

OPTINVEST – RAPPORT

RABAH ACHOUR

YANIS NEDDAF

AHCENE DAHOUMANE

1. Description

OptInvest est une application d'analyse et de simulation de stratégies d'investissement. Elle permet :

- de comparer différentes approches d'investissement progressif (Dollar-Cost Averaging) et d'investissement en une seule fois (Lump Sum) sur des données réelles ;
 - de générer des prédictions futures à l'aide de modèles statistiques.
-

2. Installation pas-à-pas

1. Cloner le dépôt GitHub.
2. Installer l'outil *uv* :

```
pip install uv
```

3. Créer puis activer l'environnement virtuel :

```
uv venv
```

```
source .venv/bin/activate # sous Windows : .venv\Scripts\activate
```

4. Installer les dépendances :

```
uv add -r requirements.txt # ou : uv pip install -r requirements.txt
```

5. Lancer l'API backend :

```
python -m app.api.main
```

6. Lancer l'interface frontend dans un second terminal :

```
streamlit run app/client/main.py
```

7. Ouvrir le navigateur à l'adresse : <http://localhost:8501>
-

3. Fonctionnalités métier

Simulation de portefeuille

- Analyse historique d'actions et d'ETF (6 actifs autorisés pour l'instant).
- Fréquences d'investissement : mensuelle, trimestrielle, semestrielle ou annuelle.
- Prise en compte des frais de gestion annuels.
- Calcul des métriques : CAGR, rendement total, volatilité, Sharpe.

Comparaison de stratégies

- DCA (aux quatre fréquences) vs Lump Sum.
- Benchmark mondial : indice ACWI IMI (ETF SSAC.DE).
- Visualisations interactives des performances.

Modélisation prédictive

- Régression linéaire pour projeter l'évolution future des portefeuilles.
 - Analyse de la qualité des modèles (R^2 , RMSE, résidus).
 - Export des résultats en PDF et Excel.
-

4. Stack technique

- **Backend :**
 - FastAPI
 - Python 3.1
 - pandas / numpy
 - scikit-learn
 - yfinance
- **Frontend :**
 - Streamlit
 - Plotly
- **Rapports :**
 - ReportLab (PDF)
 - XlsxWriter (Excel)
 - kaleido (export des graphes)

5. Architecture

```
core/          logique métier
├─ simulator.py      simulation historique
├─ predictor.py      modèles prédictifs
└─ datasources/      extraction de données

api/           endpoints REST
├─ routers/         définition des routes
└─ schemas/         schémas Pydantic

client/        interface Streamlit
├─ main.py          point d'entrée UI
├─ plots_manager.py graphiques
├─ export_manager.py exports
└─ ui_manager.py    composants d'interface
```

Endpoints REST

GET /health	vérification de l'état de l'API
POST /api/simuler	simulation d'un portefeuille
POST /api/predire	prédiction et comparaison des stratégies

6. Métriques

Catégorie	Indicateur	Description
Performance	CAGR / Rendement total	Mesure de croissance et gain net
Risque	Volatilité annualisée	Dispersion des rendements
Rendement ajusté	Ratio de Sharpe	Sur-performance par unité de risque
Qualité prédiction	R ² / RMSE / MAE / σ résidus	Fiabilité des modèles

7. Points d'amélioration

- **Bug** : l'export de données masque parfois le dashboard ; relancer une simulation rétablit l'affichage.
 - **Métier** : élargir la liste d'actifs.
 - **Statistiques** : ajouter tests de normalité robustes, tests d'autocorrélation des résidus.
 - **Packaging** : conteneurisation Docker.
-

8. Exemple de Dockerfile minimal pour déploiement

```
FROM python:3.11-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000 8501
CMD sh -c "python -m app.api.main & \
    streamlit run app/client/main.py"
```

