Spécification Technique pour le Contexte `TotalAmountContext`

1. Introduction

Le fichier `totalAmountContext.tsx` configure et fournit un contexte pour la gestion du montant total des transactions. Ce contexte permet aux composants React de lire et de mettre à jour le montant total des transactions de manière centralisée.

2. Objectifs

- Gérer le montant total des transactions dans l'application.
- Fournir un contexte global pour accéder et mettre à jour le montant total.
- Permettre aux composants React de réinitialiser le montant total.

3. Dépendances

- `react`

4. Interface

4.1. Contexte

- `TotalAmountContext`: Un contexte React créé pour stocker le montant total des transactions.

4.2. Hooks

- `useTotalAmount`: Un hook personnalisé pour accéder et manipuler le montant total depuis n'importe quel composant.

4.3. Fournisseur de Contexte

- `TotalAmountProvider`: Un fournisseur de contexte qui encapsule les composants enfants et leur fournit le montant total via le contexte `TotalAmountContext`.

5. Fonctionnalités

5.1. État du Montant Total

L'état `totalAmount` est initialisé à 0 et peut être mis à jour via la fonction `setTotalAmount`.

```
const [totalAmount, setTotalAmount] = useState(0);
```

5.2. Réinitialisation du Montant Total

Une fonction `resetTotalAmount` est définie pour réinitialiser le montant total à 0.

```
const resetTotalAmount = () => {
  setTotalAmount(0);
};
```

5.3. Création du Contexte

Un contexte React est créé pour stocker le montant total des transactions et les fonctions associées.

```
const TotalAmountContext = createContext<TotalAmountContextType |
undefined>(undefined);
```

5.4. Hook Personnalisé

Le hook `useTotalAmount` permet d'accéder au contexte `TotalAmountContext` et de s'assurer qu'il est utilisé dans un fournisseur de contexte valide.

```
export const useTotalAmount = () => {
  const context = useContext(TotalAmountContext);
  if (!context) {
    throw new Error('useTotalAmount must be used within a
TotalAmountProvider');
  }
  return context;
};
```

5.5. Fournisseur de Contexte

Le fournisseur `TotalAmountProvider` encapsule les composants enfants et leur fournit le montant total et les fonctions associées via le contexte `TotalAmountContext`.

```
export const TotalAmountProvider: React.FC<TotalAmountProviderProps>
= ({ children }) => {
  return (
    <TotalAmountContext.Provider value={{ totalAmount,
setTotalAmount, resetTotalAmount }}>
      {children}
    </TotalAmountContext.Provider>
  );
};
```

6. Structure du Composant

```
import React, { createContext, useContext, useState, ReactNode }
from 'react';

interface TotalAmountContextType {
  totalAmount: number;
  setTotalAmount: (amount: number) => void;
  resetTotalAmount: () => void;
}

const TotalAmountContext = createContext<TotalAmountContextType |
undefined>(undefined);

interface TotalAmountProviderProps {
  children: ReactNode;
}

export const TotalAmountProvider: React.FC<TotalAmountProviderProps>
= ({ children }) => {
  const [totalAmount, setTotalAmount] = useState(0);

  const resetTotalAmount = () => {
    setTotalAmount(0);
  };

  return (
    <TotalAmountContext.Provider value={{ totalAmount,
setTotalAmount, resetTotalAmount }}>
      {children}
    </TotalAmountContext.Provider>
  );
};

export const useTotalAmount = () => {
  const context = useContext(TotalAmountContext);
  if (!context) {
    throw new Error('useTotalAmount must be used within a
TotalAmountProvider');
  }
  return context;
};
```

7. Utilisation

Pour utiliser le contexte dans un composant, il faut encapsuler les composants dans `TotalAmountProvider` et utiliser le hook `useTotalAmount` pour accéder et manipuler le montant total.

Exemple d'Utilisation

```
// App.tsx
import React from 'react';
import { SocketProvider } from './context/socketContext';
import { TotalAmountProvider } from './context/totalAmountContext';
import MainApp from './MainApp';

export default () => (
  <SocketProvider>
    <TotalAmountProvider>
      <MainApp />
    </TotalAmountProvider>
  </SocketProvider>
);
```

```
// SomeComponent.tsx
import React from 'react';
import { useTotalAmount } from '../context/totalAmountContext';

const SomeComponent = () => {
  const { totalAmount, setTotalAmount, resetTotalAmount } =
useTotalAmount();

  return (
    <div>
      <p>Total Amount: {totalAmount}</p>
      <button onClick={() => setTotalAmount(totalAmount + 10)}>Add
10</button>
      <button onClick={resetTotalAmount}>Reset Amount</button>
    </div>
  );
};

export default SomeComponent;
```

8. Conclusion

Le fichier `totalAmountContext.tsx` fournit une solution simple et efficace pour gérer le montant total des transactions dans une application React Native. En encapsulant la logique de gestion du montant total dans un contexte, il permet aux composants de l'application de lire et de modifier facilement ce montant de manière centralisée et organisée.