JEGYZŐKÖNYV

Mobil programozási alapok

Féléves feladat

Bevásárlólista

Készítette: **Szalóczy Krisztián**

Neptunkód: **Y4O4X0**

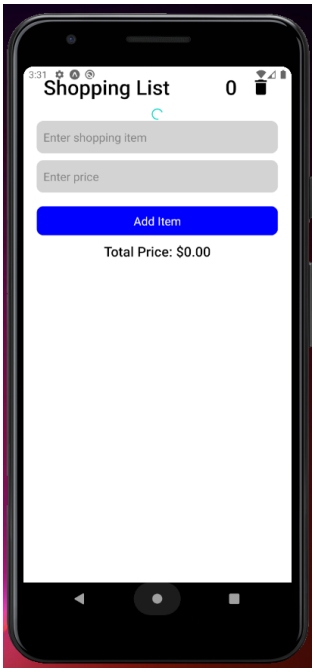
Dátum: 2024.12.09

Miskolc, 2024

Tartalomjegyzék

Bevezetés:

Ez a **React Native** alkalmazás egy dinamikus bevásárlólista-kezelő, amely lehetővé teszi a felhasználóknak, hogy tételeket adjanak hozzá, szerkesszenek, kipipáljanak, vagy töröljenek, igényeik szerint. Minden tételhez tartozik egy név, ár, valamint egy jelölőnégyzet, amely jelzi, hogy a tétel be van-e szerezve. Az alkalmazás a **Firebase Firestore** segítségével valós időben kezeli az adatokat. A felület segítségével a felhasználók követhetik a kiadásaikat is, mivel a rendszer összegzi a be nem szerzett tételek árát. Az alkalmazás felépítése egyszerű és felhasználóbarát, ezáltal könnyen kezelhető.



Üres lista 1

1. Főbb Funkciók

1. **Tételek hozzáadása**: A felhasználók megadhatják a tétel nevét és árát, majd hozzáadhatják a bevásárlólistához.
2. **Tételek megjelenítése**: A hozzáadott tételek megjelennek a listában, amely frissül minden változás után.
3. **Tételek szerkesztése**: A listában lévő tételeket szerkeszteni lehet (név és ár módosítása).
4. **Tételek kipipálása**: Minden tétel rendelkezik egy jelölőnégyzettel, amellyel jelezhető, hogy a tétel már be lett szerezve.

5. **Tételek törlése**: Egy tétel törölhető a listáról, illetve az összes tétel törölhető egyszerre.

6. **Összes ár kiszámítása**: A rendszer összegzi a még be nem szerzett tételek árát.

1. App.js

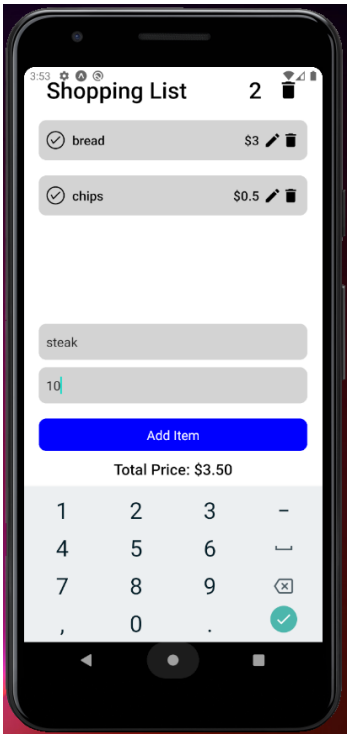
Ez a fájl tartalmazza a fő funkciókat és a felhasználói felület nagy részét.

Főbb állapotok (useState Hook-kal):

- **title**: Az új tétel nevét tárolja.
- **price**: Az új tétel árát tárolja.
- **shoppingList**: A bevásárlólista tételeinek tömbje.

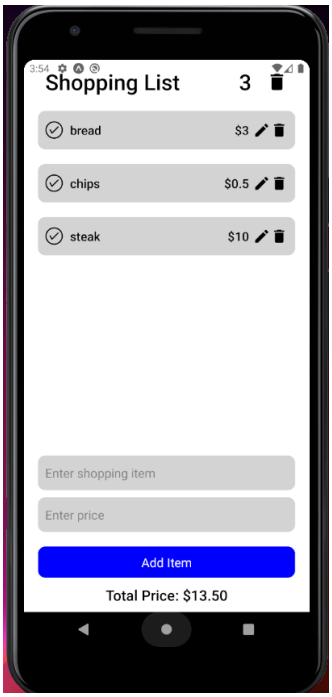
Fő Funkciók:

- **addShoppingItem**:
  - Létrehozza az új tételt a Firebase Firestore adatbázisban, majd hozzáadja azt a **shoppingList**-hez.
  - Sikeres hozzáadás után kiüríti az input mezőket, és frissíti a bevásárlólistát.



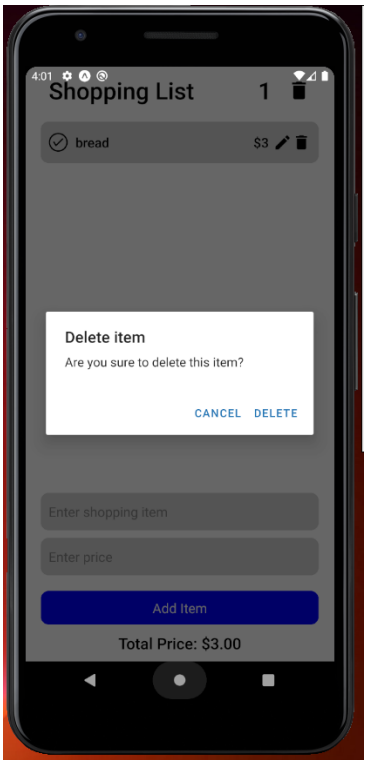
Itemek hozzáadás 2

- **getShoppingList**:
  - Lekérdezi az összes tételt a Firestore-ból, és beállítja a **shoppingList** állapotot az aktuális adatok alapján.



Itemek lekérése 3

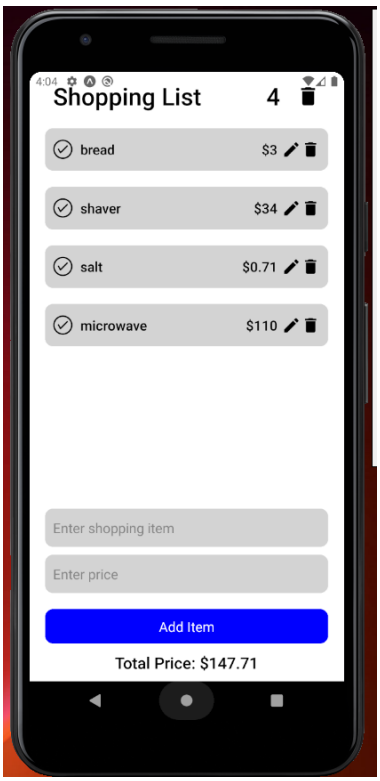
- **deleteShoppingList**:
  - Törli az összes tételt a Firestore adatbázisból, majd frissíti a bevásárlólistát.



Item törlése 4

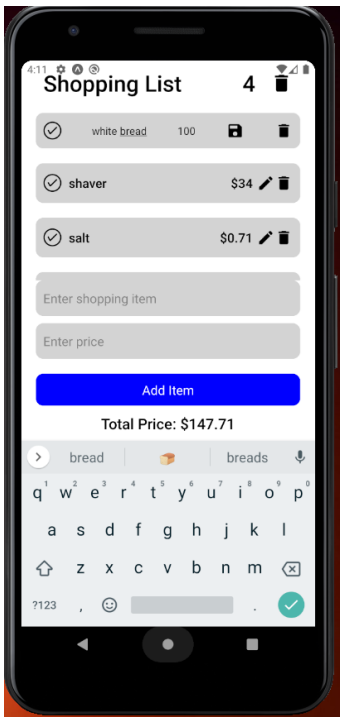
- **calculateTotalPrice**:

- Összegzi a még nem kipipált tételek árát, és megjeleníti a felületen.



Árak összegzése 5

- **Itemek módosítása:**
  - A listában levő elemek nevét és árát tudjuk módosítani



Főbb Felületi Elemek:

- **Header**: A bevásárlólista címét és a tételek számát jeleníti meg.
- **FlatList**: A shoppingList-et jeleníti meg, mely minden egyes tételt külön ShoppingItem komponensként mutat be.
- **TextInput**: Lehetővé teszi az új tétel nevének és árának bevitelét.
- **Pressable**: A felhasználó ezzel gombokat hozhat létre, például a tétel hozzáadására és a lista törlésére.

2. ShoppingItem.js

Ez a fájl tartalmazza az egyes tételek kezeléséhez szükséges funkciókat és megjelenítést.

Állapotok (useState Hook-kal):

1 **isChecked**: Jelzi, hogy a tétel be van-e szerezve.
2 **isEditing**: Szerkesztési mód be- és kikapcsolása.
3 **title** és **price**: Az aktuális tétel neve és ára.

Fő Funkciók:

- **updateIsChecked**:
  - Frissíti a isChecked állapotot a Firestore adatbázisban, jelezve, hogy a tétel be van-e szerezve.
- **deleteShoppingItem**:
  - Törli az aktuális tételt a Firestore-ból, majd frissíti a bevásárlólistát.
- **updateTitle**:
  - Frissíti az aktuális tétel nevét és árát a Firestore adatbázisban, majd kilép a szerkesztési módból, és frissíti a listát.

Felületi Elemek:

- **Pressable**: Az ellenőrző gomb (pipálás), szerkesztés, mentés és törlés gombjaihoz használt elem.
- **TextInput**: Szerkesztési módban megjelenik, így lehetőséget biztosít a tétel nevének és árának módosítására.

-

3. App.js

Állapotkezelés:

title és **price** állapotok a bevásárlólista elem nevéhez és árhoz.

shoppingList állapot a teljes lista elemeinek tárolására.

```
export default function App() {
const [title, setTitle] = useState("");
const [price, setPrice] = useState("");
const [shoppingList, setShoppingList] = useState([]);
```

Függvények:

addShoppingItem: Egy új bevásárló tételt ad hozzá a Firebase adatbázishoz (gyűjtemény neve: shopping). Sikeres mentés után kiüríti az input mezőket, és frissíti a listát.

getShoppingList: Lekéri a bevásárlólista elemeit az adatbázisból, és frissíti a shoppingList állapotot.

deleteShoppingList: Törli a bevásárlólista összes elemét az adatbázisból, majd újra lekéri a listát.

calculateTotalPrice: Számolja a bevásárlólista elemek összárát, csak azokat számítva, amelyek nincsenek kipipálva (**isChecked**).

```
const addShoppingItem = async () => {
    try {
        const docRef = await addDoc(collection(db, "shopping"), {
          title: title,
          price: parseFloat(price),
          isChecked: false,
        });
        console.log("Item added successfully: ", docRef.id);
       setTitle("");
       setPrice("");
       getShoppingList();
    } catch (e) {
      console.error("Error adding item: ",e);
    }
  }

  const getShoppingList = async () => {
    const querySnapshot = await getDocs(collection(db, "shopping"));

    setShoppingList(
```

```
      querySnapshot.docs.map((doc) => ({
        ...doc.data(),
        id: doc.id,
      }))
    )
  };

  const deleteShoppingList = async () => {
    const querySnapshot = await getDocs(collection(db, "shopping"));
    querySnapshot.docs.map((item) => deleteDoc(doc(db, "shopping",
item.id)))
    getShoppingList();
  }

  const calculateTotalPrice = () => {
    return shoppingList.reduce((total, item) => total +
(!item.isChecked ? item.price : 0), 0).toFixed(2);
  }
```

Felépítés és renderelés (return):

- **Fejléc (header)**: Mutatja a vásárlólista címét és a tételek számát. Tartalmaz egy törlő gombot is (delete ikon), ami törli az összes elemet.
- **Lista megjelenítése**: Ha vannak tételek, egy FlatList jeleníti meg őket, ShoppingItem komponenseken keresztül.
- **Input mezők**: Két TextInput mező a cím és ár megadására. Az ár mező numerikus.
- **Hozzáadás gomb**: Pressable gomb az új tétel hozzáadásához.
- **Összeg kijelzése**: A tételek összárát mutatja alul.

```
return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <Text style={styles.heading}>Shopping List</Text>
        <Text style={styles.noOfItems}>{shoppingList.length}</Text>

        <Pressable onPress={deleteShoppingList}>
          <MaterialIcons style={styles.bin} name="delete" size={30}
color="black"></MaterialIcons>
        </Pressable>
      </View>

      { shoppingList.length > 0 ? (
      <FlatList
      data={shoppingList}
      renderItem={({item}) => (
      <ShoppingItem
        title={item.title}
        price={item.price}
        isChecked={item.isChecked}
        id={item.id}
```

```
          getShoppingList={getShoppingList}
          />
        )}
        keyExtractor={item=>item.id}
        />
        ) : (
        <ActivityIndicator/>
        )}

        <TextInput
        placeholder="Enter shopping item"
        style={styles.input}
        value={title}
        onChangeText={(text) => setTitle(text)}
        onSubmitEditing={addShoppingItem}
        />
         <TextInput
          placeholder="Enter price"
          style={styles.input}
          value={price}
          onChangeText={(text) => setPrice(text)}
          keyboardType="numeric" // Csak számok beviteléhez
        />

        <Pressable onPress={addShoppingItem} style={styles.addButton}>
          <Text style={styles.addButtonText}>Add Item</Text>
        </Pressable>

        <Text style={styles.totalPrice}>Total Price:
${calculateTotalPrice()}</Text>

    </SafeAreaView>
  );
}
```