Document de Spécification des Besoins pour l'Application de Gestion de Portefeuilles Financiers

YANG Chen

29/11/2023

1 Introduction

Ce document vise à définir et décrire les besoins de développement de l'application de gestion de portefeuilles financiers. L'application offrira des outils pour créer, gérer et analyser des portefeuilles financiers personnels ou multiples, couvrant les investissements en actions et cryptomonnaies.

2 Besoins Fonctionnels

2.1 Gestion de Portefeuille

— Création et gestion de portefeuille : Permettre aux utilisateurs de créer et de gérer plusieurs portefeuilles.
— Ajout/Suppression d'actifs : Soutenir l'ajout ou la suppression d'actions et de cryptomonnaies dans le portefeuille.

2.2 Suivi des Données en Temps Réel

— Suivi de la valeur en temps réel : Afficher en temps réel la valeur de chaque actif dans le portefeuille.
— Consultation des données historiques : Fournir une fonction pour consulter la valeur et la performance du portefeuille à des moments passés.

2.3 Intégration et Stockage des Données

— Intégration API : Obtenir des données en temps réel des marchés financiers via des API publiques.
— Importation de données : Permettre aux utilisateurs d'importer des données de plateformes de transactions externes (comme Coinbase ou Binance).
— Cache de données local : Pour améliorer la vitesse et l'efficacité d'accès, stocker les données localement.

2.4 Interface Utilisateur

— Interface graphique : Offrir une interface intuitive et facile à utiliser.
— Visualisation des données : Présenter les actifs et la valeur du portefeuille sous forme de graphiques.

3 Besoins Non Fonctionnels

3.1 Performance

— Réactivité rapide : Le temps de réponse de l'interface ne doit pas dépasser quelques secondes.
— Fréquence de mise à jour des données : La mise à jour des données financières ne devrait pas avoir plus d'une minute de retard.

3.2 Sécurité

— Cryptage des données : Les données sensibles doivent être cryptées lors du stockage local.
— Sécurité d'accès : Nécessiter un mot de passe utilisateur pour accéder à l'application.

4 Fonctionnalités Avancées (Bonus)

4.1 Fonctions d'Analyse

— Fournir des outils d'analyse de la performance des actifs et des portefeuilles.
— Encourager l'innovation et les fonctionnalités d'analyse personnalisées.

4.2 Suivi des Transactions

— Suivre les transactions importantes sur la blockchain, en particulier les transactions de grande valeur (connues sous le nom de "traque des baleines").

4.3 Choix de la Devise

— Permettre aux utilisateurs de choisir la devise de référence pour afficher la valeur des actifs (par exemple, EUR, USD, etc.).

5 Besoins de l'Interface Utilisateur

— Intuitivité : L'interface doit être intuitive et facile à comprendre, simplifiant l'opération pour l'utilisateur.
— Personnalisation : Offrir une certaine mesure de personnalisation de l'interface et des options fonctionnelles.

6 Contraintes Système

— Limitations techniques : L'application doit être compatible avec les systèmes d'exploitation et dispositifs principaux.
— Conformité légale : Respecter les lois et réglementations relatives aux données financières et à la vie privée.

7 Spécification Du Code

— Langage de programmation : Java
— Style de codage
  Suivez les conventions de code Java d'Oracle
  Utilisez une indentation appropriée (généralement quatre espaces)
— Structure du projet
  Suivez la structure standard des projets Maven
  Le code source est placé dans src/main/java et le code de test dans src/test/java
— Conventions de dénomination
  Les noms de classes et d'interfaces sont nommés en CamelCase
  Les noms de méthodes et de variables sont nommés en camelCase
  Les constantes sont séparées par des lettres majuscules et des traits de soulignement (par exemple, MAX_VALUE)
— Spécification des annotations
  Utilisez Javadoc pour annoter les classes, les méthodes et les champs publics
  Utilisez des commentaires de ligne pour les segments de code logiques complexes
— Gestion des erreurs
  Préférez la gestion des exceptions au renvoi de codes d'erreur
  Les exceptions personnalisées doivent être claires et significatives
  Évitez les blocs catch vides
— Meilleures pratiques en matière de performances
  Évitez de créer des objets inutiles dans les boucles
  Utilisez des structures de données appropriées
  Optimisez les interactions avec les bases de données et les appels au réseau

— Spécification des tests
    Écrivez des tests unitaires qui couvrent les principales fonctionnalités et les conditions limites
    Utilisez JUnit ou d'autres cadres de test
— Contrôle de version
    Gérez le code à l'aide d'un système de contrôle de version tel que Github
    Suivez un processus clair de gestion des branches et de demande de fusion

8 Annexes

— Glossaire
— Références

9 Révision et Approbation

Ce document doit être révisé et approuvé par l'équipe de projet et les parties prenantes concernées.