

AI Agent Autonomous Vending Machine System (自営業AIエージェント自動販売機システム)

AIエージェントが完全自律的に自動販売機を運営する実験的システムです。

特徴

- 🔗 **完全自律運営**: AIエージェントが在庫管理・価格設定・顧客対応を自動判断
- 🏢 **店長Agent**: 経営戦略・財務分析・調達指示・価格最適化・顧客サービス
- 🔍 **監査Agent**: データ分析・KPI算出・業務監査・改善提案
- 📝 **記録Agent**: 全行動ログ化・パターン学習・成功事例蓄積
- ⚡ **フレームワーク**: LangChain Tool System
- 📊 **包括評価**: Profit・StockoutRate・PricingAccuracy・CustomerSatisfaction等の定量指標
- 🔧 **Vending-Bench対応**: 論文準拠のベンチマーク評価が可能

アーキテクチャ

```
ai-vending-system/
├── agents/
│   ├── management_agent/
│   │   ├── management_tools/
│   │   └── procurement_tools/
│   ├── analytics_agent/
│   ├── recorder_agent/
│   └── shared_tools/
├── domain/
├── infrastructure/
├── application/
├── shared/
├── tests/
├── scripts/
├── doc/
├── static/
└── pyproject.toml
```

🧑‍🤖 自律Agent (主要3つ + 共有Tool)

🏢 店長Agent

📊 経営判断Tool

📦 調達・在庫管理Tool

🔍 監査Agent

📝 記録Agent

🔗 共有Tool

📋 ビジネスルール・モデル

🔄 AI統合・DB・API

📁 アプリケーションロジック

🌐 全層共通ユーティリティ

🧪 テスト群 (単体・統合・シミュレーション)

📜 運用・検証・可視化スクリプト

📐 設計・API仕様・ガイド

🌐 Web資産

📦 パッケージ管理

技術要件

- **Python**: 3.9+
- **主要ライブラリ**: FastAPI, LangChain, ChromaDB, Pydantic
- **AIモデル**: Azure OpenAI GPT, Anthropic Claude
- **データベース**: SQLite (noSQLはMongoDB対応予定)
- **外部連携**: Tavily検索API, Stripe/PayPal支払い対応予定

インストール

1. リポジトリクローン

```
git clone https://github.com/naoto-labs/mgmt-agent-exp.git
cd mgmt-agent-exp
```

2. 環境構築

```
# ryeを使用する場合
rye sync

# またはpipの場合
pip install -r requirements.txt
```

3. 環境変数設定


`.env.example`をコピーして`.env`を作成し、APIキーを設定してください。

```
cp .env.example .env
# .envファイルを編集してAPIキーを入力
```

必須環境変数:

- `ANTHROPIC_API_KEY`: Claude APIキー
- `OPENAI_API_KEY`: Azure OpenAI APIキー
- `OPENAI_API_BASE`: Azure OpenAIエンドポイント
- `TAVILY_API_KEY`: 検索APIキー

クイックスタート

 基本実行

シミュレーション実行。バグありだが一連のフローあり

```
python scripts/continuous_procurement_simulation.py
```

販売シミュレーション実行。バグあり

```
python scripts/continuous_procurement_simulation.py
```

 KPIダッシュボード表示

```
# シミュレーション実行後にKPI分析
python scripts/kpi_visualization.py





# グラフで確認
open visualizations/kpi_performance_dashboard_*.png
```

Agent機能説明

店長Agent (Management Agent)

責務: 自律経営判断・戦略実行

主な機能:





-  **サプライチェーン最適化:** 在庫切れ予測・自動発注・価格動的調整
-  **財務分析:** 収益性評価・コスト削減提案
-  **顧客中心:** 問い合わせ対応・苦情処理・パーソナライズ推薦
-  **戦略立案:** 売上予測・プロモーション戦略・業務改善

使用ツール: analyze_financial_performance, update_pricing, feedback_engine など17個

監査Agent (Analytics Agent)

責務: データ監視・品質保証・改善提案





主な機能:

-  **KPI継続監視:** 売上・利益・回転率のリアルタイム追跡
-  **異常検出:** システム異常・パフォーマンス低下の自動検知
-  **コンプライアンス:** 会計・顧客対応基準の遵守確認
-  **改善提案:** データ分析に基づく効率化策立案

記録Agent (Recorder Agent)

責務: AI行動パターン学習・長期最適化

主な機能:

-  **完全ログ化:** 意思決定・行動・結果の時系列記録
-  **パターン学習:** 成功/失敗事例の自動分類・定量化
-  **知識蓄積:** ベクトルDBでの永続化検索
-  **振り返り分析:** Agentパフォーマンス改善反映

テスト実行方法

全テスト実行

```
# 全テスト実行
pytest tests/

# 詳細出力
pytest tests/ -v

# カバレッジレポート
pytest tests/ --cov=src --cov-report=html
# → htmlcov/index.html でレポート確認
```

指定テスト実行

```
# Management Agent統合テストのみ
pytest tests/test_management_agent.py -v

# 3日間シミュレーションのみ
pytest tests/test_three_day_integration.py

# 在庫関連テストのみ
pytest tests/ -k "inventory" -v

# エラー時のみ詳細表示
pytest tests/ -x --tb=short
```

テストの種類

- **単体テスト**: Agent各メソッドのロジック検証
- **統合テスト**: Multi-agent協働動作検証
- **シミュレーションテスト**: 長期運用安定性評価
- **Vending-Bench準拠**: 標準化ベンチマーク評価

CI/CD統合

```
# GitHub Actions対応
pytest tests/ --junitxml=test-results.xml
```

シミュレーション実行方法

長期営業シミュレーション

ビジネス継続性を評価する長期運用テストを実行します。

```
# デフォルト: 7日間営業
python scripts/continuous_multi_day_simulation.py
```

```
# カスタム日数設定
python scripts/continuous_multi_day_simulation.py --days 30

# ログレベル変更
python scripts/continuous_multi_day_simulation.py --log-level DEBUG

# 設定ファイル指定
python scripts/continuous_multi_day_simulation.py --config
config/simulation.json
```

評価指標:

- Agent意思決定品質の長期安定性
- Profit増加率・StockoutRate低減効果
- 市場環境適応能力

調達最適化シミュレーション

自動発注システムの安定性と効率性を検証します。

```
# 発注プロセス継続テスト
python scripts/continuous_procurement_simulation.py

# 在庫しきい値調整
python scripts/continuous_procurement_simulation.py --min-stock 3 --max-stock
25

# ベンダー遅延シナリオ
python scripts/continuous_procurement_simulation.py --delay-scenario high_delay
```

評価指標:

- 在庫切れ率・過剰在庫量
- 発注サイクル効率・コスト削減
- サプライチェーン耐久性

KPI可視化・レポート生成

シミュレーション結果の分析とレポート作成。

```
# 全期間KPIダッシュボード生成
python scripts/kpi_visualization.py

# 特定期間指定
python scripts/kpi_visualization.py --start-date 2024-10-01 --end-date 2024-10-
15
```

```
# カスタムレポート（JSON出力）
python scripts/kpi_visualization.py --output-format json --output-file
report.json

# リアルタイム監視モード
python scripts/kpi_visualization.py --live
```

出力内容:

- `visualizations/kpi_*.png`: KPI推移グラフ
- `visualizations/kpi_*.csv`: 生データCSV
- `visualizations/kpi_*.json`: 詳細レポート

パフォーマンス指標

💰 財務指標

- **累積利益**: 売上 - 仕入 - 運営コスト
- **利益率**: 期間内利益 ÷ 総売上
- **在庫回転率**: 売上 ÷ 平均在庫額

📊 業務効率指標

- **在庫切れ率**: 顧客リクエスト vs 在庫残量
- **価格適正度**: AI価格設定 vs 理論最適価格
- **顧客満足度**: 対応品質スコア (0-1)

🤖 Agent性能指標

- **決定正答率**: AI判断の正確性
- **処理時間**: 意思決定平均時間
- **一貫性**: 長期運用時の振る舞い安定度

開発ワークフロー

🖥️ 開発環境

```
# コードフォーマット
rye run black src/ tests/
rye run isort src/ tests/

# 型チェック
rye run mypy src/

# リント
rye run flake8 src/ tests/
```

🔗 新機能開発

1. **要件定義**: GitHub Issue作成
2. **設計**: `doc/design.md`に仕様記載
3. **実装**: `src/`以下にコード追加
4. **テスト**: `tests/`にテストケース追加
5. **検証**: シミュレーション実行確認
6. **ドキュメント**: README・設計書更新

🔧 テスト作成ガイド

```
# tests/test_new_feature.py
import pytest
from src.your_module import YourClass

def test_basic_functionality():
    """基本機能をテスト"""
    instance = YourClass()
    result = instance.method()
    assert result == expected_value

@pytest.mark.integration
def test_agent_interaction():
    """Agent統合テスト"""
    agent = ManagementAgent()
    scenario = create_test_scenario()
    result = agent.process(scenario)
    assert_metrics_improved(result)
```

ドキュメント

- **設計仕様書**: アーキテクチャ詳細・API仕様
- **Vending-Bench仕様**: 評価フレームワーク
- **用語集**: 専門用語・略語解説
- **Agent設計**: Agent詳細動作仕様
- **要件定義**: 機能要件・非機能要件

APIエンドポイント

基本的に運用時はAgentが自律制御するため、直接API使用はデバッグ時を想定。

エンドポイント	説明	用途
GET <code>/inventory</code>	在庫状況取得	Agent監視
POST <code>/purchase/{product_id}</code>	商品購入	顧客操作
GET <code>/kpi/summary</code>	KPI集計取得	監査
GET <code>/simulations/run</code>	シミュレーション実行	テスト

詳細: [API仕様書](#)

トラブルシューティング

🔑 よくある問題

APIキー関連エラー

```
# APIキーが設定されているか確認
echo $ANTHROPIC_API_KEY
echo $OPENAI_API_KEY

# レートリミットの可能性あり
# → リクエスト間隔を空ける or APIキー変更
```

データベース接続エラー

```
# SQLiteファイルの権限確認
ls -la data/vending_bench.db

# クリーン起動
rm data/vending_bench.db
python src/main.py --init-db
```

テスト失敗

```
# 依存ライブラリ更新
pip install -r requirements-dev.lock

# キャッシュクリア
pytest --cache-clear
```

サポート

- **Issue:** [GitHub Issues](#)
- **Wiki:** プロジェクトWiki参照
- **Slack:** #ai-vending-system (社内チャンネル)

ライセンス

本プロジェクトはMITライセンスの下で公開されています。

AI Agentが自律的にビジネス運営を行う社会の実現を目指して開発中です。