- | --- |
| **항목** | **요구사항** |
| **화면 캡처 → 백엔드 업로드** | 각 프레임은 **1초 이내** 전송 완료 |
| **Adaptive Sampling 판단** | 백엔드에서 샘플링 여부 결정 ≤ **300ms** |
| **AI 분석 서버 응답** | 이미지 설명 + OCR 완료 ≤ **2초** |
| **질문 예측 (Prediction LLM)** | Top-3 질문 생성 ≤ **2초** |
| **질문 응답 (Answering LLM)** | 초기 응답 ≤ **2초**, 최종 응답 ≤ **15초** |
| **UI 렌더링** | Overlay 및 Sidebar UI 갱신 지연 ≤ **200ms** |
| **벡터 검색 (FAISS)** | Top-K 검색 ≤ **200ms**, K=8 |
| **동시 사용자 처리** | 1,000명 동시 접속 시 평균 응답 시간 ≤ **3초** |

**5‑2. 보안 & 개인정보 (Security & Privacy)**

시스템은 민감정보 보호와 데이터 안전성을 보장해야 한다:

|  |  |
| --- | --- |
| **항목** | **요구사항** |
| **PII 처리** | OCR 결과에서 **PII 탐지 후 마스킹** (비밀번호, 카드번호, 주민번호 등) |
| **전송 보안** | 모든 데이터 전송 시 **TLS 1.3 이상** 적용 |
| **저장 보안** | 이미지 및 메타데이터는 **AES-256** 암호화 저장 (AWS S3 at-rest encryption) |
| **인증** | JWT 기반 인증 (Access Token + Refresh Token) |
| **접근 제어** | RBAC(Role-Based Access Control) 적용 |
| **로그 민감정보 차단** | PII 포함 데이터는 절대 로그에 기록하지 않음 |
| **GDPR/KISA 준수** | 사용자 데이터 삭제 요청 시 7일 이내 처리 |
| **API Key 보안** | OpenAI API 키는 환경변수/Secret Manager 관리 |

**5‑3. 확장성 & 가용성 (Scalability & Availability)**

서비스는 트래픽 증가와 장애 상황에도 안정적으로 동작해야 한다:

|  |  |
| --- | --- |
| **항목** | **요구사항** |
| **확장성** | 백엔드(Spring Boot), AI 분석(FastAPI), DB 서비스는 **Kubernetes 기반 오토스케일링** 지원 |
| **고가용성** | 백엔드 및 AI 서버는 **3개 AZ(Availability Zone)** 분산 배포 |
| **SLA** | 서비스 가용성 **99.9% 이상** |
| **DB 복제** | PostgreSQL Read Replica + 벡터 DB(FAISS) 백업 |
| **장애 대응** | 장애 감지 후 **30초 이내 Failover** |
| **캐싱** | CloudFront/CDN 활용, 정적 리소스 응답 시간 ≤ **100ms** |

**5‑4. 로깅 & 모니터링 (Logging & Monitoring)**

운영 및 장애 대응을 위해 모든 주요 이벤트를 추적 가능해야 한다:

|  |  |
| --- | --- |
| **항목** | **요구사항** |
| **로그 수집** | Elastic Stack(ELK) 또는 Loki 기반 로그 수집 |
| **메트릭 모니터링** | Prometheus + Grafana 대시보드 구축 |
| **주요 KPI** | - 캡처 업로드 성공률- AI 응답 시간- PII 탐지율- 예상 질문 정확도 |
| **알림** | 임계치 초과 시 Slack/Email 알림 |
| **로그 보존 정책** | 90일 (개인정보는 익명화 후 저장) |
| **트레이싱** | OpenTelemetry 기반 분산 트레이싱 (백엔드↔AI 서버↔Agent) |

**6. API 스펙 & 이벤트 처리**

본 장에서는 **Agent 앱 ↔ 백엔드 서버 ↔ AI 분석 서버** 간 통신을 정의한다.

**6‑1. REST API 스펙**

**6-1-1. 사용자 인증**

|  |  |
| --- | --- |
| **항목** | **내용** |
| **URL** | /api/auth/login |
| **Method** | POST |
| **설명** | 사용자 로그인, JWT 발급 |
| **요청 Body** | json { "email": "user@example.com", "password": "string" } |
| **응답** | json { "accessToken": "jwt-token", "refreshToken": "jwt-refresh" } |

**6-1-2. 화면 캡처 업로드**

|  |  |
| --- | --- |
| **항목** | **내용** |
| **URL** | /api/capture/upload |
| **Method** | POST |
| **설명** | Agent 앱이 캡처 이미지 업로드 |
| **Headers** | Authorization: Bearer {JWT} |
| **요청 Body** | Multipart Form Data - file: 이미지 파일 (JPEG/PNG) - metadata: JSON 문자열 → { "userId": "string", "timestamp": "2025-07-27T12:00:00Z" } |
| **응답** | json { "status": "success", "imageId": "uuid" } |

**6-1-3. AI 분석 요청 (백엔드 → FastAPI)**

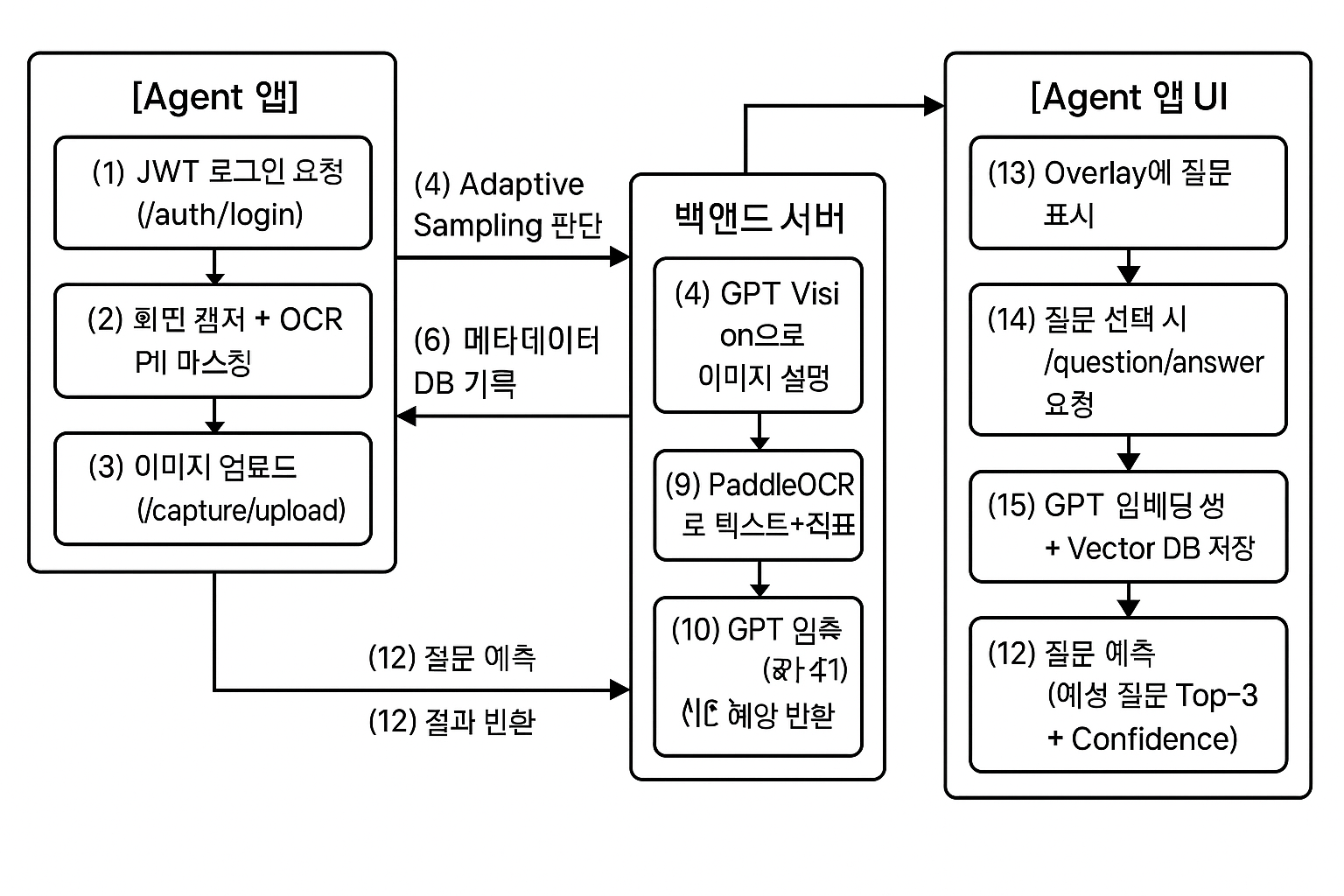
|  |  |
| --- | --- |
| **항목** | **내용** |
| **URL** | /api/analyze |
| **Method** | POST |
| **설명** | 백엔드가 AI 분석 서버에 이미지 분석 요청 |
| **요청 Body** | json { "imageUrl": "https://s3.bucket/image.jpg", "sessionId": "uuid", "userId": "uuid" } |
| **응답** | json { "description": "브라우저에서 이메일 입력 화면", "ocr": [{"text":"로그인","bbox":[10,20,50,40]}], "questions": ["질문1","질문2","질문3"], "confidence": 0.82 } |

**6-1-4. 질문 응답 요청**

|  |  |
| --- | --- |
| **항목** | **내용** |
| **URL** | /api/question/answer |
| **Method** | POST |
| **설명** | Agent 앱이 특정 질문에 대한 응답 요청 |
| **요청 Body** | json { "question": "PPT 테마 변경 방법은?", "sessionId": "uuid" } |
| **응답** | json { "answer": "디자인 탭에서 테마를 선택하면 됩니다.", "sources": ["https://support.microsoft.com/..."] } |

**6-1-5. 세션 요약 보고서 생성**

|  |  |
| --- | --- |
| **항목** | **내용** |
| **URL** | /api/session/summary |
| **Method** | GET |
| **설명** | 세션 단위 요약 보고서 반환 |
| **응답** | json { "summary": "오늘 작업: 이메일 작성, PPT 편집", "recommendations": ["PPT 단축키 학습 권장"] } |

아래는 이에 대한 전체적 구조이다.

**7. 테스트 전략 (Testing Strategy) 6**

7-1. 단위 테스트 6

7-2. 통합 테스트 6

7-3. E2E 테스트 6

**8. 배포 & 버전 관리 (Deployment & Versioning) 6**

**9. 추후 개선 항목 (Future Enhancements) 6**

위 내용은 굳이 써야 되나 싶다.

나중에 보고서 쓸때나 한번 써봐야지