Rapport Technique : Gestionnaire de Risque Contrepartie

1. Instructions de Configuration

Pour mener à bien ce projet, plusieurs outils et technologies ont été utilisés :

Prérequis

1. **Node.js** : Pour installer et gérer des dépendances.
2. **Python** : Utilisé pour le backend interactif avec Web3.py.
3. **Ganache** : Un environnement Ethereum local pour prendre l'adresses de conttrepartie .
4. **MetaMask** : Extension de navigateur pour gérer les portefeuilles Ethereum.
5. **Streamlit** : Pour construire l'interface utilisateur avec une interaction en temps réel.
6. **web3.py** : Bibliothèque Python pour interagir avec les contrats intelligents.

2. Architecture du projet

Composants principaux:

1. Contrat intelligent Solidity :
   - Gère les contreparties (ajout, mise à jour d'exposition, calcul de risques et désactivation).
   - Implémente des fonctions comme:
     - `ajouterContrepartie`,
     - `mettreAJourExposition`,
     - `calculerRisque`,
     - `calculerRatioCouverture`,
     - `desactiverContrepartie`.

2. Frontend Streamlit :
   - Permet à l'utilisateur de se connecter à un portefeuille, d'ajouter des contreparties, de mettre à jour leurs expositions et d'effectuer divers calculs de risque.

3. Interaction via Web3.py :
   - Connecte l'application frontend au contrat intelligent déployé sur le réseau blockchain.

Étapes de Configuration

1. **Déploiement du Contrat Intelligent** :
    - Utilisation de Remix IDE (https://remix.ethereum.org) pour écrire et compiler le contrat **`GestionnaireRisqueContrepartie`.**
    - Le contrat intelligent a été déployé via Remix avec MetaMask comme fournisseur injecté. (**Adresse du contrat** : 0x985BDdDA7E299dAffb2A372448B6dd9632e1B341)
    - Ajout de tokens à l'aide du Polygon faucet
    - Déploiement du contrat et récupération de son adresse et ABI pour les intégrer au frontend.
2. **Configuration de l'Interface Streamlit** :
    - Création d'un fichier Python contenant le code Streamlit.
    - Mise à jour de l'adresse du contrat dans la variable `contract_address`.
    - Lancement de l'application avec la commande : `streamlit run app.py`
    - Accès à l'application via http://localhost:8501.

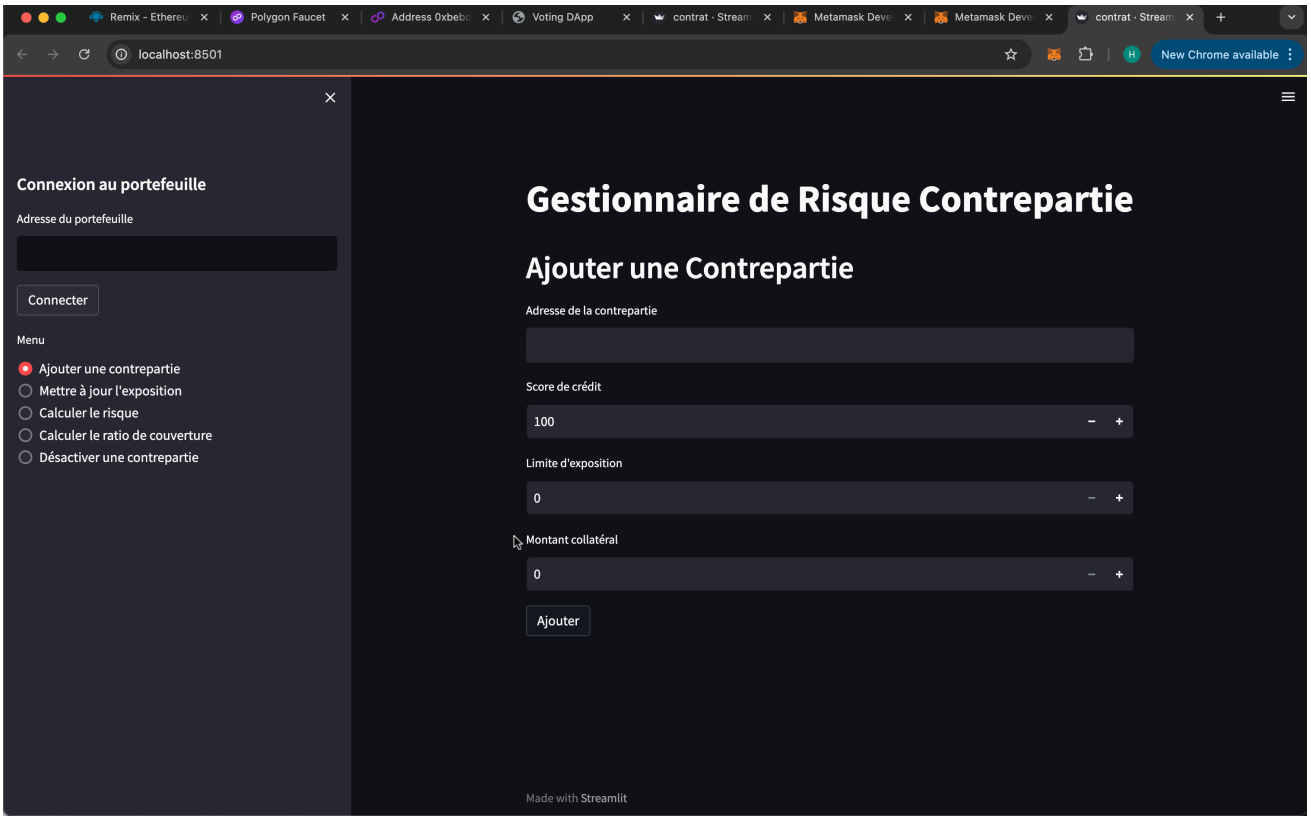
2. Spécifications des Fonctions

2.1 Ajouter une Contrepartie

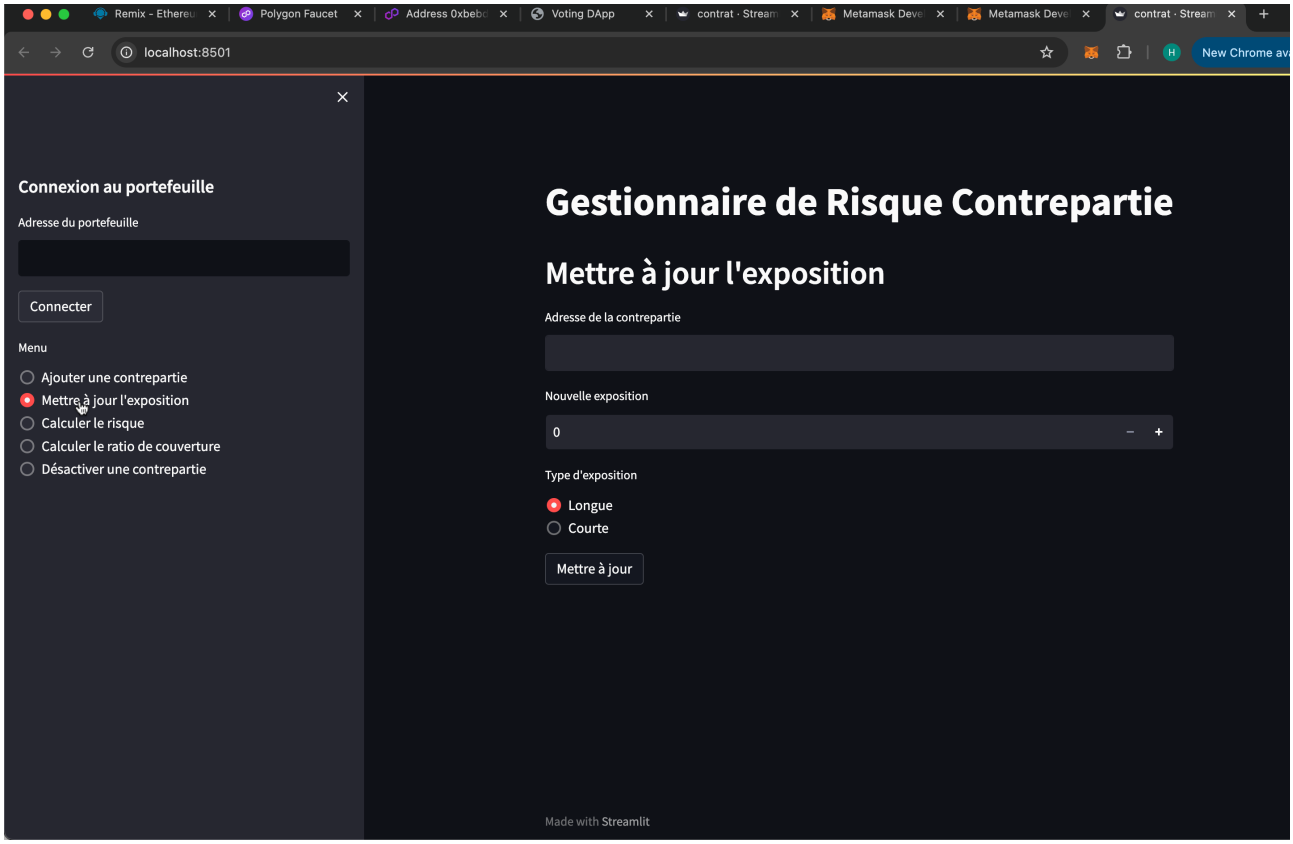
- **Entrées** :
    - Adresse de la contrepartie
    - Score de crédit
    - Limite d'exposition
    - Montant du collatéral
- **Résultat** : Crée une nouvelle contrepartie et génère un événement `ContrepartieAjoutee`.

2.2 Mettre à jour l'Exposition

- **Entrées** :
    - Adresse de la contrepartie
    - Nouvelle exposition (longue ou courte)

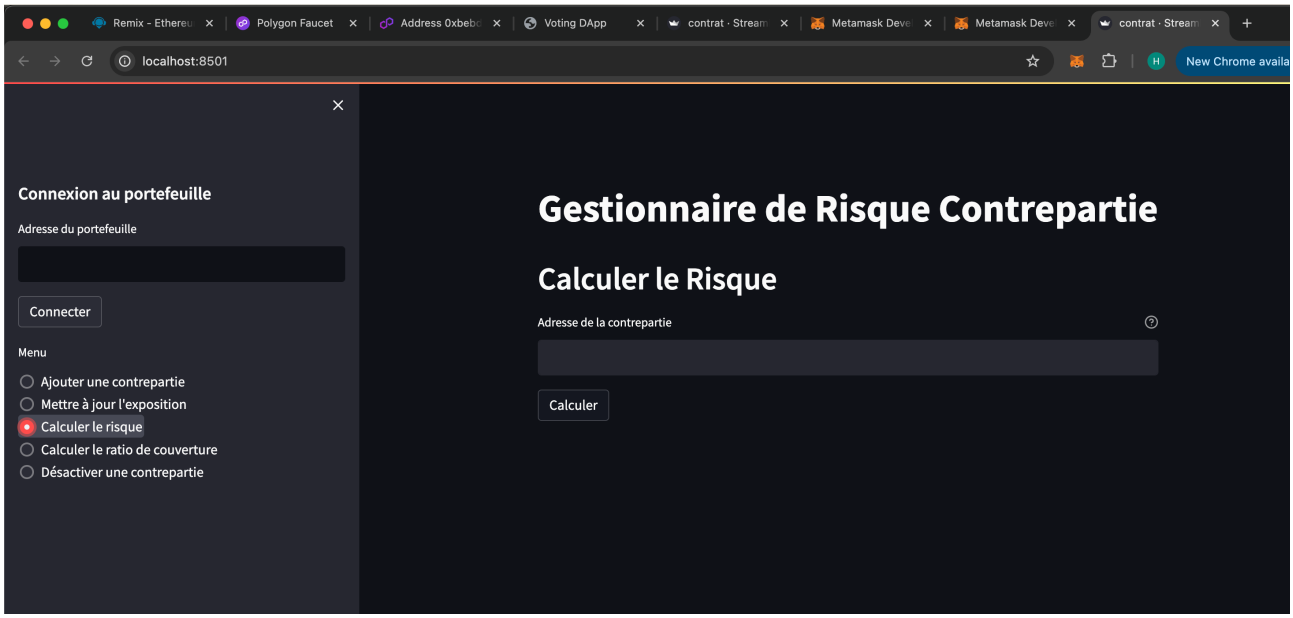


- **Résultat** : Met à jour l'exposition nette et vérifie les limites.
    - Génère un événement `ExpositionMiseAJour`.
    - Si la limite est dépassée, génère un événement `LimiteDepassee`.



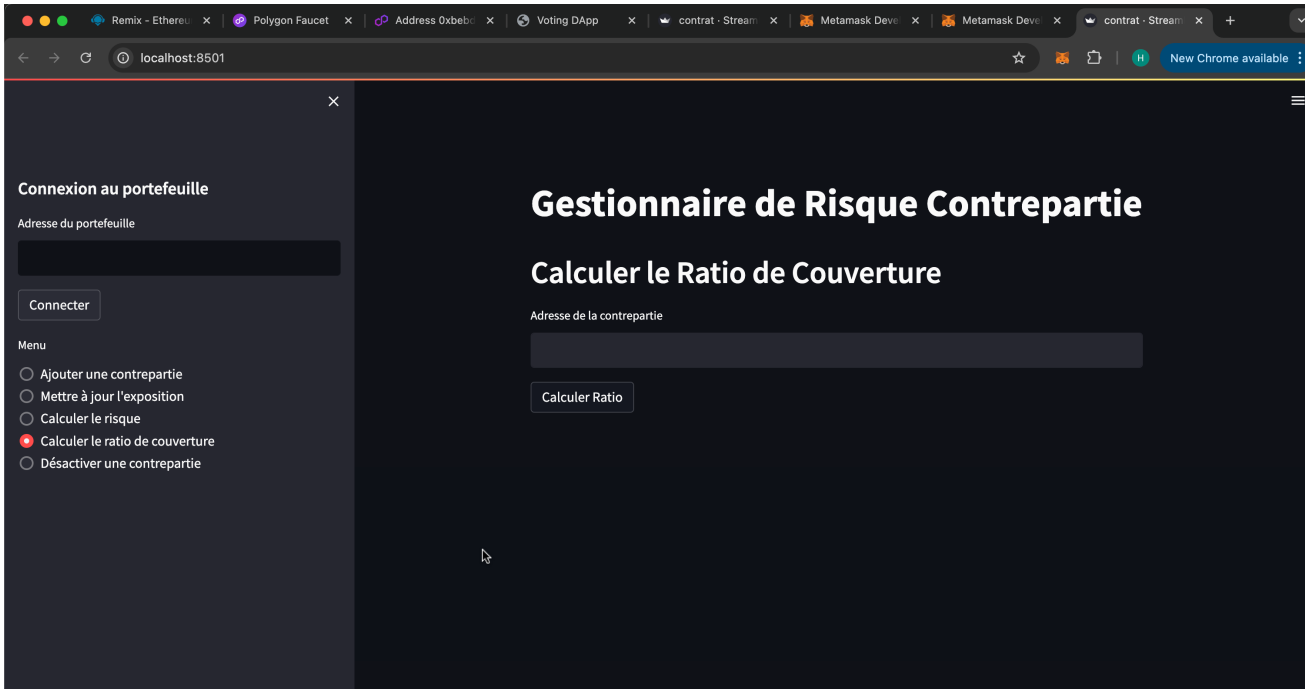
2.3 Calculer le Risque

- **Entrées** : Adresse de la contrepartie
- **Résultat** : Retourne le risque calculé en fonction de l'exposition courante, de la limite d'exposition, et du score de crédit.



2.4 Calculer le Ratio de Couverture

- **Entrées** : Adresse de la contrepartie
- **Résultat** : Retourne le ratio de couverture (collatéral / exposition totale).



2.5 Désactiver une Contrepartie

- **Entrées** : Adresse de la contrepartie
- **Résultat** : Marque la contrepartie comme inactive et empêche toute mise à jour.

3. Procédures de Test

3.1 Tests Unitaires

- **Objectif** : Vérifier le bon fonctionnement de chaque fonction du contrat intelligent.
- **Outils** : Ganache, Remix IDE.
- **Cas de Test** :
    - Ajouter une contrepartie valide.
    - Mise à jour d'une exposition longue ou courte.
    - Calcul du risque avec différents paramètres.
    - Calcul du ratio de couverture.
    - Désactivation et vérification de l'inaccessibilité.

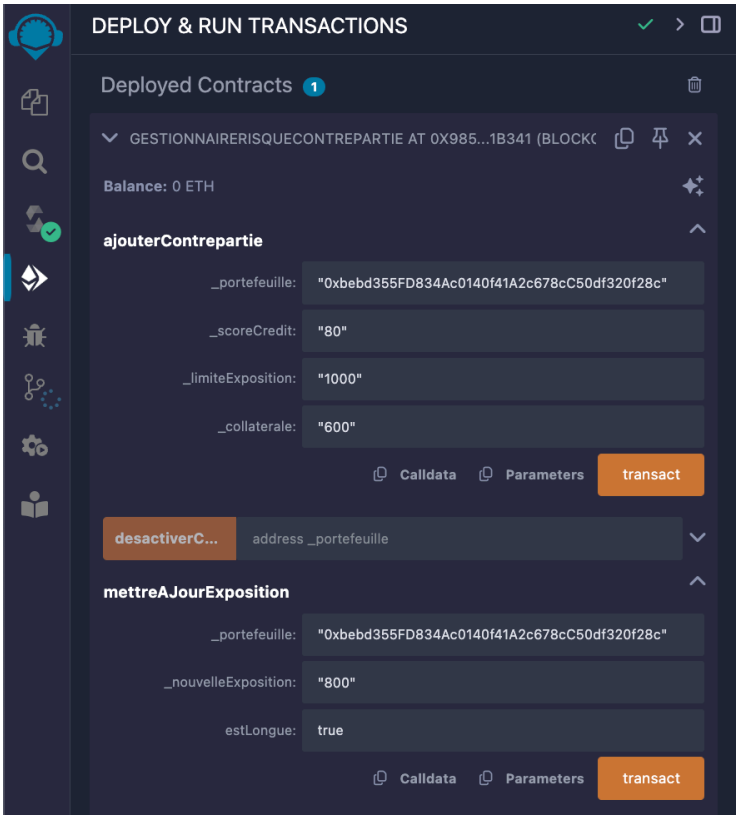
3.2 Tests d'Intégration

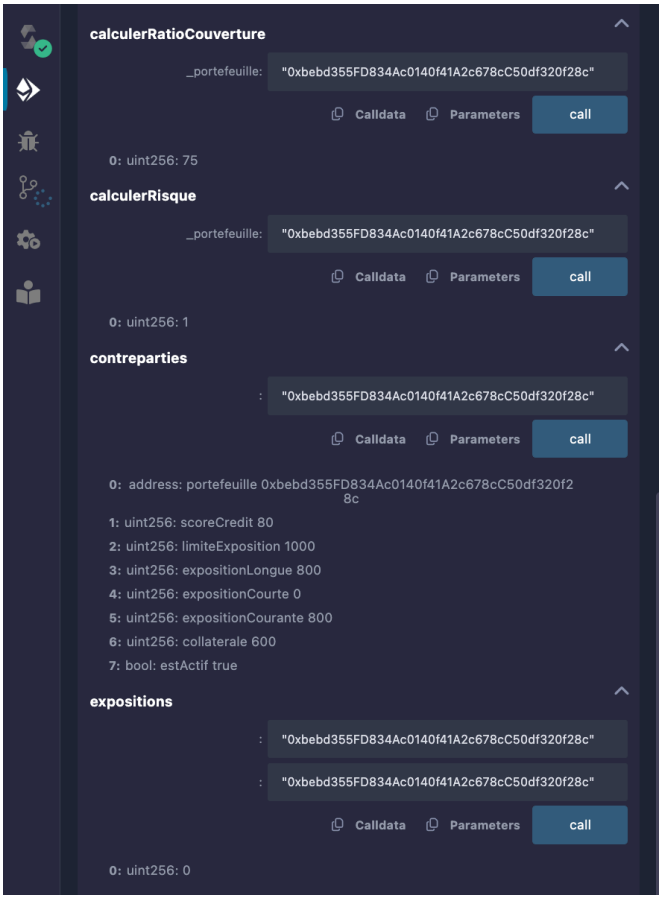
- **Objectif** : Vérifier la connexion entre le frontend et le contrat intelligent via Web3.
- **Cas de Test** :

- Connexion au portefeuille utilisateur.
- Envoi de transactions depuis l'interface Streamlit.
- Affichage des résultats calculés.

3.3 Validation Manuelle

- **Objectif** : S'assurer que les événements et les états sont corrects.
- **Procédure** :
  - Vérifiez les logs des événements dans Remix et Streamlit.
  - Confirmez les résultats dans MetaMask.





4. Considérations de Sécurité

1. **Validation des Entrées** :
   - Vérifiez que toutes les adresses sont valides.
   - Empêchez les dépassements et les sous-dépassements dans les calculs.

2. **Gestion des Exceptions** :
    - Capturez les erreurs pour éviter les échecs lors de la mise à jour de l'exposition ou de l'ajout d'une contrepartie.
3. **Réputation des Portefeuilles** :
    - Implémentez un système pour noter la fiabilité des portefeuilles.
4. **Protection contre les Attaques :**
    - Empêchez les appels non autorisés.
    - Limitez les échecs de gas en estimant correctement les limites.

Conclusion

Ce projet démontre la puissance de la blockchain dans la gestion des risques financiers. En automatisant les calculs et en renforçant la transparence des transactions, ce système est une base solide pour des solutions financières avancées. Les prochaines étapes pourraient inclure :

- L'intégration d'un système de notifications.
- L'amélioration des règles de sécurité pour les portefeuilles suspects.
- L'extension du projet pour inclure une gestion multi-actifs.

Avec une approche modulaire, ce projet peut être adapté à divers scénarios financiers nécessitant une gestion efficace des contreparties et des expositions.