

**SK네트웍스 Family AI과정 12기  
 모델링 및 평가 시스템 아키텍처**



**□ 개요**

* 산출물 단계 : 모델링 및 평가
* 평가 산출물 : 시스템 아키텍처
* 제출 일자 : 2025. 07. 18.
* 깃허브 경로 : https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN12-FINAL-2TEAM
* 작성 팀원 : 박진양

| **컴포넌트 다이어그램** | * **구성 요소**   + 클라이언트 환경   + 로컬 서버   + 비즈니스 로직 계층   + AI 도구 계층   + 데이터 계층   + 외부 서비스 * **설명**   + **클라이언트 환경** : 사용자가 Chrome 브라우저를 통해 [Next.js](http://next.js) Frontend에 접속하고, Redux Store로 상태를 관리하며, HTTP와 WebSocket으로 로컬 서버와 통신   + **로컬 서버** : FastAPI Web Server가 uvicorn으로 실행되어 REST API를 제공하고, ChatBot Server가 별도로 실행되어 WebSocket 실시간 채팅을 담당   + **비즈니스 로직 계층** : 도메인별 템플릿이 각각의 비스니스 로직을 처리하고, 서비스가 데이터베이스, 캐시, 큐 등의 인프라를 관리   + **AI 도구 계층** : LangGraph Router가 GPT-4o 모델과 15개의 금융 분석 도구를 조율하여 고도화된 AI 분석을 제공   + **데이터 계층** : MySQL로 구조화된 데이터를 샤딩하여 저장하고, Redis로 세션과 캐시를 고속 처리   + **외부 서비스** : 실시간 금융 데이터와 뉴스를 외부 API를 통해 수집하여 AI 분석에 활용 |
| --- | --- |
| **시퀀스 다이어그램** | * **참여자 (Actors)**   + **사용자** : 시스템 사용자이며, 투자 질문 입력 및 결과를 확인   + [**Next.js**](http://next.js) **Frontend(F)** : 클라이언트 애플리케이션이며, UI 랜더링 및 상태 관리   + **FastAPI Web Server(W)** : 메인 백엔드 서버이며 인증 및 API 라우팅   + **ChatBot Server(C)** : 실시간 채팅 서버   + **LangGraph Router(R)** : AI 도구 선택 및 실행 관리   + **AI Tools Ensemble(T)** : AI 분석 도구 집합, 금융 데이터 분석 및 예측   + **External APIs(API)** : 외부 데이터 제공자의 역할을 하며, 금융 데이터 & 뉴스 & 경제 지표를 제공   + **Database Services(D)** : 데이터 저장소의 역할을 하며, MySQL/Redis를 통한 데이터 관리 * **주요 흐름**   + **1. 인증 및 연결 단계**      - 사용자 인증 확인     - WebSocket 연결 수립     - 세션 검증   + **2. 질문 분석 단계**     - 사용자 질문 수신     - LangGraph를 통한 질문 분석     - 적절한 AI 도구 선택   + **3. 병렬 분석 단계**     - 다중 AI 도구 동시 실행     - 외부 API 데이터 수집     - 분석 결과 통합   + **4. 실시간 응답 단계**     - GPT-4o 종합 분석     - 실시간 토큰 스트리밍     - 최종 결과 표시 |
| **액티비티 다이어그램** | * **구성**   + **시작/종료 노드** : 프로세스의 시작점과 끝점   + **결정 노드** : 조건 분기 및 검증 단계   + **액션 노드** : 실제 작업을 수행하는 단계   + **병합 노드** : 병렬 프로세스 결과 통합   + **에러 처리** : 오류 상황 대응 로직 * **주요 액션 노드**   + **입력 처리 노드**     - **질문 분석** : GPT-4o를 통한 자연어 이해     - **도구 선택** : LangGraph Router의 최적 도구 매칭     - **병렬 분석** : 다중 AI 도구 동시 실행   + **기본 분석 노드**     - **재무제표 분석** : FinancialStatementTool 실행     - **기술적 분석** : 차트 패턴 및 지표 분석     - **시장 데이터** : 실시간 주가 및 거래량 분석     - **뉴스 분석** : 감정 분석 및 영향도 평가   + **고급 AI 노드**     - **시장 상태 분석** : Bull/Bear/Sideways 판별     - **최적 시점 예측** : 칼만 필터를 통한 진입점 예측     - **머신러닝 신호** : XGBoost + LSTM 앙상블     - **포트폴리오 최적화** : Black-Litterman 모델 적용   + **시스템 통합 노드**     - **통합 트레이딩** : 모든 분석 결과 종합     - **리스크 관리** : 스트레스 테스트 및 VaR 계산     - **성과 모니터링** : 실시간 성과 추적   + **검증 및 출력 노드**     - **데이터 검증** : 수집된 데이터 품질 확인     - **AI 분석 검증** : AI 분석 결과 신뢰도 검증     - **히스토리 저장** : Redis에 대화 기록 저장     - **실시간 응답** : WebSocket 토큰 스트리밍   + **오류 처리 노드**     - **오류 처리** : 예외 상황 대응     - **로그 기록** : 오류 상황 기록 및 분석     - **재시도 검증** : 재시도 가능 여부 판단 |

**1. 컴포넌트 다이어그램**

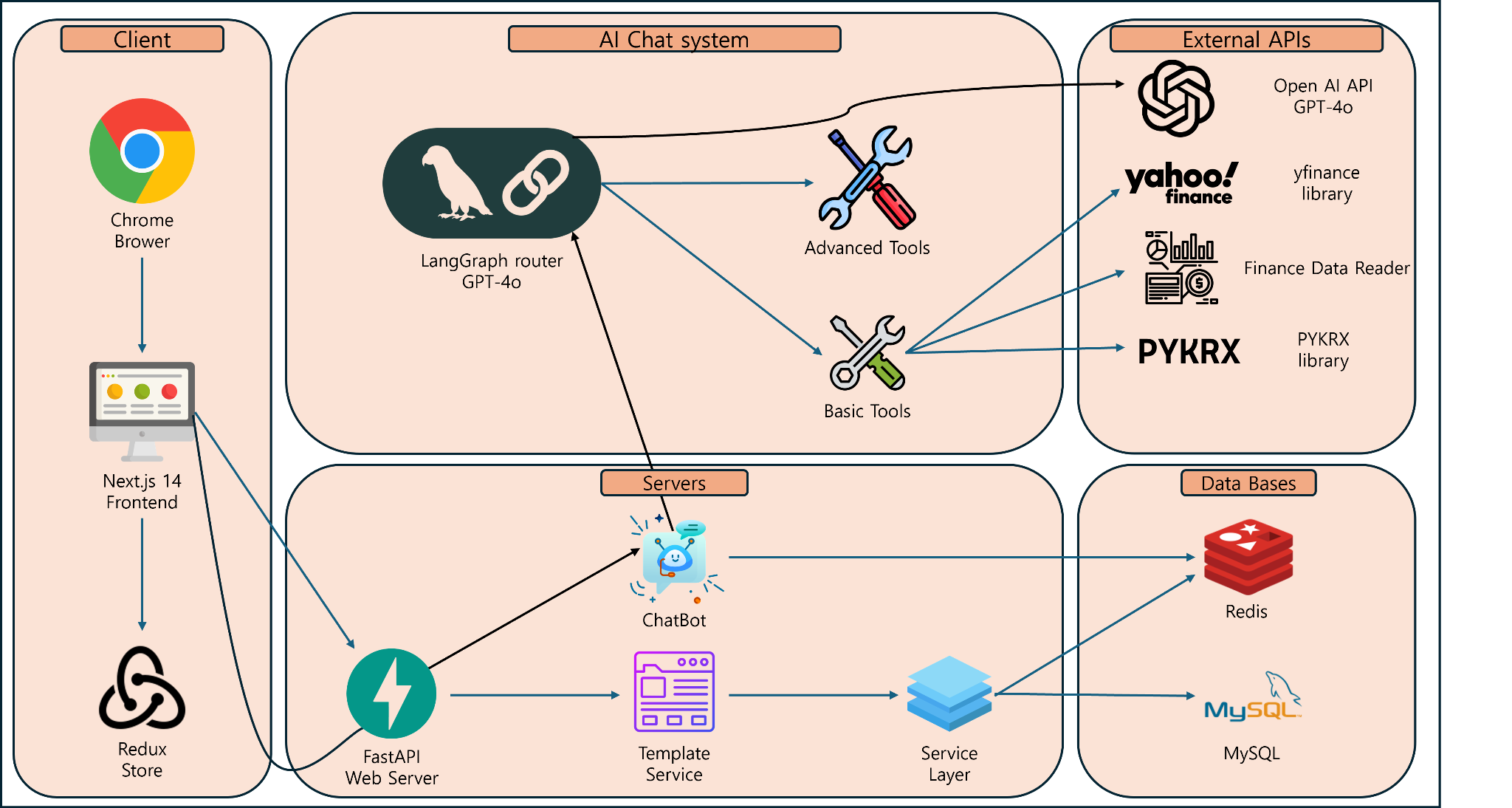
****

Figure 1. 컴포넌트 다이어그램

**구성 요소**

1. **클라이언트(Client):**

**- 컴포넌트:**

- Chrome 브라우저

- [Next.js](http://next.js) 15.2.4 (App Router)

- Redux Store (상태 관리)

**- 인터페이스:**

- HTTP 통신 (REST API)

- JSON 데이터 전송 (HTTPS)

1. **Servers:**

**- FastAPI 웹서버:**

- uvicorn (ASGI 서버)

- 11개 라우터 (account, admin, autotrade, chat, crawler, dashboard, market,

notification, portfolio, settings, tutorial)

- Template Service(비즈니스 로직 처리)

**- ChatBot 서버:**

- [app.py](http://app.py) (별도 FastAPI 인스턴스)

- LangGraph 통합

- Redis 채팅 히스토리 관리

**- Service Layer :**

- DatabaseService, CacheService, ExternalService 등 10개 서비스

1. **AI 채팅 시스템(AI Chat System):**

**- LangGraph Router:**

- GPT-4o 모델 통합

- AI 도구 조율 및 실행

**- BasicTools:**

- FinancialStatementTool, MarketDataTool, TechnicalAnalysisTool 등

**- Advanced Tools:**

- BlackLittermanOptimizerTool, MLSignalEnsembleTool,  
 MarketRegimeDetectorTool 등

1. **데이터베이스:**

**- MySQL:**

- 사용자 데이터, 포트폴리오, 거래 내역 저장

**- Redis:**

- 채팅 히스토리

- 세션 관리 및 캐시

1. **외부 API(External APIs):**

- yfinance(library):

- OpenAI API (GPT-4o)

- PyKrx (한국거래소 데이터)

- FinanceDataReader (금융 데이터)

**설명**

- 클라이언트는 Chrome 브라우저를 통해 [Next.js](http://next.js) 15.2.4 프론트엔드에 접속하고, Redux Store를 통해 상태를 관리하여, HTTP 통신으로 로컬 서버와 데이터를 주고받음

- 로컬 서버는 FastAPI 웹서버와 ChatBot 서버로 분리되어 있으며, FastAPI는 11개 라우터를 통해 비즈니스 로직을 처리하고, ChatBot 서버는 AI 채팅 기능을 담당함

- AI 채팅 시스템은 LangGraph Router가 사용자 질문을 분석하여 적절한 AI 도구들(BasicTools, Advanced Tools)을 선택하고 실행하여 투자 분석 결과를 제공

- 데이터베이스는 MySQL에서 구조화된 데이터를 저장하고, Redis에서 채팅 히스토리와 캐시 데이터를 관리

- 외부 API는 실시간 금융 데이터 수집을 위해 Yahoo Finance, PyKrx, FinanceDataReader를 사용하고, AI 분석을 위해 OpenAI GPT-4o API를 연동

**2. 시퀀스 다이어그램**

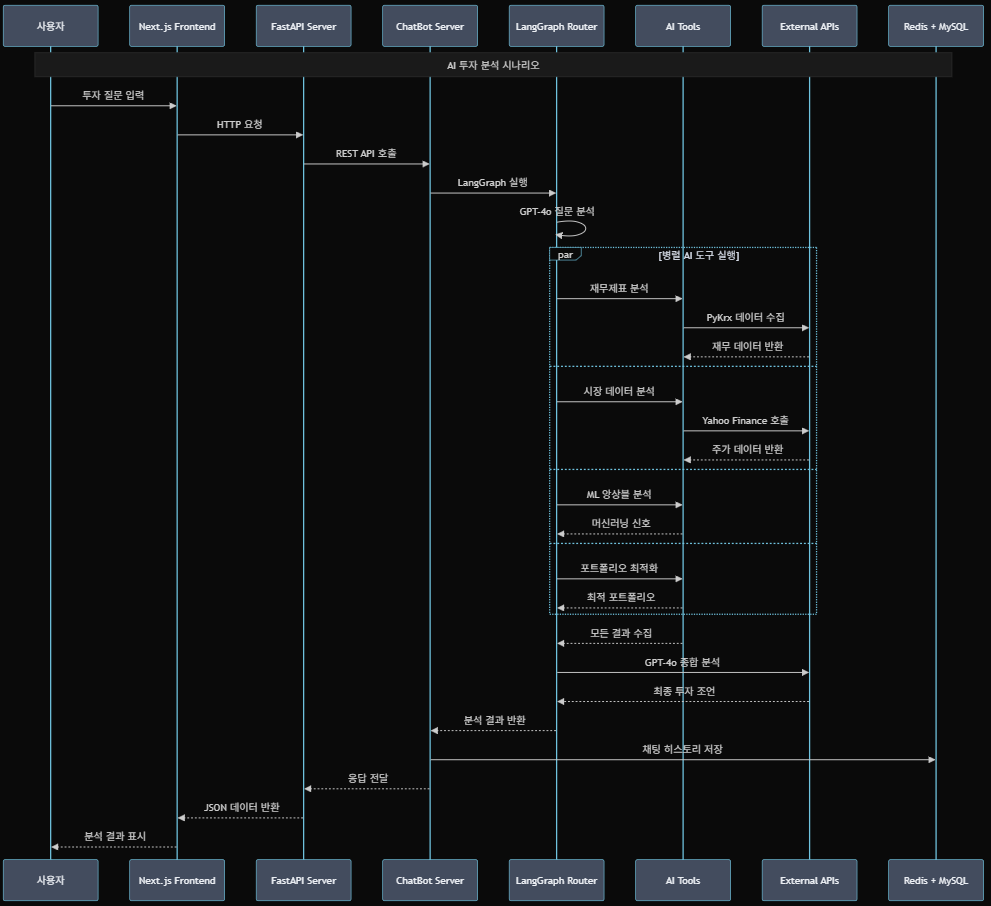
****

Figure 2. 시퀀스 다이어그램

**참여자 (Actors)**

1. **User**: Chrome 브라우저를 통해 [Next.js](http://next.js) 프론트엔드에서 AI 투자 분석을 요청하는 사용자
2. [**Next.js**](http://next.js) **Frontend** : 대략 20개의 페이지가 구현된 프론트엔드, useState로 로컬 상태 관리하여 HTTP 통신으로 백엔드와 연동
3. **FastAPI Web Server** : 메인 백엔드이며 라우터와 Template Service를 구현
4. **ChatBot Server** : [app.py](http://app.py)로 구현된 별도 FastAPI 인스턴스, REST API /chat 엔드포인트 제공
5. **LangGraph Router** : AIChat.Router.run\_question() 함수로 구현된 GPT-4o 기반 AI 도구 조율 시스템
6. **AI Tools** : 실제 구현된 Basic Tools와 Advanced Tools 임. 각각 독립적인 분석 기능 제공
7. **External APIs** : 실제 사용 중인 외부 서비스들 - yfinance(library), OpenAI GPT-4o, PyKrx, FinanceData
8. **Redis + MySQL** : localhost에서 실행되는 데이터베이스, Redis는 RedisChatMessageHistory로 채팅 저장, MySQL은 구조화된 데이터 저장.

**주요 흐름**

1. 사용자가 [Next.js](http://next.js) 프론트엔드에서 투자 관련 질문을 입력
2. 프론트엔드는 useState로 상태를 관리하며 FastAPI 웹서버로 HTTP POST 요청
3. FastAPI 웹서버는 Template Service의 ChatTemplateImpl을 통해 요청을 처리하고 ChatBot 서버의 REST API 호출
4. ChatBot 서버는 LangGraph Router를 실행하여 GPT-4o로 질문을 분석하고 적절한 AI 도구들을 선택
5. 선택된 AI 도구들이 병렬로 실행되어 각각 외부 API에서 데이터를 수집하거나 로컬에서 계산을 수행
6. 모든 도구 결과가 수집되면 LangGraph Router가 OpenAI GPT-4o API를 호출하여 종합적인 투자 조언을 생성
7. 생성된 결과와 대화 내역을 Redis의 RedisChatMessageHistory에 세션별로 저장
8. 최종 분석 결과가 HTTP 응답을 통해 프론트엔드로 전달되고, 사용자에게 투자 분석 결과가 표시됨
9. 모든 통신은 HTTP REST API로 이루어짐

**3. 액티비티 다이어그램**

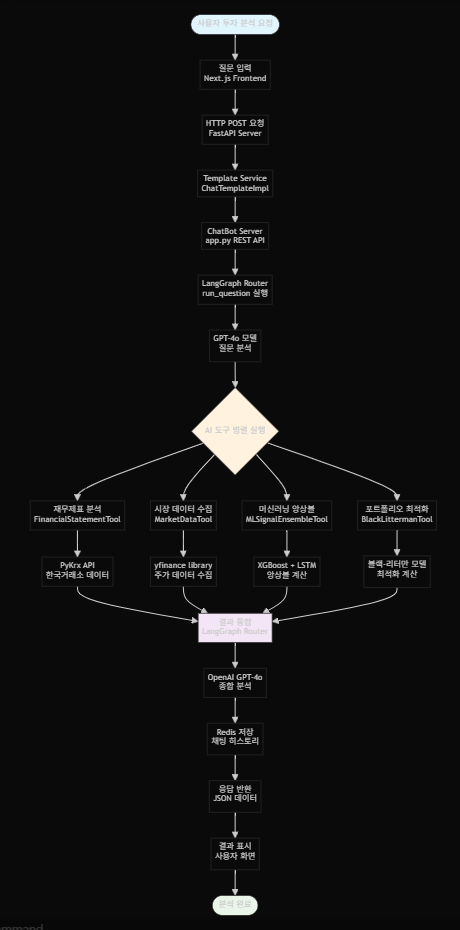
****

Figure 3. 액티비티 다이어그램

**구성**

**1. 시작 지점:** 사용자가 [Next.js](http://next.js) 프론트엔드에서 투자 분석을 요청

**2. 클라이언트 활동:**

- Chrome 브라우저를 통해 투자 질문을 입력

- useState를 사용하여 프론트엔드에서 로컬 상태를 관리

**3. 데이터 전송:**

- HTTP POST 요청으로 FastAPI 웹서버에 데이터를 전송

- JSON 형식의 데이터가 안전하게 전달됨

**4. 백엔드 처리:**

- Template Service의 ChatTemplateImpl이 요청을 처리

- ChatBot Server([app.py](http://app.py))의 REST API를 호출하여 AI 분석을 처리

**5. LangGraph 모델 호출:**

- LangGraph Router가 AIChat.Router.run\_question() 함수를 실행

- GPT-4o 모델이 질문을 분석하고 적절한 AI 도구들을 선택

**6. AI 도구 병렬 실행:**

- 16개 AI 도구가 동시에 실행되어 각각의 전문 분석을 수행

- 외부 API (PyKrx, yfinance)에서 실시간 데이터를 수집

- 로컬 계산 (머신러닝 앙상블, 포트폴리오 최적화)을 병렬로 처리

**7. 데이터베이스 작업:**

- Redis를 통해 RedisChatMessageHistory로 채팅 히스토리를 영구 저장

**8. 외부 서비스 호출:**

- OpenAI GPT-4o API를 통해 수집된 모든 데이터를 종합하여 최종 투자 조언을

생성

**9. 결과 반환:**

- 처리된 결과가 클라이언트로 반환되어 사용자 화면에 표시됨

**10. 종료 지점:** 작업 완료

**주요 액션 노드**

**- 클라이언트 활동:**

**- 사용자 투자 질문 입력** : [Next.js](http://next.js) Frontend에서 사용자 인터페이스 제공

**- useState 로컬 상태 관리** : 프론트엔드에서 실시간 상태 관리

**- 데이터 전송:**

**- HTTP POST 요청** : FastAPI Web Server로 안전한 데이터 전송

**- Template Service 처리** : ChatTemplateImpl을 통한 비즈니스 로직 처리

**- AI 분석 엔진:**

**- LangGraph Router 실행** : AIChat.Router.run\_question() 함수 호출

**- GPT-4o 질문 분석** : 사용자 질문 분석 및 적절한 도구 선택

**- 병렬 AI 도구 실행:**

**- FinancialStatementTool** : 재무제표 분석 및 PyKrx API 연동

**- MarketDataTool** : yfinance library를 통한 시장 데이터 수집

**- MLSignalEnsembleTool** : XGBoost + LSTM 앙상블 머신러닝 분석

**- BlackLittermanOptimizerTool** : 블랙-리터만 모델 포트폴리오 최적화

**- 데이터 통합 및 저장:**

**- 모든 도구 결과 통합** : LangGraph Router에서 결과 취합

**- OpenAI GPT-4o 종합 분석** : 최종 투자 조언 생성

**- Redis 채팅 히스토리 저장** : RedisChatMessageHistory를 통한 대화 영구 저장

**- 결과 표시:**

**- JSON 응답 반환** : FastAPI Web Server를 통한 구조화된 데이터 전송

**- 로컬 상태 업데이트** : setChatMessages를 통한 프론트엔드 상태 갱신

**- 투자 분석 결과 표시** : 사용자 화면에 최종 결과 렌더링