MCP Financial Markets Analysis Tool

An AI-powered financial analysis platform that combines **Model Context Protocol (MCP)**, **smolagents**, **FastAPI**, and **Streamlit** to deliver professional-grade investment analysis reports. The system uses Large Language Models to orchestrate trading strategy tools and interpret financial data, transforming raw market data into actionable investment insights.

Overview

This application provides **5 distinct analysis types** through a modern web interface:

Analysis Type	Description	Use Case
Technical Analysis	4 trading strategies on single stock	Deep dive into price patterns
Market Scanner	Compare multiple stocks simultaneously	Find best opportunities
Fundamental Analysis	Financial statements interpretation	Assess company health
Multi-Sector Analysis	Cross-sector comparison	Portfolio diversification
Combined Analysis	Technical + Fundamental together	Complete investment thesis

What Makes This Different?

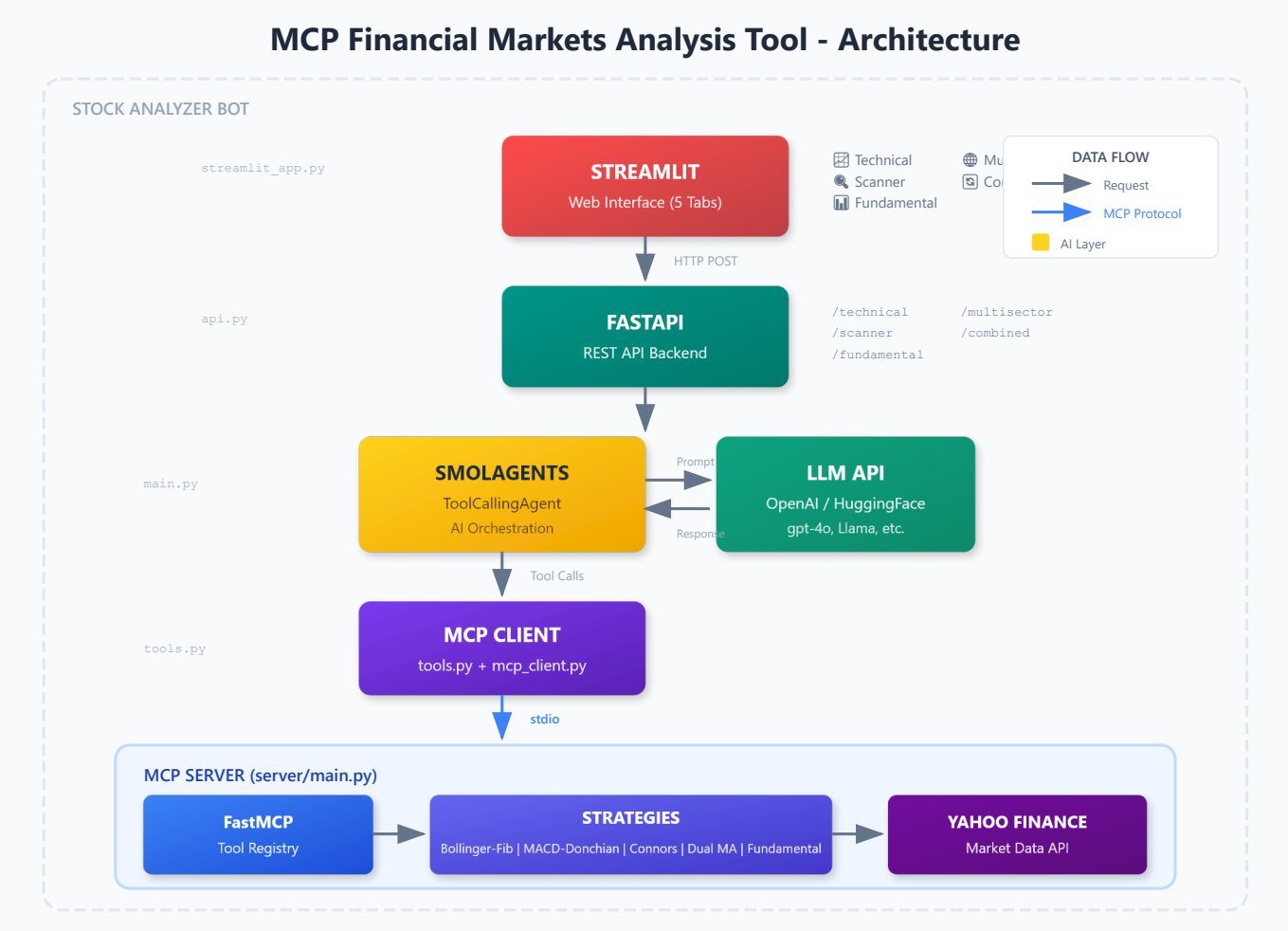
Unlike traditional analysis tools that just display numbers, this system uses **AI to interpret** the data:

```
Traditional Tool: "RSI = 28.5, MACD = -2.3, P/E = 15.2"

This Application: "AAPL shows oversold conditions with RSI at 28.5, suggesting
                   a potential mean reversion opportunity. Combined with strong
                   fundamentals (P/E of 15.2 below sector average), this presents
                   a BUY signal with high conviction..."
```

Architecture

System Components



Data Flow Summary:

1. **Streamlit** → User interacts with web interface (5 analysis tabs)
2. **FastAPI** → REST API receives requests, validates input
3. **smolagents** → AI agent decides which tools to call
4. **LLM API** → OpenAI/HuggingFace processes prompts, guides tool selection
5. **MCP Client** → Bridges smolagents to MCP server via stdio
6. **MCP Server** → Executes financial analysis tools
7. **Strategies** → Calculate technical indicators, run backtests
8. **Yahoo Finance** → Provides market data

Folder Structure

```
mcp_financial_markets_analysis_tool/
│
├── server/                          # MCP Server (Financial Tools)
│   ├── main.py                      # Server entry point
│   ├── strategies/                  # Trading strategy implementations
│   │   ├── bollinger_fibonacci.py   # Bollinger + Fibonacci
│   │   ├── macd_donchian.py         # MACD + Donchian Channel
│   │   ├── connors_zscore.py        # Connors RSI + Z-Score
│   │   ├── dual_moving_average.py   # 50/200 EMA Crossover
│   │   ├── bollinger_zscore.py      # Bollinger + Z-Score
│   │   ├── fundamental_analysis.py  # Financial Statements
```

```
│   │   ├── performance_tools.py     # Backtesting Tools
│   │   └── unified_market_scanner.py# Multi-Stock Scanner
│   ├── utils/
│   │   └── yahoo_finance_tools.py   # Data & Indicator Calculations
│   └── README.md                    # 📄 Detailed Server Documentation
│
├── stock_analyzer_bot/              # Smolagents Bot (AI Orchestration)
│   ├── __init__.py
│   ├── main.py                      # Analysis functions & LLM prompts
│   ├── api.py                       # FastAPI REST endpoints
│   ├── tools.py                     # Smolagents tool wrappers
│   ├── mcp_client.py                # MCP connection manager
│   └── README.md                    # 📄 Detailed Bot Documentation
│
├── streamlit_app.py                 # Web UI (5 Analysis Tabs)
├── .env                             # Environment variables
├── requirements.txt                 # Python dependencies
└── README.md                        # 📄 This file
```

🤖 Understanding Smolagents

What is Smolagents?

Smolagents is an open-source Python library from Hugging Face that makes it easy to build AI agents that can use tools. It's the "brain" of our application.

> *"smolagents is designed to make it extremely easy to build and run agents using just a few lines of code."*
> - HuggingFace

Key Smolagents Features We Use

Feature	How We Use It
ToolCallingAgent	Orchestrates calls to our 7 financial tools
LiteLLMModel	Connects to OpenAI (gpt-4o, gpt-4-turbo)
InferenceClientModel	Connects to HuggingFace models
@tool decorator	Wraps our MCP tools for agent use
Multi-step reasoning	Agent decides which tools to call and in what order

How Smolagents Works in Our App

```python
# 1. Define tools that the agent can use
from smolagents import tool

@tool
def bollinger_fibonacci_analysis(symbol: str, period: str = "1y") -> str:
    """Analyze stock using Bollinger Bands + Fibonacci retracement."""
```

```
    # Calls MCP server, returns analysis data
    return _call_mcp_tool("analyze_bollinger_fibonacci_performance", {...})

# 2. Create an agent with the LLM and tools
from smolagents import ToolCallingAgent, LiteLLMModel

model = LiteLLMModel(model_id="gpt-4o")
agent = ToolCallingAgent(
    tools=[bollinger_fibonacci_analysis, macd_donchian_analysis, ...],
    model=model,
    max_steps=25,
)

# 3. Run the agent with a prompt
report = agent.run("""
    Analyze AAPL stock using all 4 technical strategies.
    Create a comprehensive markdown report with recommendations.
""")
```

The Agent's Decision Process

When you ask for analysis, the agent:

```
1. READS the prompt: "Analyze AAPL with 4 strategies"

2. PLANS: "I need to call 4 tools: bollinger_fibonacci, macd_donchian,
          connors_zscore, dual_moving_average"

3. EXECUTES: Calls each tool, receives data

4. SYNTHESIZES: Combines all results, identifies patterns

5. GENERATES: Creates professional markdown report with recommendation
```

This is why results are **interpreted**, not just displayed.

🖩 Streamlit Interface - 5 Analysis Types

Tab 1: 📈 Technical Analysis

Purpose: Deep dive into a single stock using 4 trading strategies

What It Does:

- Calls 4 strategy tools for one stock
- Each strategy provides: signal, score, return %, Sharpe ratio, max drawdown
- AI synthesizes into cohesive report with recommendation

Strategies Used:

1. **Bollinger-Fibonacci** - Support/resistance with volatility bands
2. **MACD-Donchian** - Momentum with breakout detection
3. **Connors RSI + Z-Score** - Mean reversion signals
4. **Dual Moving Average** - Trend following (Golden/Death Cross)

Best For: "Should I buy/sell/hold this specific stock?"

Tab 2: 🔍 Market Scanner

Purpose: Compare multiple stocks and rank opportunities

What It Does:

- Runs all 4 strategies on each stock in your list
- Compares performance across stocks
- Ranks from best to worst opportunity
- Identifies consensus picks

Example Input: `AAPL, MSFT, GOOGL, META, NVDA, AMD`

Best For: "Which stock in this group is the best opportunity?"

Tab 3: 📊 Fundamental Analysis

Purpose: Analyze company financial health from statements

What It Does:

- Retrieves income statement, balance sheet, cash flow
- Calculates key ratios: P/E, ROE, debt-to-equity, margins
- AI interprets financial health
- Creates investment thesis

Metrics Analyzed:

- **Profitability:** Revenue, Net Income, Margins
- **Growth:** Revenue growth, earnings growth
- **Liquidity:** Current ratio, quick ratio
- **Leverage:** Debt ratios, interest coverage
- **Returns:** ROE, ROA

Best For: "Is this company financially healthy?"

Tab 4: 🌐 Multi-Sector Analysis

Purpose: Compare stocks across different market sectors

What It Does:

- Analyzes multiple sectors (Banking, Technology, Clean Energy, etc.)

- Runs 4 strategies on every stock in every sector
- Compares performance ACROSS sectors
- Identifies best opportunities from entire universe

Default Sectors:

```
Banking: JPM, BAC, WFC, C, GS, MS, USB, PNC, TFC, COF
Technology: AAPL, MSFT, GOOGL, META, NVDA, AMD, CRM, ORCL, ADBE, INTC
Clean Energy: TSLA, NIO, RIVN, LCID, PLUG, SEDG, NEE, ICLN, ENPH
```

Best For: "Where should I invest across the entire market?"

⚠ **Note:** This is compute-intensive (120+ tool calls for 3 sectors × 10 stocks)

Tab 5: 🔄 Combined Analysis

Purpose: Merge Technical and Fundamental analysis for complete picture

Philosophy:

- **Fundamental Analysis** = "WHAT to buy" (company quality)
- **Technical Analysis** = "WHEN to buy" (timing)
- **Combined** = 360-degree investment view

Signal Alignment:

FA Signal	TA Signal	Interpretation
Bullish	Bullish	☑ High conviction BUY
Bullish	Bearish	⚠ Good company, bad timing - WAIT
Bearish	Bullish	⚠ Technical bounce, weak fundamentals - CAUTION
Bearish	Bearish	❌ High conviction AVOID

Best For: "Give me the complete investment picture"

🚀 Quick Start

Prerequisites

- Python 3.10+
- OpenAI API key (recommended) or HuggingFace token
- Internet connection (Yahoo Finance data)

Installation

```
# Clone the repository
git clone <repository-url>
cd mcp_financial_markets_analysis_tool

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# or
.\venv\Scripts\activate   # Windows

# Install dependencies
pip install -r requirements.txt
```

Environment Setup

Create a `.env` file in the project root:

```
# Required - LLM API Key (choose one)
OPENAI_API_KEY=sk-your-openai-key-here
# OR
HF_TOKEN=hf_your-huggingface-token

# Optional - Model Configuration
SMOLAGENT_MODEL_ID=gpt-4o           # Default model
SMOLAGENT_MODEL_PROVIDER=litellm     # litellm or inference
SMOLAGENT_MAX_STEPS=25               # Max reasoning steps

# Optional - Defaults
DEFAULT_ANALYSIS_PERIOD=1y
DEFAULT_SCANNER_SYMBOLS=AAPL,MSFT,GOOGL,AMZN
```

Running the Application

```
# Terminal 1: Start the FastAPI backend
uvicorn stock_analyzer_bot.api:app --reload --port 8000

# Terminal 2: Start the Streamlit frontend
streamlit run streamlit_app.py
```

Open your browser to `http://localhost:8501`

🔧 Core Components

1. MCP Server (`server/`)

The **Model Context Protocol Server** provides all financial analysis tools. It's a standalone process that the bot connects to via stdio.

Key Features:

- 5 technical analysis strategies
- Performance backtesting with metrics
- Fundamental analysis from financial statements
- Multi-stock market scanner

Detailed Documentation: server/README.md

2. Stock Analyzer Bot (`stock_analyzer_bot/`)

The **smolagents-powered orchestration layer** that uses LLMs to call tools and generate reports.

Key Features:

- ToolCallingAgent with 7 wrapped tools
- 5 analysis functions for different use cases
- Intelligent prompt engineering
- FastAPI REST endpoints

Detailed Documentation: stock_analyzer_bot/README.md

3. Streamlit Frontend (`streamlit_app.py`)

The **web interface** providing 5 analysis tabs with session history.

Key Features:

- 5 analysis type tabs
- Model configuration sidebar
- Analysis history tracking
- Markdown report rendering

API Reference

Available Endpoints

Endpoint	Method	Description
`/health`	GET	Health check and version info
`/technical`	POST	Single stock, 4 strategies
`/scanner`	POST	Multi-stock comparison
`/fundamental`	POST	Financial statement analysis
`/multisector`	POST	Cross-sector analysis
`/combined`	POST	Technical + Fundamental

Example API Call

```
curl -X POST "http://localhost:8000/technical" \
  -H "Content-Type: application/json" \
  -d '{"symbol": "AAPL", "period": "1y"}'
```

Response Format

```
{
  "report": "# AAPL Comprehensive Technical Analysis\n...",
  "symbol": "AAPL",
  "analysis_type": "technical",
  "duration_seconds": 45.2
}
```

⚙ Configuration

Supported LLM Models

Provider	Model ID	Best For
OpenAI	`gpt-4o`	Best quality, recommended
OpenAI	`gpt-4o-mini`	Faster, cost-effective
OpenAI	`gpt-4-turbo`	Good balance
HuggingFace	`meta-llama/Llama-3.1-70B-Instruct`	Open source

Analysis Periods

Valid periods: `1d`, `5d`, `1mo`, `3mo`, `6mo`, `1y`, `2y`, `5y`, `10y`, `ytd`, `max`

Strategy Parameters

Strategy	Key Parameters
Bollinger-Fibonacci	window=20, num_std=2
MACD-Donchian	fast=12, slow=26, signal=9
Connors RSI	rsi_period=3, streak=2, rank=100
Dual MA	short=50, long=200, type=EMA

🖌 Example Reports

Technical Analysis Report Structure

```
# AAPL Comprehensive Technical Analysis
*Analysis Date: 2024-01-15*
*Current Price: $185.92*

## Executive Summary
[2-3 paragraphs synthesizing all strategy findings]

## Strategy Performance Comparison
| Strategy | Signal | Score | Return | Sharpe | Max DD |
|----------|--------|-------|--------|--------|--------|
| Bollinger-Fib | BUY | +45 | 12.3% | 1.2 | -8.5% |
| MACD-Donchian | HOLD | +15 | 8.1% | 0.9 | -12.1% |
| ... | ... | ... | ... | ... | ... |

## Individual Strategy Analysis
[Detailed breakdown of each strategy]

## Risk Assessment
[Volatility, drawdown analysis]

## Final Recommendation: **BUY**
[Supporting rationale]
```

Security & Disclaimers

API Key Security

- Never commit `.env` files to version control
- Use environment variables for all sensitive data
- API keys are never logged or stored

Financial Disclaimer

⚠ **IMPORTANT:** This software is for **educational and research purposes only**.

- All analysis results should be independently verified
- Past performance does not guarantee future results
- This is NOT financial advice
- Consult a licensed financial advisor before investing
- The authors assume no liability for investment decisions

Data Sources

- Market data from Yahoo Finance (subject to their terms of service)
- Data may have delays, gaps, or inaccuracies
- Always verify data against official sources

Troubleshooting

Issue	Solution
"MCP server not found"	Verify `server/main.py` exists at project root
"Connection refused"	Start FastAPI: `uvicorn stock_analyzer_bot.api:app --port 8000`
"Authentication error"	Check `OPENAI_API_KEY` in `.env`
"Timeout"	Reduce number of stocks or increase timeout
"Agent stopped early"	Increase `max_steps` parameter

Additional Documentation

Document	Description
server/README.md	MCP Server tools, strategies, parameters
stock_analyzer_bot/README.md	Smolagents integration, API endpoints
HuggingFace Smolagents Docs	Official smolagents documentation
MCP Documentation	Model Context Protocol specification

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/new-strategy`)
3. Implement your changes
4. Add tests if applicable
5. Submit a pull request

Adding New Strategies

1. Create strategy module in `server/strategies/`
2. Register with MCP server in `server/main.py`
3. Create tool wrapper in `stock_analyzer_bot/tools.py`
4. Update prompts in `stock_analyzer_bot/main.py`

License

This project is provided for educational purposes. Users must comply with:

- Yahoo Finance Terms of Service
- OpenAI / HuggingFace Terms of Service
- Applicable local financial regulations

Acknowledgments

- Smolagents by Hugging Face

- FastMCP for MCP framework
- yfinance for market data
- FastAPI for REST API
- Streamlit for web interface
