Vanilla Trading Basic

- 서비스 개요 및 기능 명세서 -

1. 서비스 개요

1.1 서비스 명

- Vanilla Trading Basic – 키움증권 REST API 기반 조건식 자동매매 시스템

1.2 목적

- 키움증권 HTS의 조건검색 기능과 REST API를 연동하여, 사용자가 설정한 조건식에 편입된 종목을 자동으로 매수하고, 목표 수익률(TP) 또는 손절률(SL)에 도달 시 자동으로 매도하는 완전 자동화 매매 시스템

1.3 핵심 가치

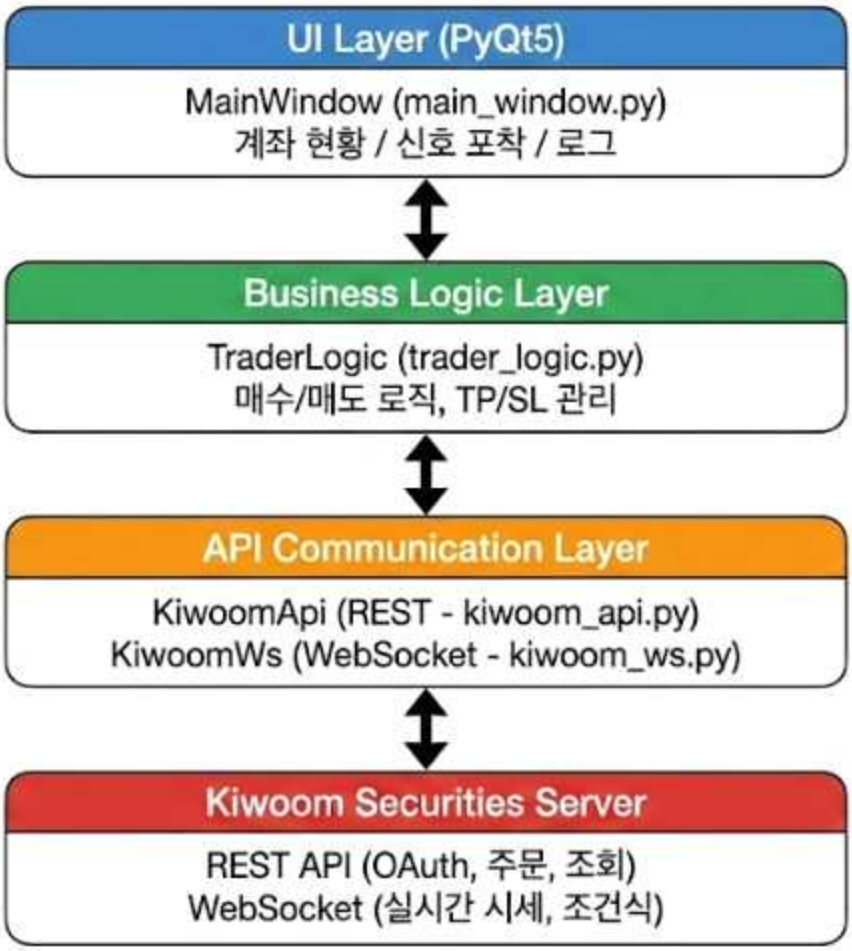
- 완전 자동화: 조건식 신호 포착 → 자동 매수 → TP/SL 자동 매도
- 실시간 모니터링: WebSocket 기반 실시간 시세 및 조건식 신호
- 리스크 관리: 최대 보유 종목 수, 1주 가격 상한, 당일 재진입 차단
- 사용자 친화적: 직관적인 GUI, 설정 저장/복원, 다중 전략 관리

1.4 기술 스택

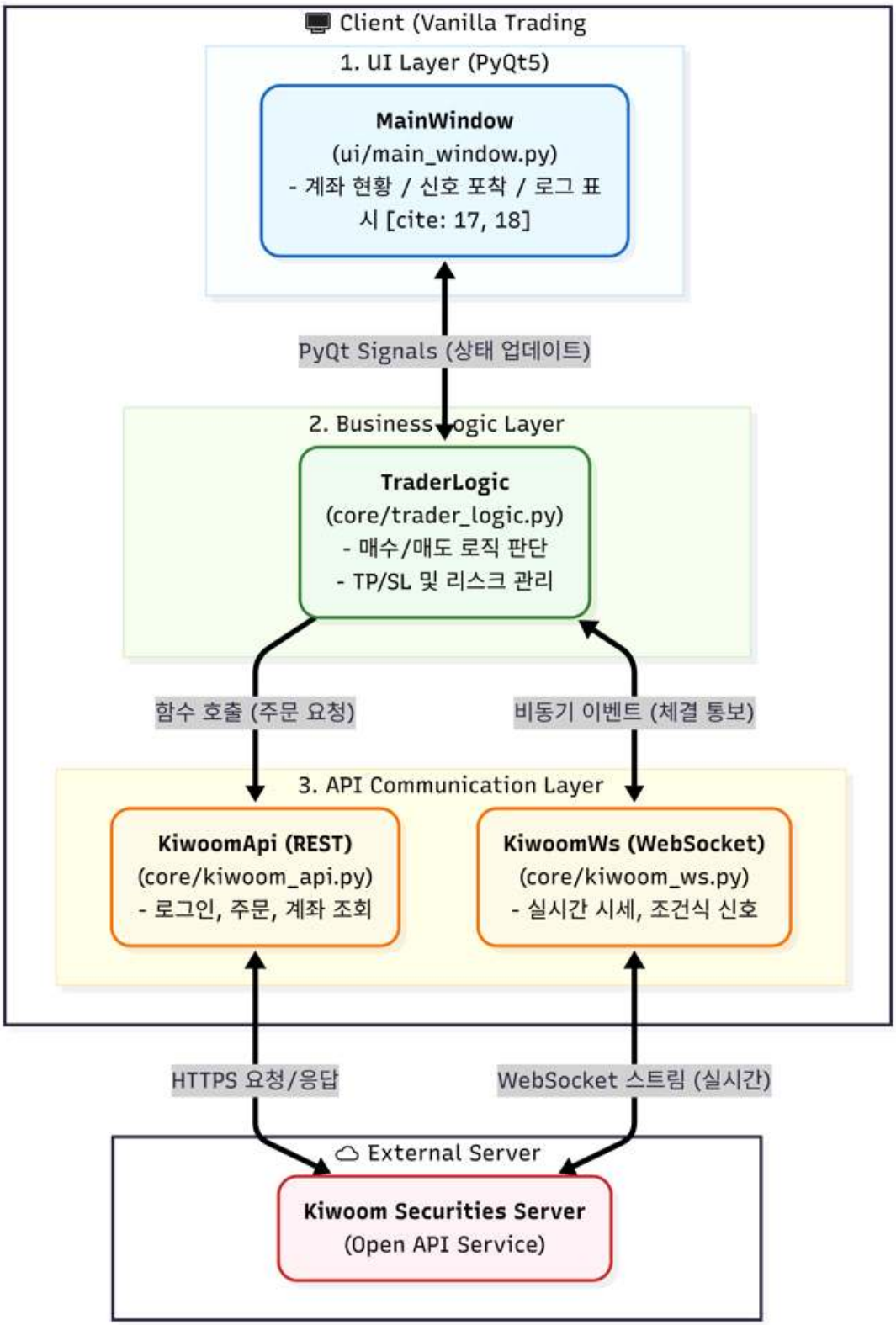
구분	기술
언어	Python 3.8+
GUI	PyQt5
API 통신	requests (REST), websockets (실시간)
비동기 처리	asyncio
설정 관리	configparser (INI 파일)
주문 방식	시장가 매수/매도 (고정)

2. 시스템 아키텍처

2.1 계층 구조



2.2 아키텍처


Client (Vanilla Trading
1. UI Layer (PyQt5)
MainWindow
(ui/main_window.py)
- 계좌 현황 / 신호 포착 / 로그 표시 [cite: 17, 18]
PyQt Signals (상태 업데이트)
2. Business Logic Layer
TraderLogic
(core/trader_logic.py)
- 매수/매도 로직 판단
- TP/SL 및 리스크 관리
함수 호출 (주문 요청)
비동기 이벤트 (체결 통보)
3. API Communication Layer
KiwoomApi (REST)
(core/kiwoom_api.py)
- 로그인, 주문, 계좌 조회
KiwoomWs (WebSocket)
(core/kiwoom_ws.py)
- 실시간 시세, 조건식 신호
HTTPS 요청/응답
WebSocket 스트림 (실시간)
External Server
Kiwoom Securities Server
(Open API Service)

2.3 파일 구조

```
VanillaTrader/
├── app.py                    # 메인 진입점 (환경 검증, QApplication)
├── config.ini                # 설정 파일 (조건식, TP/SL, 시간 등)
│
├── core/                     # 비즈니스 로직 및 API 레이어
│   ├── __init__.py
│   ├── kiwoom_api.py         # REST API 클라이언트
│   ├── kiwoom_ws.py          # WebSocket 클라이언트
│   └── trader_logic.py       # 자동매매 핵심 로직
│
└── ui/                       # UI 레이어
    ├── __init__.py
    └── main_window.py        # PyQt5 메인 윈도우
```

3. 핵심 기능 명세

3.1 인증 및 연결

기능 ID	기능 명	설명	구현 위치
AUTH-001	OAuth 토큰 발급	APP_KEY/SECRET으로 Bearer 토큰 발급	KiwoomApi.login()
AUTH-002	토큰 자동 갱신	1시간마다 토큰 유효성 확인 및 재발급	KiwoomApi.ensure_token()
CONN-001	WebSocket 연결	실시간 시세/조건식용 WS 연결	KiwoomWs.run()
CONN-002	자동 재연결	연결 끊김 시 지수 백오프 재연결	KiwoomWs._handle_reconnect()

3.2 계좌 및 잔고 조회

기능 ID	기능 명	설명	구현 위치
ACC-001	예수금 조회	매수 가능 금액 조회 (ka01690)	KiwoomApi.get_current_balance()
ACC-002	계좌 잔고 UI 표시	예수금을 UI에 실시간 표시	MainWindow.update_account_status()
ACC-003	종목명 캐싱	계좌 잔고 조회 시 종목명 자동 캐싱	TraderLogic._stock_names

3.3 조건식 관리

기능 ID	기능 명	설명	구현 위치
COND-001	조건식 목록 조회	HTS 조건식 목록 가져오기 (REST 우선)	KiwoomApi.get_condition_list()
COND-002	조건식 선택	콤보박스에서 조건식 선택	MainWindow.on_condition_changed()
COND-003	조건식 실시간 구독	선택한 조건식의 실시간 신호 수신 (CNSRREQ)	KiwoomWs.subscribe_condition()
COND-004	조건식 신호 수신	ADD/DEL 이벤트 수신 (CNSR)	TraderLogic.on_realtime_signal()
COND-005	조건식 설정 저장	선택한 조건식 번호를 config.ini에 저장	MainWindow._save_global_settings()

3.4 시세 조회 및 실시간 모니터링

기능 ID	기능 명	설명	구현 위치
PRICE-001	현재가 조회	REST로 현재가/등락률/거래량 조회 (ka10006)	KiwoomApi.get_stock_price()
PRICE-002	호가 조회	매수/매도 1호가 조회 (ka10004)	KiwoomApi.get_stock_price()
PRICE-003	실시간 시세 구독	WebSocket으로 실시간 시세 구독 (REG, type=0A)	KiwoomWs.subscribe_price()
PRICE-004	실시간 시세 파싱	REAL 메시지에서 현재가/등락률/거래량 추출	KiwoomWs.parse_realtime_price()
PRICE-005	시세 스냅샷 폴백	WebSocket 장애 시 REST로 폴백 조회	TraderLogic._fetch_price_snapshot()

3.5 자동 매수 로직

기능 ID	기능 명	설명	구현 위치
BUY-001	조건식 신호 포착	조건식에 편입된 종목 감지	TraderLogic._handle_condition_signal()
BUY-002	매수 필터링	1주 가격이 BUY_AMOUNT 이하인지 확인	TraderLogic._auto_buy()
BUY-003	최대 종목 수 제한	MAX_STOCKS 도달 시 매수 스킵	TraderLogic._auto_buy()
BUY-004	시장가 매수 주문	1주 고정 시장가 매수 (kt10000)	KiwoomApi.buy_market_order()
BUY-005	포지션 등록	매수 성공 시 open_positions에 추가	TraderLogic._auto_buy()
BUY-006	예수금 차감	매수 금액만큼 예수금 차감	TraderLogic._update_cash()
BUY-007	당일 재진입 차단	같은 날 매도한 종목 재매수 금지	TraderLogic._can_reenter_today()
BUY-008	매수 거부 리스트	특정 종목 매수 스킵 (사용자 토글)	TraderLogic.rejected_codes

3.6 자동 매도 로직

기능 ID	기능 명	설명	구현 위치
SELL-001	TP/SL 실시간 체크	WebSocket REAL 틱으로 수익률 계산	TraderLogic.on_realtime_signal()
SELL-002	익절 (Take Profit)	수익률 ≥ TP% 시 전량 매도	TraderLogic.on_realtime_signal()
SELL-003	손절 (Stop Loss)	수익률 ≤ SL% 시 전량 매도	TraderLogic.on_realtime_signal()
SELL-004	시장가 매도 주문	전량 시장가 매도 (kt10001)	KiwoomApi.sell_market_order()

SELL-005	**포지션 해제**	매도 성공 시 open_positions에서 제거	TraderLogic._auto_sell()
SELL-006	**예수금 증가**	매도 금액만큼 예수금 증가	TraderLogic._update_cash()
SELL-007	**REST 폴백 체크**	WebSocket 장애 시 REST로 TP/SL 체크	TraderLogic._check_positions()

3.7 UI 기능

기능 ID	기능 명	설명	구현 위치
UI-001	**자동매매 ON/OFF**	버튼 클릭으로 자동매매 시작/중지	MainWindow.on_start/stop_trading_clicked()
UI-002	**매수 조건 설정**	조건식, 매수 금액, 최대 종목 수 설정	MainWindow (좌측 그룹)
UI-003	**매도 조건 설정**	TP/SL 비율 설정, 매도 전략 저장/로드	MainWindow (우측 그룹)
UI-004	**자동매매 시간 설정**	시작/종료 시간 설정 (기본: 09:00~15:30)	MainWindow.time_start/end
UI-005	**계좌 현황 탭**	매수 가능 금액 표시	MainWindow.tab_account
UI-006	**신호 포착 탭**	조건식 편입 종목 실시간 표시	MainWindow.tab_signal
UI-007	**자동매매 현황 탭**	매수/매도 로그 실시간 표시	MainWindow.tab_log
UI-008	**매수 거부 버튼**	신호 포착 탭에서 종목별 매수 거부 토글	MainWindow.on_reject_signal_clicked()
UI-009	**실시간 시세 업데이트**	신호 포착 테이블 실시간 갱신	MainWindow.update_signal_row_realtime()
UI-010	**다크 테마 UI**	전체 다크 모드 스타일시트 적용	MainWindow.setStyleSheet()

3.8 설정 관리

기능 ID	기능 명	설명	구현 위치
CFG-001	글로벌 설정 저장	조건식 번호, 매수 금액, 최대 종목 수, 시간	MainWindow._save_global_settings()
CFG-002	매도 전략 저장	TP/SL 비율을 전략 이름으로 저장	MainWindow.save_sell_strategy()
CFG-003	매도 전략 로드	저장된 전략 선택 시 TP/SL 자동 로드	MainWindow.on_sell_condition_changed()
CFG-004	다중 전략 관리	여러 개의 매도 전략 프리셋 관리	config.ini [SELL_STRATEGY:*]
CFG-005	설정 복원	프로그램 재시작 시 마지막 설정 자동 로드	MainWindow._load_global_settings()

3.9 리스크 관리

기능 ID	기능 명	설명	구현 위치
RISK-001	1주 가격 상한	BUY_AMOUNT 초과 종목 매수 차단	TraderLogic._auto_buy()
RISK-002	최대 보유 종목 수	MAX_STOCKS 도달 시 신규 매수 차단	TraderLogic._auto_buy()
RISK-003	당일 재진입 차단	같은 날 매도한 종목 재매수 금지	TraderLogic.reentry_block
RISK-004	자동매매 시간 제한	설정한 시간대에만 매수/매도 실행	TraderLogic.start_time/end_time
RISK-005	매수 거부 리스트	사용자가 특정 종목 매수 차단	TraderLogic.rejected_codes

3.10 로깅 및 모니터링

기능 ID	기능 명	설명	구현 위치
LOG-001	시스템 로그	초기화, 자동매매 시작/중지 등	TraderLogic._emit_log()
LOG-002	매수 주문 로그	매수 요청/성공/실패 기록	TraderLogic._auto_buy()
LOG-003	매도 주문 로그	매도 요청/성공/실패 기록	TraderLogic._auto_sell()
LOG-004	UI 로그 테이블	최대 1000개 행 유지 (자동 삭제)	MainWindow._add_log_entry()
LOG-005	콘솔 디버그 로그	상세한 디버깅 정보 출력	print() 문
LOG-006	WebSocket 상태 모니터링	HEARTBEAT 주기적 출력	KiwoomWs._heartbeat_loop()

3.11 예외 처리 및 안정성

기능 ID	기능 명	설명	구현 위치
ERR-001	전역 예외 처리	예상치 못한 오류 팝업 및 로그	app.excepthook()
ERR-002	API 오류 처리	return_code != 0 시 로그 기록	각 API 메서드
ERR-003	WebSocket 재연결	연결 끊김 시 지수 백오프 재연결	KiwoomWs._handle_reconnect()
ERR-004	JSON 파싱 오류 방어	잘못된 응답 시 안전하게 처리	try-except 블록
ERR-005	환경 검증	Python 버전, 의존성 패키지 확인	app.validate_*()

4. 데이터 흐름

4.1 초기화 플로우

1. app.py 시작
    └──> 환경 검증 (Python 버전, 패키지, 파일 구조)
    └──> config.ini 검증 (CONDITION_SEQ 확인)
    └──> QApplication 생성
    └──> MainWindow 생성
        └──> TraderLogic 생성
            └──> KiwoomApi 생성 (APP_KEY, APP_SECRET)
            └──> WebSocket 스레드 준비

2. TraderLogic.initialize_background()
    └──> KiwoomApi.login() (OAuth 토큰 발급)
    └──> WebSocket 스레드 시작
    └──> KiwoomWs.run() (연결 및 LOGIN 메시지)
    └──> 조건식 목록 조회 (REST 우선, WS 폴백)
    └──> 계좌 잔고 조회 (종목명 캐싱)
    └──> 기본 조건식 실시간 구독

3. UI 표시
    └──> 조건식 콤보박스 채우기
    └──> 매수 가능 금액 표시
    └──> 자동매매 OFF 상태

4.2 자동매매 ON 플로우

1. 사용자: [자동매매 ON] 버튼 클릭

2. MainWindow.on_start_trading_clicked()
   └──> UI 설정을 TraderLogic에 전달
      - buy_amount, max_stock_limit
      - start_time, end_time
      - stop_loss_rate, profit_cut_rate
      - condition_seq

3. TraderLogic.start_auto_trading()
   └──> is_trading = True
   └──> position_timer 시작 (5초 간격)
   └──> 현재 조건 편입 종목 즉시 매수 시도

4. 대기 상태
   └──> WebSocket 조건식 신호 대기
   └──> WebSocket 실시간 시세 대기

4.3 매수 플로우

1. WebSocket 조건식 신호 수신 (CNSR, type=ADD)
   └──> TraderLogic.on_realtime_signal()

2. TraderLogic._handle_condition_signal(종목코드)
   └──> REST로 현재가 스냅샷 조회
   └──> 신호 포착 테이블에 행 추가
   └──> WebSocket 실시간 시세 구독
   └──> 자동매매 ON 확인
   └──> 필터링 검사:

- 매수 거부 리스트?
- 당일 재진입 차단?
- 최대 종목 수 도달?
- 이미 보유 중?

└──> TraderLogic._auto_buy()

3. TraderLogic._auto_buy(종목코드, 스냅샷)
   └──> 1주 가격 확인 (BUY_AMOUNT 이하?)
   └──> 예수금 확인
   └──> KiwoomApi.buy_market_order(종목코드, 1주)
   └──> 성공 시:
      - open_positions에 추가
      - 예수금 차감
      - UI 업데이트

4. WebSocket 실시간 시세 수신 (REAL, type=0A)
   └──> 신호 포착 테이블 실시간 갱신

4.4 매도 플로우

1. WebSocket 실시간 시세 수신 (REAL, type=0A)
   └──> TraderLogic.on_realtime_signal()

2. REAL 틱 파싱
   └──> KiwoomWs.parse_realtime_price()
   └──> 종목코드, 현재가, 등락률, 거래량 추출

3. 보유 포지션 확인
   └──> 종목코드가 open_positions에 있는가?
   └──> 수익률 계산: (현재가 - 진입가) / 진입가 * 100

4. TP/SL 조건 확인
   └──> 수익률 ≥ profit_cut_rate? → 익절
   └──> 수익률 ≤ stop_loss_rate? → 손절

5. TraderLogic._auto_sell(종목코드, 수량, 현재가)
   └──> KiwoomApi.sell_market_order(종목코드, 수량)
   └──> 성공 시:
      - open_positions에서 제거
      - 예수금 증가
      - 당일 재진입 차단 등록
      - UI 업데이트

4.5 WebSocket 장애 시 폴백

1. WebSocket 연결 끊김 감지
   └──> KiwoomWs.connected = False

2. REST 폴백 타이머 활성화
   └──> TraderLogic._check_positions() (5초 간격)
   └──> TraderLogic._refresh_signals() (5초 간격)

3. REST API로 시세 조회
   └──> KiwoomApi.get_stock_price()
   └──> TP/SL 체크
   └──> 신호 포착 테이블 갱신

4. WebSocket 재연결 성공 시
   └──> REST 타이머는 유지 (이중 안전장치)
   └──> REAL 틱 우선 사용

5. 설정 파일 구조 (config.ini)

5.1 기본 구조

```ini
[KIWOOM_API]
APP_KEY =your_app_key_here
APP_SECRET =your_app_secret_here

[GLOBAL_SETTINGS]
CONDITION_SEQ =1                    # 조건식 번호 (0이면 작동 안함!)
BUY_AMOUNT =200000                  # 1주 최대 가격 (이하만 매수)
MAX_STOCKS =5                       # 최대 보유 종목 수
START_TIME =09:00                   # 자동매매 시작 시간
END_TIME =15:30                     # 자동매매 종료 시간

[SELL_STRATEGY:기본 매도 전략]
STOP_LOSS_RATE =-1.50               # 손절률 (%)
PROFIT_CUT_RATE =1.50               # 익절률 (%)

[SELL_STRATEGY:공격적 전략]
STOP_LOSS_RATE =-2.00
PROFIT_CUT_RATE =3.00

[SELL_STRATEGY:보수적 전략]
STOP_LOSS_RATE =-1.00
PROFIT_CUT_RATE =1.00
```

5.2 설정 항목 설명

섹션	키	설명	기본값	비고
KIWOOM_API	APP_KEY	키움 REST API 앱키	–	필수
	APP_SECRET	키움 REST API 시크릿키	–	필수
GLOBAL_SETTINGS	CONDITION_SEQ	조건식 번호	0	0이면 경고 표시
	BUY_AMOUNT	1주 최대 가격	200000	원 단위
	MAX_STOCKS	최대 보유 종목 수	5	1~50
	START_TIME	자동매매 시작 시간	09:00	HH:mm
	END_TIME	자동매매 종료 시간	15:30	HH:mm
SELL_STRATEGY:*	STOP_LOSS_RATE	손절률	-1.50	% (음수)
	PROFIT_CUT_RATE	익절률	1.50	% (양수)

6. 교육용 구현 순서 (권장)

Phase 1: 기반 구축 (1-2일)

1.1. ✅ 프로젝트 구조 생성 (app.py, core/, ui/)
1.2. ✅ config.ini 파일 생성 및 파싱 로직
1.3. ✅ 환경 검증 로직 (validate_* 함수들)
1.4. ✅ PyQt5 기본 윈도우 띄우기

Phase 2: REST API 통신 (2-3일)

1.1. ✅ KiwoomApi 클래스 생성
1.2. ✅ OAuth 토큰 발급 (login())
1.3. ✅ 토큰 자동 갱신 (ensure_token())
1.4. ✅ 계좌 잔고 조회 (get_current_balance())
1.5. ✅ 현재가 조회 (get_stock_price())
1.6. ✅ 조건식 목록 조회 (get_condition_list())
1.7. ✅ 매수 주문 (buy_market_order())
1.8. ✅ 매도 주문 (sell_market_order())

Phase 3: WebSocket 통신 (3-4일)

1.1. ✅ KiwoomWs 클래스 생성
1.2. ✅ WebSocket 연결 및 LOGIN
1.3. ✅ PING/PONG 처리
1.4. ✅ 조건식 실시간 구독 (CNSRREQ)
1.5. ✅ 조건식 신호 수신 (CNSR)
1.6. ✅ 실시간 시세 구독 (REG, type=0A)

1.7. ✅ 실시간 시세 파싱 (REAL)
1.8. ✅ 자동 재연결 로직

Phase 4: 자동매매 로직 (4-5일)

1.1. ✅ TraderLogic 클래스 생성
1.2. ✅ 초기화 로직 (initialize_background())
1.3. ✅ 조건식 신호 처리 (_handle_condition_signal())
1.4. ✅ 자동 매수 로직 (_auto_buy())
1.5. ✅ TP/SL 체크 로직 (REAL 기반)
1.6. ✅ 자동 매도 로직 (_auto_sell())
1.7. ✅ 리스크 관리 (1주 가격 상한, 최대 종목 수, 재진입 차단)
1.8. ✅ 매수 거부 리스트

Phase 5: UI 구현 (3-4일)

1.1. ✅ 상단 컨트롤 바 (자동매매 ON/OFF, 시간 설정)
1.2. ✅ 매수 조건 그룹 (조건식, 매수 금액, 최대 종목 수)
1.3. ✅ 매도 조건 그룹 (TP/SL, 전략 저장/로드)
1.4. ✅ 계좌 현황 탭
1.5. ✅ 신호 포착 탭 (실시간 시세 표시)
1.6. ✅ 자동매매 현황 탭 (로그)
1.7. ✅ 매수 거부 버튼
1.8. ✅ 다크 테마 스타일시트

Phase 6: 통합 테스트 및 최적화 (2-3일)

1.1. ✅ 시그널 연결 확인
1.2. ✅ 캐싱 최적화 (종목명, 신호 테이블 행 인덱스)
1.3. ✅ 로그 테이블 최대 행 수 제한
1.4. ✅ 예외 처리 강화
1.5. ✅ 모의투자 환경 테스트
1.6. ✅ 실전투자 환경 테스트

7. 주의 사항 및 제약사항

7.1 기술적 제약

- 시장가 주문만 지원 (지정가 미지원)
- 1주 고정 매수 (수량 조절 불가)
- 조건식 번호 0은 작동하지 않음 (반드시 유효한 번호 설정)
- WebSocket REAL 틱이 오지 않으면 TP/SL이 작동하지 않음 (REST 폴백은 5초 간격)

7.2 운영 제약

- 당일 재진입 차단: 같은 날 매도한 종목은 재매수 불가
- 최대 보유 종목 수: 설정한 수를 초과하면 신규 매수 차단
- 자동매매 시간: 설정한 시간 외에는 매수/매도 실행 안 함
- 매수 거부 리스트: 수동으로 토글하지 않으면 자동 해제 안 됨

7.3 API 제약

- 토큰 유효 시간: 1시간 (자동 갱신 필요)
- WebSocket 연결 안정성: 간헐적으로 끊길 수 있음 (재연결 로직 필수)
- 조건식 목록 조회: 간혹 실패할 수 있음 (WS 폴백 필요)

8. 향후 확장 가능성

기능	설명	우선순위
지정가 주문	시장가 외 지정가 매수/매도	High
수량 조절	1주 고정이 아닌 금액 기반 수량 계산	High
다중 조건식	여러 조건식 동시 구독	Medium
백테스팅	과거 데이터로 전략 검증	Medium
알림 기능	매수/매도 시 카카오톡/텔레그램 알림	Medium
매수 타이밍 조절	조건 편입 직후가 아닌 특정 패턴 후 매수	Low
분할 매도	일부만 매도하는 전략	Low
손익 분석	일일/주간/월간 손익 통계	Low

-

☑ 체크리스트

환경 설정

- ☐ Python 3.8+ 설치
- ☐ PyQt5, requests, websockets 패키지 설치
- ☐ 키움 REST API 앱키/시크릿키 발급
- ☐ config.ini 파일 생성 및 설정

구현 완료

- ☐ KiwoomApi 클래스 (REST)
- ☐ KiwoomWs 클래스 (WebSocket)
- ☐ TraderLogic 클래스 (자동매매 로직)
- ☐ MainWindow 클래스 (UI)
- ☐ app.py (메인 진입점)

테스트 완료

- ☐ OAuth 로그인 성공
- ☐ 조건식 목록 조회 성공
- ☐ 실시간 시세 수신 확인
- ☐ 조건식 신호 수신 확인
- ☐ 자동 매수 테스트 (모의투자)
- ☐ TP/SL 자동 매도 테스트 (모의투자)
- ☐ UI 동작 확인 (버튼, 탭, 테이블)

배포 준비

- ☐ 예외 처리 완료
- ☐ 로그 시스템 확인
- ☐ 설정 저장/복원 확인
- ☐ 사용 설명서 작성
- ☐ 실전투자 환경 최종 검증