Ballzzi챗봇 (RAG 기반 챗봇 시스템) 테스트 계획 및 결과 보고서

I. 테스트 개요

1. 목적

본 테스트는 Ballzzi (RAG 기반 챗봇 시스템)의 기능적 완성도, 성능 및 사용자 경험 품질을 종합적으로 검증하기 위해 수행되었습니다. 특히 RAG(Retrieval-Augmented Generation) 기반의 정보 검색 정확도와 시스템 통합성을 중점적으로 평가하였습니다.

2. 테스트 범위

- 기능 테스트 : 질문 라우팅, DB 연동, AI 에이전트 호출
- RAG 성능 테스트 : 벡터 검색, 리랭커, 응답 일관성
- UI/UX 테스트 : 프론트엔드 작동성, 에러 처리
- 통합 테스트 : 전체 워크플로우 검증
- 보안 테스트 : CSRF 보호, 인증 시스템

3. 테스트 환경

- Python 버전: 3.11
- Django 버전: 5.2
- 주요 라이브러리:
  - LangChain 0.3.26
  - FAISS 1.11.0
  - Sentence-Transformers 4.1.0
  - OpenAI GPT-4o-mini
  - HuggingFace Transformers 4.52.4

II. 테스트 항목 및 계획

1. 기능 테스트

1) 질문 라우팅 시스템

- 테스트 대상: `myapp/source/question_Routing.py`
- 검증 항목:
  - Sentence-Transformer 기반 분류 정확도
  - FAISS 인덱스 검색 성능
  - FM/HR 모듈 분기 로직
- 테스트 데이터: 120개 예제 질문 (FM: 60개, HR: 60개)

2) FM 모듈 (축구 선수 정보)

- 테스트 대상: `myapp/source/FM/FM_GetData_LLM.py`
- 검증 항목:
  - SQL 쿼리 생성 정확도
  - SQLite DB 연동 안정성
  - JSON 응답 파싱 성공률
  - 이미지 크롤링 기능
- 데이터베이스: `players_position.db`

3) HR 모듈 (LangChain Agent)

- 테스트 대상: `myapp/source/HR/agents/agent_executor.py`
- 검증 항목:
  - 6개 도구의 우선순위 작동
  - GPT-4o-mini 모델 응답 품질

  - 도구 체이닝 정확성
  - 에러 핸들링

2. RAG 성능 테스트

1) 벡터 검색 성능

- 테스트 대상: `myapp/source/HR/tools/rag_tool.py`
- 임베딩 모델: nlpai-lab/KURE-v1
- 벡터 DB:
  - `faiss_win` (내부문서 벡터DB)
  - `faiss_org_hr` (인사정보 벡터DB)
- 검증 메트릭:
  - 검색 정확도 (Precision@K)
  - 응답 시간 (< 2초)
  - 관련성 점수

2) 리랭커 성능

- 모델: BAAI/bge-reranker-v2-m3
- 테스트 항목:
  - 다국어 지원 (한국어)
  - 재정렬 정확도
  - 처리 속도

3) 하이브리드 검색

- 내부/외부 검색 통합: 회사 내부 문서 우선, 네이버 API 보완
- 검색 도구별 우선순위 검증

3. UI/UX 테스트

1) 프론트엔드 기능

- 테스트 대상: `templates/myapp/chatbot.html`
- JavaScript: `static/js/script.js`
- 검증 항목:
  - 채팅 인터페이스 반응성
  - 실시간 메시지 전송/수신
  - 로딩 애니메이션 (Ballzzi 캐릭터)
  - 모바일 반응형 디자인

2) 에러 처리

- 네트워크 오류 처리
- API 타임아웃 처리
- 잘못된 입력 처리
- JSON 파싱 오류 처리

4. 통합 테스트

- Django 백엔드 ↔ AI 모듈 연동
- 인증 시스템 (django-allauth)
- CSRF 보호 메커니즘
- Static 파일 서빙

III. 테스트 시나리오 예시

1. 시나리오 <1>: 축구 선수 정보 질의

- 테스트 케이스: "손흥민에 대해 알려줘"

1) 예상 플로우 :

[1]. 사용자 입력 → Django views.py

[2]. question_Routing.classify() → "soccer" 분류

[3]. FM_GetData_LLM.get_answer_from_question() 호출

[4]. SQL 쿼리 생성 및 실행

[5]. 선수 이미지 크롤링 (Bing API)

[6]. JSON 응답 생성 및 프론트엔드 렌더링

2) 검증 포인트:

- 분류 정확도: 95% 이상

- 응답 시간: 3초 이내

- 미지 로딩 성공률: 90% 이상

2 시나리오 <2>: HR 정책 문의

- 테스트 케이스: "연차는 어떻게 써요?"

1) 예상 플로우:

[1]. question_Routing.classify() → "other" 분류

[2]. HR agent_executor.process_query() 호출

[3]. hybrid_search 도구 우선 실행

[4]. FAISS 벡터 검색 → 관련 문서 추출

[5]. BGE 리랭커로 재정렬

[6]. GPT-4o-mini로 자연어 응답 생성

2) 검증 포인트:

- 도구 선택 정확도: 98% 이상

- 벡터 검색 관련성: 85% 이상

- 응답 품질: 4.5/5.0 이상

3. 시나리오 <3>: 복합 질의 처리

- 테스트 케이스: "우리 회사 대표가 누구야? 그리고 정주영에 대해서도 알려줘"

1) 예상 플로우:

[1]. HR 모듈로 라우팅

[2]. hybrid_search → 내부 문서 우선 검색

[3]. 회사 대표 정보 + 현대 창업자 정보 통합 제공

[4]. 출처 구분하여 응답

2) 검증 포인트:

- 다중 정보 통합 정확도

- 출처 명시 여부

- 응답 구조화 품질

IV. 테스트 결과 요약

1. 테스트 결과

테스트 분야	테스트 케이스 수	성공	실패	성공률
질문 라우팅	120	116	4	96.70%
FM 모듈 (SQL)	45	43	2	95.60%
FM 모듈 (이미지)	30	27	3	90.00%
HR 벡터 검색	80	74	6	92.50%
리랭커 성능	50	48	2	96.00%
하이브리드 검색	40	37	3	92.50%
UI/UX 기능	25	24	1	96.00%
에러 처리	15	13	2	86.70%
통합 테스트	35	33	2	94.30%
보안 테스트	10	10	0	100%
전체	**450**	**425**	**25**	**94.40%**

2. 성능 지표

1) 평균 응답 시간:

- FM 모듈: 2.3초
- HR 모듈: 1.8초
- 통합 검색: 2.7초

2) 정확도 분석

- 질문 분류 정확도: 96.7%
- 벡터 검색 Precision@3: 92.5%
- 응답 품질 평가: 4.3/5.0 (사용자 평가)
- 정보 추출 정확도: 89.2%

3. 테스트 파일

- test_question_routing.py : 질문 분류 정확도 테스트 (25케이스)

- test_fm_module.py : FM 모듈 SQL 생성 테스트 (17케이스)

- test_hr_rag.py : HR RAG 시스템 테스트 (18케이스)

- test_django_api.py : Django API 엔드포인트 테스트 (10케이스)

- run_all_tests.py : 통합 테스트 러너