**AInvestorAgent 投资前检查清单**

**版本**: 1.0  
**更新日期**: 2025-10-15  
**适用场景**: 真实资金投资前的全面验证

**📋 使用说明**

* **目标**: 确保系统在真实投资前万无一失
* **要求**: 每一项都必须打勾（✓）才能进入下一阶段
* **时间**: 完整检查预计需要 3-5 天
* **建议**: 按阶段执行，不要跳过任何步骤

**🎯 阶段 0: 环境准备（15分钟）**

**0.1 系统运行状态**

* [ ] 后端正常启动（python run.py，无报错）
* [ ] 前端正常启动（npm run dev，无报错）
* [ ] 能访问 http://localhost:8000/health 返回 {"status":"ok"}
* [ ] 能访问 http://localhost:3000 看到仪表盘

**0.2 数据库状态**

* [ ] 数据库文件存在（db/AInvestorAgent.sqlite）
* [ ] 数据库大小 > 10MB（表明有足够数据）
* [ ] 运行数据验证脚本：
* python scripts/validate\_data.py
* [ ] **验证结果**: ✓ 所有验证通过

**0.3 API密钥确认**

* [ ] .env 文件包含所有必需的API密钥
* [ ] AlphaVantage API密钥有效（测试拉取AAPL价格）
* [ ] News API密钥有效（测试拉取AAPL新闻）
* [ ] LLM API密钥有效（测试情绪分析）

**✅ 阶段0完成标志**: 所有项打勾，系统可正常访问

**🔍 阶段 1: 数据质量验证（1天）**

**1.1 价格数据完整性**

运行以下测试：

python -c "

from backend.storage.db import SessionLocal

from backend.storage.models import PriceDaily

from datetime import datetime, timedelta

db = SessionLocal()

symbols = ['AAPL', 'MSFT', 'GOOGL', 'TSLA', 'NVDA', 'SPY']

one\_year\_ago = datetime.now().date() - timedelta(days=365)

for symbol in symbols:

count = db.query(PriceDaily).filter(

PriceDaily.symbol == symbol,

PriceDaily.date >= one\_year\_ago

).count()

print(f'{symbol}: {count} 个数据点 (需要≥200)')

db.close()

"

**检查项**:

* [ ] AAPL: ≥200 个数据点
* [ ] MSFT: ≥200 个数据点
* [ ] GOOGL: ≥200 个数据点
* [ ] TSLA: ≥200 个数据点
* [ ] NVDA: ≥200 个数据点
* [ ] SPY: ≥200 个数据点（基准必需）
* [ ] 最新价格数据 ≤ 2天前
* [ ] 无缺失字段（close, volume非NULL）

**1.2 新闻与情绪数据**

python -c "

from backend.storage.db import SessionLocal

from backend.storage.models import NewsRaw, NewsScore

db = SessionLocal()

news\_count = db.query(NewsRaw).count()

scored\_count = db.query(NewsScore).count()

print(f'新闻总数: {news\_count}')

print(f'已评分: {scored\_count}')

print(f'评分覆盖率: {scored\_count/news\_count\*100:.1f}%')

db.close()

"

**检查项**:

* [ ] 新闻总数 ≥ 500 条
* [ ] 评分覆盖率 ≥ 80%
* [ ] 情绪分数在 -1 到 1 范围内
* [ ] 最近7天有新闻更新

**1.3 基本面数据**

python -c "

from backend.storage.db import SessionLocal

from backend.storage.models import Fundamental

db = SessionLocal()

symbols = ['AAPL', 'MSFT', 'GOOGL', 'TSLA', 'NVDA']

for symbol in symbols:

fund = db.query(Fundamental).filter(Fundamental.symbol == symbol).first()

if fund:

print(f'{symbol}: PE={fund.pe}, PB={fund.pb}, ROE={fund.roe}')

else:

print(f'{symbol}: 无基本面数据')

db.close()

"

**检查项**:

* [ ] 至少5支主要股票有基本面数据
* [ ] PE、PB、ROE字段有值（非NULL）
* [ ] 市值（market\_cap）有值

**1.4 手工数据抽查**

随机选择3支股票，对比系统数据与外部源（如Yahoo Finance）：

**股票1: AAPL**

* [ ] 最新收盘价误差 < 1%
* [ ] PE比率误差 < 10%
* [ ] 最近新闻标题能对应到实际新闻

**股票2: MSFT**

* [ ] 最新收盘价误差 < 1%
* [ ] PE比率误差 < 10%
* [ ] 最近新闻标题能对应到实际新闻

**股票3: SPY**

* [ ] 最新收盘价误差 < 1%
* [ ] 作为基准数据完整

**✅ 阶段1完成标志**: 所有数据质量检查通过

**🧮 阶段 2: 因子与评分验证（半天）**

**2.1 因子计算正确性**

测试单支股票的因子计算：

curl -X POST "http://localhost:8000/api/analyze/AAPL" | python -m json.tool

**检查项**:

* [ ] 返回状态码 200
* [ ] 包含 factors 字段
* [ ] 四个因子都存在：value, quality, momentum, sentiment
* [ ] 每个因子值在 0-1 范围内
* [ ] score 字段存在且在 0-100 范围

**2.2 评分公式验证**

手工验证评分公式：score = 100 \* (0.25\*value + 0.20\*quality + 0.35\*momentum + 0.20\*sentiment)

**测试AAPL**:

# 假设返回的因子值为:

# value: 0.65, quality: 0.80, momentum: 0.55, sentiment: 0.70

# 手工计算: 100 \* (0.25\*0.65 + 0.20\*0.80 + 0.35\*0.55 + 0.20\*0.70) = 64.75

* [ ] 手工计算与系统返回评分误差 < 1.0
* [ ] version\_tag 字段存在（如 "v1.0.0"）

**2.3 批量评分性能**

curl -X POST "http://localhost:8000/api/score/batch" \

-H "Content-Type: application/json" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","TSLA","NVDA","AMZN","META","NFLX","DIS","COST"]}' \

| python -m json.tool

**检查项**:

* [ ] 返回10支股票的评分
* [ ] 响应时间 < 10秒
* [ ] 所有评分在 0-100 范围
* [ ] 评分有差异性（不是所有都一样）

**✅ 阶段2完成标志**: 因子和评分系统工作正常

**📊 阶段 3: 组合构建验证（半天）**

**3.1 组合生成测试**

curl -X POST "http://localhost:8000/api/portfolio/propose" \

-H "Content-Type: application/json" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","TSLA","NVDA","AMZN","META","NFLX","DIS","COST"]}' \

| python -m json.tool > portfolio\_test.json

**检查项**:

* [ ] 返回状态码 200
* [ ] holdings 数组包含 5-15 支股票
* [ ] 权重总和 = 100% (±0.5%)
* [ ] 单票权重 ≤ 30%
* [ ] 每支股票有 reasons 字段（入选理由）
* [ ] 包含 sector\_concentration 字段
* [ ] 行业集中度 ≤ 50%

**3.2 约束机制验证**

测试极端输入：

**测试1: 全部科技股**

curl -X POST "http://localhost:8000/api/portfolio/propose" \

-H "Content-Type: application/json" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","META","NVDA","AMD","INTC","CRM","ORCL","ADBE"]}'

* [ ] 系统限制科技行业 ≤ 50% 或给出警告

**测试2: 仅3支股票**

curl -X POST "http://localhost:8000/api/portfolio/propose" \

-H "Content-Type: application/json" \

-d '{"symbols": ["AAPL","MSFT","GOOGL"]}'

* [ ] 系统拒绝或调整至最小5支

**测试3: 空列表**

curl -X POST "http://localhost:8000/api/portfolio/propose" \

-H "Content-Type: application/json" \

-d '{"symbols": []}'

* [ ] 返回 400 或 422 错误

**3.3 可复现性测试**

运行两次相同的组合生成请求：

* [ ] 两次返回的持仓列表完全一致
* [ ] 两次返回的权重完全一致（相同 version\_tag）
* [ ] 证明系统是确定性的，不是随机的

**✅ 阶段3完成标志**: 组合构建逻辑正确且稳定

**🔄 阶段 4: 回测验证（1-2天）**

**4.1 基础回测功能**

curl -X POST "http://localhost:8000/api/backtest/run" \

-H "Content-Type: application/json" \

-d '{

"holdings": [

{"symbol": "AAPL", "weight": 25},

{"symbol": "MSFT", "weight": 25},

{"symbol": "GOOGL", "weight": 25},

{"symbol": "NVDA", "weight": 25}

],

"window": "1Y",

"rebalance": "weekly",

"cost": 0.001

}' | python -m json.tool > backtest\_1y.json

**检查项**:

* [ ] 返回状态码 200
* [ ] 包含 dates, nav, benchmark\_nav, metrics
* [ ] 初始净值 = 1.0
* [ ] 最终净值 > 0（没有破产）
* [ ] metrics 包含: annualized\_return, sharpe, max\_dd, win\_rate
* [ ] Sharpe比率不是 NaN 或 Inf
* [ ] 最大回撤在合理范围（通常 -5% 到 -40%）

**4.2 回测可复现性**

运行两次相同参数的回测：

# 第一次

curl -X POST "http://localhost:8000/api/backtest/run" \

-d '{"holdings": [...], "window": "6M"}' > backtest\_run1.json

# 第二次

curl -X POST "http://localhost:8000/api/backtest/run" \

-d '{"holdings": [...], "window": "6M"}' > backtest\_run2.json

# 对比

diff backtest\_run1.json backtest\_run2.json

* [ ] 两次回测的最终净值完全一致（误差 < 0.0001）
* [ ] 所有指标完全一致

**4.3 回测合理性检查**

分析 backtest\_1y.json 的结果：

**年化收益率检查**:

* [ ] 如果 > 100%: 警告（可能过度乐观）
* [ ] 如果 < -50%: 警告（可能有bug）
* [ ] 合理范围: -20% 到 +60%

**Sharpe比率检查**:

* [ ] 如果 > 3: 警告（可能过拟合）
* [ ] 如果 < 0: 说明策略输给无风险利率
* [ ] 合理范围: 0.5 到 2.0

**最大回撤检查**:

* [ ] 如果 > -50%: 警告（风险过高）
* [ ] 如果 = 0%: 可疑（可能数据问题）
* [ ] 合理范围: -10% 到 -30%

**胜率检查**:

* [ ] 如果 > 80%: 警告（可能过拟合）
* [ ] 如果 < 30%: 说明策略胜率低
* [ ] 合理范围: 45% 到 65%

**4.4 不同窗口期对比**

运行多个窗口期的回测，检查一致性：

# 6个月

curl -X POST "http://localhost:8000/api/backtest/run" \

-d '{"holdings": [...], "window": "6M"}' > backtest\_6m.json

# 1年

curl -X POST "http://localhost:8000/api/backtest/run" \

-d '{"holdings": [...], "window": "1Y"}' > backtest\_1y.json

# 2年

curl -X POST "http://localhost:8000/api/backtest/run" \

-d '{"holdings": [...], "window": "2Y"}' > backtest\_2y.json

**检查项**:

* [ ] 3个窗口期都能正常运行
* [ ] Sharpe比率方向一致（都>0或都<0）
* [ ] 没有明显矛盾（如6M巨亏但1Y大赚）

**4.5 与基准对比**

* [ ] SPY基准净值曲线合理（接近真实SPY走势）
* [ ] 组合与SPY的相对表现符合预期
* [ ] 计算超额收益：Portfolio Return - SPY Return

**目标**:

* [ ] 超额收益（Alpha）> 0%（跑赢基准）
* [ ] 如果Alpha < 0，理解原因（如高成本、高换手）

**✅ 阶段4完成标志**: 回测功能正确，结果合理

**🤖 阶段 5: 智能体协同验证（1天）**

**5.1 完整决策链路测试**

curl -X POST "http://localhost:8000/api/orchestrator/decide" \

-H "Content-Type: application/json" \

-d '{"topk": 10, "mock": false}' \

| python -m json.tool > orchestrator\_test.json

**检查项**:

* [ ] 返回状态码 200
* [ ] 包含 trace\_id（用于追踪）
* [ ] 包含 holdings（最终组合）
* [ ] 持仓数 5-15 支
* [ ] 权重总和 ≈ 100%
* [ ] 响应时间 < 120秒

**5.2 Trace可追溯性**

使用返回的 trace\_id 查询执行链路：

# 假设 trace\_id = "abc123..."

curl "http://localhost:8000/api/trace/abc123..." | python -m json.tool

**检查项**:

* [ ] 能查询到该trace的详细记录
* [ ] 显示各Agent执行顺序
* [ ] 显示各Agent执行状态（success/failed）
* [ ] 显示各Agent耗时

**5.3 风险管理验证**

检查 orchestrator\_test.json 中的组合：

* [ ] 单票权重 ≤ 30%（风险控制生效）
* [ ] 行业权重 ≤ 50%（风险控制生效）
* [ ] 有 actions 或 adjustments 字段说明风控调整

**5.4 多次决策一致性**

运行3次决策，检查稳定性：

for i in 1 2 3; do

curl -X POST "http://localhost:8000/api/orchestrator/decide" \

-d '{"topk": 10}' > decide\_run$i.json

done

**检查项**:

* [ ] 3次都能成功执行
* [ ] 3次返回的股票池有 ≥50% 重合度
* [ ] 3次的总体策略方向一致
* [ ] 权重分配相对稳定（波动 < 10%）

**✅ 阶段5完成标志**: 智能体协同工作正常，决策稳定

**🖥️ 阶段 6: 前端功能验证（半天）**

**6.1 首页/仪表盘**

访问 http://localhost:3000

**检查项**:

* [ ] Portfolio Snapshot 卡片显示数据
* [ ] Watchlist Rankings 表格有数据
* [ ] News & Sentiment 显示最近新闻
* [ ] Last Backtest 显示历史回测
* [ ] 所有数据有时间戳
* [ ] 无JavaScript错误（F12检查控制台）

**6.2 个股页面**

访问个股页（如搜索 AAPL）

**检查项**:

* [ ] 价格图表正确渲染（折线图 + MA线）
* [ ] 基本面卡片显示 PE, PB, ROE
* [ ] 因子雷达图显示4个维度
* [ ] 动量条形图显示 1M/3M/12M
* [ ] 新闻情绪时间轴有数据点
* [ ] 点击新闻可跳转到原文

**6.3 组合页面**

生成一个组合后访问组合页

**检查项**:

* [ ] 权重饼图正确显示
* [ ] 行业条形图正确显示
* [ ] 持仓明细表包含：代码、权重、分数、理由
* [ ] 可导出CSV文件
* [ ] CSV内容正确

**6.4 模拟器页面**

运行回测后访问模拟器页

**检查项**:

* [ ] 净值曲线（组合 vs 基准）双线显示
* [ ] 回撤图显示
* [ ] 指标面板显示：年化、Sharpe、回撤、胜率
* [ ] 参数展示：窗口、成本、频率、版本
* [ ] 可导出PNG和CSV

**6.5 监控页面**

访问 Monitor 页面

**检查项**:

* [ ] 情绪时间轴显示14天数据
* [ ] 新闻列表可滚动
* [ ] 情绪徽标颜色正确（绿/红/灰）
* [ ] 无数据时显示占位符

**6.6 响应式测试**

* [ ] 在1920x1080分辨率下正常显示
* [ ] 在1366x768分辨率下正常显示
* [ ] 所有按钮可点击
* [ ] 无横向滚动条（除列表内）

**✅ 阶段6完成标志**: 所有前端功能可用且美观

**⚡ 阶段 7: 性能与稳定性测试（1天）**

**7.1 API响应时间测试**

测试关键端点的性能：

# 健康检查

time curl "http://localhost:8000/health"

# 价格查询

time curl "http://localhost:8000/api/prices/AAPL?range=1Y"

# 批量评分

time curl -X POST "http://localhost:8000/api/score/batch" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","TSLA","NVDA"]}'

# 组合生成

time curl -X POST "http://localhost:8000/api/portfolio/propose" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","TSLA","NVDA","AMZN"]}'

**性能目标**:

* [ ] 健康检查: < 500ms
* [ ] 价格查询: < 1s
* [ ] 批量评分: < 5s
* [ ] 组合生成: < 3s
* [ ] 回测1年: < 20s

**7.2 并发测试**

使用简单脚本测试并发：

# 创建 concurrent\_test.sh

for i in {1..5}; do

curl "http://localhost:8000/health" &

done

wait

**检查项**:

* [ ] 5个并发请求都成功
* [ ] 无502或504错误
* [ ] 响应时间增长 < 2倍

**7.3 长时间运行测试**

让系统持续运行24小时：

**Day 1 开始前**:

* [ ] 记录内存使用: \_\_\_\_\_\_ MB
* [ ] 记录CPU使用: \_\_\_\_\_\_ %

**Day 1 结束后**:

* [ ] 记录内存使用: \_\_\_\_\_\_ MB
* [ ] 内存增长 < 500MB
* [ ] 无内存泄漏迹象
* [ ] 系统仍可正常响应
* [ ] 日志文件大小 < 100MB

**7.4 错误处理测试**

测试各种错误场景：

# 404 错误

curl "http://localhost:8000/api/nonexistent"

# 预期: 返回 404

# 空参数

curl -X POST "http://localhost:8000/api/portfolio/propose" -d '{"symbols": []}'

# 预期: 返回 400 或 422

# 无效股票代码

curl "http://localhost:8000/api/prices/ZZZZZ"

# 预期: 返回 404 或优雅降级

**检查项**:

* [ ] 所有错误返回适当的HTTP状态码
* [ ] 错误信息清晰易懂
* [ ] 不泄露内部错误细节
* [ ] 系统不会因错误崩溃

**✅ 阶段7完成标志**: 系统性能达标，稳定可靠

**🧪 阶段 8: 实盘模拟（7-30天）**

**8.1 纸上交易准备**

创建虚拟账户记录表：

**模拟账户信息**:

* 初始资金: $10,000
* 开始日期: \_\_\_\_\_\_\_\_\_\_
* 预计结束日期: \_\_\_\_\_\_\_\_\_\_（至少7天后）

**8.2 每周操作流程**

**每周一上午9:00（或市场开盘前）**:

1. **更新数据**:

# 更新价格

python scripts/fetch\_prices.py --symbols AAPL,MSFT,GOOGL,TSLA,NVDA,SPY --range 1Y

# 更新新闻

python scripts/fetch\_news.py --symbols AAPL,MSFT,GOOGL,TSLA,NVDA,SPY --days 14 --noproxy

# 重算因子和评分

python scripts/rebuild\_factors.py --symbols AAPL,MSFT,GOOGL,TSLA,NVDA,SPY

python scripts/recompute\_scores.py --symbols AAPL,MSFT,GOOGL,TSLA,NVDA,SPY

1. **生成组合建议**:

curl -X POST "http://localhost:8000/api/orchestrator/decide" \

-d '{"topk": 10}' > weekly\_decision\_$(date +%Y%m%d).json

1. **记录虚拟持仓**:
   * 打开生成的JSON文件
   * 记录到Excel或Google Sheets：
     + 日期
     + 股票代码
     + 建议权重
     + 虚拟买入价格（当日开盘价）
     + 入选理由
2. **计算虚拟净值**:
   * 每周五收盘后计算账户总值
   * 记录收益率
   * 记录最大回撤

**8.3 模拟交易记录表**

| **日期** | **操作** | **股票** | **数量** | **价格** | **金额** | **余额** | **备注** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2025-10-20 | 初始 | - | - | - | - | $10,000 | 开始 |
| 2025-10-20 | 买入 | AAPL | 15 | $175.00 | $2,625 | $7,375 | 26.25% |
| ... | ... | ... | ... | ... | ... | ... | ... |

**8.4 每周检查清单**

**Week 1** (**/**):

* [ ] 数据更新成功
* [ ] 系统生成组合建议
* [ ] 虚拟买入执行
* [ ] 无技术故障
* [ ] 本周收益率: \_\_\_\_\_%

**Week 2** (**/**):

* [ ] 数据更新成功
* [ ] 系统生成调仓建议
* [ ] 虚拟调仓执行
* [ ] 无技术故障
* [ ] 累计收益率: \_\_\_\_\_%
* [ ] 最大回撤: \_\_\_\_\_%

**Week 3** (**/**):

* [ ] 数据更新成功
* [ ] 系统生成调仓建议
* [ ] 虚拟调仓执行
* [ ] 无技术故障
* [ ] 累计收益率: \_\_\_\_\_%
* [ ] 最大回撤: \_\_\_\_\_%

**Week 4** (**/**):

* [ ] 数据更新成功
* [ ] 系统生成调仓建议
* [ ] 虚拟调仓执行
* [ ] 无技术故障
* [ ] 累计收益率: \_\_\_\_\_%
* [ ] 最大回撤: \_\_\_\_\_%
* [ ] vs SPY收益: \_\_\_\_\_%

**8.5 实盘模拟通过标准**

**最低标准（必须满足）**:

* [ ] 30天内无系统崩溃
* [ ] 累计收益 > -10%（不能亏太多）
* [ ] 最大回撤 < 15%（风险可控）
* [ ] 无异常交易（如权重>30%）
* [ ] 至少3次成功调仓

**理想标准（鼓励达到）**:

* [ ] 累计收益 > 0%（盈利）
* [ ] Sharpe比率 > 0.5
* [ ] 跑赢SPY基准
* [ ] 换手率 < 30%/月

**⚠️ 如果出现以下情况，必须停止并检查**:

* [ ] 单周亏损 > 10%
* [ ] 累计回撤 > 20%
* [ ] 系统连续2次崩溃
* [ ] 出现明显不合理的建议（如全仓一只股票）

**✅ 阶段8完成标志**: 实盘模拟表现符合预期，无重大问题

**📝 阶段 9: 人工审核与合规（1天）**

**9.1 投资逻辑审核**

随机选择10支系统推荐的股票，人工验证逻辑：

**股票1: \_\_\_\_\_\_**

* [ ] 入选理由合理（高ROE+低PE确实成立）
* [ ] 评分计算正确
* [ ] 无明显错误信号

**股票2-10: 重复以上检查**

* [ ] ...
* [ ] ...

**9.2 风险评估**

邀请至少1位有金融背景的朋友审核系统：

**审核人**: \_\_\_\_\_\_\_\_\_\_ **审核日期**: \_\_\_\_\_\_\_\_\_\_

**审核要点**:

* [ ] 组合是否过度集中
* [ ] 是否忽略已知风险（如某公司财务丑闻）
* [ ] 回测是否过拟合（Sharpe > 3）
* [ ] 是否有明显逻辑漏洞

**审核意见**:

**9.3 免责声明确认**

* [ ] 我理解系统输出仅供参考
* [ ] 我理解不构成投资建议
* [ ] 我理解需要自己做最终决策
* [ ] 我理解投资有风险
* [ ] 我已设置止损线（-15%）

**9.4 合规检查**

* [ ] 所有股票都可交易（未停牌、未退市）
* [ ] 没有违规行为（内幕交易、操纵市场）
* [ ] 前端显示免责声明

**✅ 阶段9完成标志**: 通过人工审核，符合合规要求

**🎯 阶段 10: 最终确认（关键！）**

**10.1 技术就绪度**

* [ ] 所有阶段0-9的检查项都已打勾
* [ ] 没有P0级别的未解决问题
* [ ] 系统稳定运行至少7天
* [ ] 实盘模拟通过标准

**10.2 心理准备**

* [ ] 我理解可能会亏损
* [ ] 我不会因为系统建议而完全放弃自己的判断
* [ ] 我会定期（至少每周）监控表现
* [ ] 我设定了明确的止损线（-15%）
* [ ] 我准备好了应急预案

**10.3 资金计划**

**初始投资计划**:

* 第一批投资: $10,000（总资金的10%）
* 观察期: 1个月
* 如果表现良好（收益>0%, 回撤<10%）:
  + 第二批: +$10,000
  + 第三批: +$30,000
  + 目标上限: $50,000
* 如果触发止损（-15%）:
  + 立即停止投资
  + 全面检查系统
  + 找出问题根源

**10.4 监控计划**

**每日**:

* [ ] 检查系统运行状态（健康检查）
* [ ] 记录持仓变化

**每周**:

* [ ] 对比系统建议 vs 实际表现
* [ ] 记录收益率和回撤
* [ ] 生成周报

**每月**:

* [ ] 运行完整测试套件
* [ ] 评估因子有效性
* [ ] 调整策略权重（如需要）
* [ ] 生成月报

**10.5 应急预案**

**如果出现以下情况，立即暂停系统并人工介入**:

1. **技术故障**:
   * 系统连续2次崩溃
   * 数据更新失败超过3天
   * API返回大量错误
2. **表现异常**:
   * 单周亏损 > 10%
   * 累计回撤 > 15%（触发止损）
   * 连续3周跑输SPY超过5%
3. **市场剧变**:
   * 市场单日暴跌 > 5%
   * 重大黑天鹅事件
   * 持仓股票出现重大利空

**应急操作**:

1. 暂停自动决策
2. 手工审核所有持仓
3. 考虑减仓或清仓
4. 等待市场稳定后再重启

**✅ 最终签字确认**

**我郑重声明**:

* [x] 我已仔细阅读并完成本检查清单的所有项目
* [x] 我理解系统的局限性和风险
* [x] 我已设置止损线为 -15%
* [x] 我不会投入超出承受能力的资金
* [x] 我会持续监控系统表现
* [x] 我明白这不是金融建议

**初始投资金额**: $\_\_\_\_\_\_\_\_ **止损金额**: $\_\_\_\_\_\_\_\_ **开始日期**: \_\_\_\_\_\_\_\_\_\_

**签名**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **日期**: \_\_\_\_\_\_\_\_\_\_

**📊 附录A: 快速自检命令**

复制以下命令进行快速自检：

# 1. 健康检查

curl http://localhost:8000/health

# 2. 数据验证

python scripts/validate\_data.py

# 3. 单股测试

curl -X POST "http://localhost:8000/api/analyze/AAPL" | python -m json.tool

# 4. 组合测试

curl -X POST "http://localhost:8000/api/portfolio/propose" \

-H "Content-Type: application/json" \

-d '{"symbols": ["AAPL","MSFT","GOOGL","TSLA","NVDA"]}' | python -m json.tool

# 5. 回测测试

curl -X POST "http://localhost:8000/api/backtest/run" \

-H "Content-Type: application/json" \

-d '{"holdings": [{"symbol":"AAPL","weight":50},{"symbol":"MSFT","weight":50}], "window": "6M"}' \

| python -m json.tool

# 6. 完整决策

curl -X POST "http://localhost:8000/api/orchestrator/decide" \

-H "Content-Type: application/json" \

-d '{"topk": 10}' | python -m json.tool

**📚 附录B: 常见问题解答**

**Q1: 某个测试失败了怎么办？** A: 先确定是P0（阻断）还是P1/P2（非阻断）问题。P0必须修复，P1/P2可以记录并在投资后逐步改进。

**Q2: 实盘模拟亏损超过10%，还能继续吗？** A: 不建议。应先停止，分析原因，优化策略后重新模拟。

**Q3: 系统建议与我的判断冲突，听谁的？** A: 相信自己的判断！系统只是辅助工具，你才是最终决策者。

**Q4: 多久重新运行一次完整测试？** A: 建议每月运行一次完整检查清单，确保系统持续健康。

**Q5: 止损后多久可以重启？** A: 至少等待1周，找出问题根源，修复后重新实盘模拟至少2周。

**版本历史**:

* v1.0 (2025-10-15): 初始版本
* 基于设计文档、测试文档、已有模块文档编制
* 针对$10,000初始投资规模定制