Projektbericht: Footprint Hero - CO2-Berechnungs-App

Ihr Name

August 28, 2023

1 Einleitung

Die **Footprint Hero** App ist eine mobile Anwendung, die Benutzern hilft, bewusstere Entscheidungen zu treffen, indem sie ihre CO2-Emissionen erfassen, berechnen und verfolgen können. Footprint Hero soll den Nutzern ein generelles Bewusstsein für ihren CO2-Fußabdruck geben, indem sie über eine grafische Oberfläche einen Überblick über ihre Emissionen mit verschiedenen Fahrzeugen über verschiedene Zeiträume erhalten.

2 Funktionen der App

Die **Footprint Hero** App bietet eine Reihe von Funktionen, die den Benutzern dabei helfen, bewusstere Entscheidungen in Bezug auf ihre CO2-Emissionen im Verkehrssektor zu treffen. Nachfolgend werden die Funktionen im Detail erläutert:

1. **Verkehrsmittel- und Dauerwahl**: Benutzer können aus einer Liste von Verkehrsmitteln auswählen, darunter Auto, Bahn, Bus, Fahrrad, Taxi und Flugzeug. Diese Auswahl erfolgt über den MainScreen, der durch das `MainViewModel` gesteuert wird. Hierbei wird die ausgewählte Option des Benutzers erfasst und an das `CO2CalculationViewModel` weitergegeben. Beispielhaft dazu der Codeauszug:

```
fun onVehicleSelected(vehicle: String) {
    selectedVehicle.value = vehicle
    co2CalculationViewModel.onVehicleSelected(vehicle)
}
```

2. **CO2-Berechnung**: Die App verwendet vordefinierte Emissionswerte für verschiedene Verkehrsmittel, die im `CO2CalculationViewModel` definiert sind. Nach der Auswahl eines Verkehrsmittels und der Eingabe der Dauer berechnet die App die geschätzten CO2-Emissionen. Diese Berechnung

erfolgt durch die Methode `calculateCO2` im `CO2CalculationViewModel`. Hier ist ein Beispiel, wie die CO2-Berechnung implementiert ist:

```
fun calculateCO2() {
    val co2Emission = model.transportationCO2[model.
        ↪ selectedTransportation] ?: 0f
    val calculatedCo2 = (co2Emission * model.duration) /
        ↪ 60f
    if (!calculatedCo2.isNaN()) {
        model.co2 = calculatedCo2
    }
    Log.d("CO2CalculationModel", "Calculated␣CO2:␣
        ↪ $calculatedCo2")
}
```

ToDo Quellen zu den Werten

3. **Login**: Der LoginScreen ist der Startpunkt der App. Hier können Benutzer ihre Anmeldeinformationen eingeben, um Zugriff auf die Funktionalitäten der App zu erhalten. Der Login-Bildschirm stellt sicher, dass die persönlichen Daten und CO2-Berechnungen eines Benutzers privat und sicher bleiben.

   xxxx:

```
\textcolor{red}{ToDo}
```

4. **Datenverfolgung**: Die berechneten CO2-Daten werden im `MainViewModel` gespeichert und verwaltet. Die Methode `merchList` aktualisiert die Liste der CO2-Daten und speichert sie in der Firebase-Datenbank. Diese Daten können dann auf dem MainScreen angezeigt werden, um den Benutzern einen Überblick über ihre CO2-Emissionen im Laufe der Zeit zu geben. Hier ist ein Beispiel für die Aktualisierung der CO2-Datenliste:

```
private fun merchList(co2: Float) {
    // ...
    val consumptionData = ConsumptionData(
        abbreviatedDayOfWeek, co2, getCurrentCalendarWeek
            ↪ ()
    )
    val updatedList = _co2DataList.value.co2Data.
        ↪ toMutableList()
    updatedList.add(consumptionData)
    _co2DataList.value = ConsumptionDataList(updatedList)
    writeCO2Data(_co2DataList.value)
    // ...
}
```

5. **Bewegungserkennung**: Die App erfasst Bewegungsdaten mithilfe des Beschleunigungssensors des Geräts. Dies erfolgt durch den `MotionDetectionService`. Wenn eine Bewegungsdauer von 30 Minuten erreicht wird, zeigt die App eine Benachrichtigung an, die den Benutzer dazu auffordert, die App zu verwenden. Hier ist die Methode zur Berechnung der Bewegungsdauer:

```
private fun calculateMotionDuration(x: Float, y: Float, z
    ↪ : Float): Int {
    val acceleration = sqrt(x * x + y * y + z * z)
    val motionDurationMillis = System.currentTimeMillis()
        ↪  - motionStartTimeMillis
    return if (acceleration > 0.1f) {
        (motionDurationMillis / (1000 * 60)).toInt()
    } else {
        0
    }
}
```

3 Architektur: Model-View-ViewModel (MVVM)

Die Footprint Hero App folgt dem MVVM-Architekturmuster, das eine klare Trennung von Daten, Benutzeroberfläche und Geschäftslogik ermöglicht.

3.1 Modell (Model)

Im Modellbereich werden die Datenstrukturen definiert, die zur Speicherung von Informationen verwendet werden. Die `ConsumptionData`-Klasse speichert CO2-Daten wie den Wochentag, die CO2-Emissionen und die Kalenderwoche. Die `User`-Klasse enthält Informationen ToDo.

4 Sicht (View)

Die Benutzeroberflächenelemente und die Logik zur Anzeige von Daten werden im View-Bereich erstellt. Die Benutzerinteraktion erfolgt hier, indem sie Verkehrsmittel und Dauer auswählen. Die berechneten CO2-Daten werden angezeigt.

4.1 MainScreen

Hier können Benutzer ihre Auswahl des Verkehrsmittels und der Dauer treffen, um die CO2-Berechnungen durchzuführen. Zusätzlich bietet der MainScreen eine Übersicht über die bisherigen CO2-Emissionen für den ausgewählten Zeitraum. Die Benutzer können auch auf die Einstellungen zugreifen, um benutzerdefinierte CO2-Emissionswerte hinzuzufügen.

4.2 LoginScreen

Im LoginScreen können Benutzer ihre Anmeldeinformationen eingeben, um auf den MainScreen weitergeleitet zu werden. Der LoginScreen stellt sicher, dass die persönlichen Daten¡ eines Benutzers privat und sicher bleiben.

4.3 ViewModel

ToDo Die anderen ViewModels?
Die ViewModel-Klassen `MainViewModel` und `CO2CalculationViewModel` sind für die Funktionalität der App verantwortlich. Sie verarbeiten die Auswahl von Verkehrsmitteln und Dauer, führen die CO2-Berechnung durch und aktualisieren die CO2-Datenliste.

4.4 Datenbank und Benachrichtigungen

Die `FirestoreDatabase`-Klasse behandelt die Kommunikation mit der Firebase Firestore-Datenbank. Sie bietet Methoden zum Schreiben, Lesen, Aktualisieren und Löschen von CO2-Daten.

5 Bewegungserfassung durch den Bewegungssensor

Die erfassten Bewegungsdaten werden in Echtzeit abgefragt und auf eine Bewegungsdauer von 30 Minuten überprüft. Daraufhin wird eine entsprechende Notification einmalig ausgelöst.

6 Notification

Die Benachrichtigung erfolgt durch die Abfrage des Bewegungssensors und wird erst nach 30 Minuten ausgelöst, wenn der Benutzer stehen bleibt. In diesem Fall startet die dreißigminütige Zählung von Neuem. Die `NotificationHelper`-Klasse vereinfacht das Erstellen und Anzeigen von Benachrichtigungen, um Benutzer zur Verwendung der App zu motivieren.
ToDos: Code?

7 Observer

Die Absorber befinden sich im MainViewModel. Sie erfassen sämtliche Veränderungen in der Liste der CO2-Werte und informieren entsprechenden Views. Hier die ensprechende Codestelle mit dem StateFlow:

```
private val _co2DataList = MutableStateFlow(ConsumptionDataList
    ↪ (mutableListOf()))
```

```
val co2DataList: StateFlow<ConsumptionDataList> get() =
    ↪ _co2DataList
```

8 Erweiterung der Funktionen

ToDos:

- Coroutines / ASYNCHRONISM / Permission
- Weitere Einzelheiten zur Implementierung der CO2-Berechnung und -Verfolgung.:

9 Nicht umgesetze Ideen / Überlegungen

Nutzerfeedback und Verbesserungsvorschläge für zukünftige Versionen der App.

Anpassungsmöglichkeiten für den Benutzer, wie die Möglichkeit, benutzerdefinierte CO2-Emissionswerte hinzuzufügen.:

1. **Schritt 1:** Der Benutzer navigiert zur Einstellungsseite der App, wo die Option zur Anpassung der CO2-Emissionswerte verfügbar ist.
2. **Schritt 2:** Die App listet die verschiedenen Verkehrsmittel auf, für die der Benutzer Emissionswerte definieren kann.
3. **Schritt 3:** Der Benutzer wählt ein Verkehrsmittel aus, für das er den Emissionswert anpassen möchte.
4. **Schritt 4:** Die App präsentiert dem Benutzer ein Eingabefeld, in dem er den benutzerdefinierten Emissionswert eingeben kann. Zusätzlich kann er eine kurze Beschreibung oder eine Begründung für den gewählten Wert angeben.
5. **Schritt 5:** Nachdem der Benutzer den neuen Wert eingegeben hat, bestätigt er die Änderung, und die App speichert den benutzerdefinierten Emissionswert für zukünftige Berechnungen.

10 Herausforderungen / Probleme

Nutzerfeedback und Verbesserungsvorschläge für zukünftige Versionen der App.

11 Zusammenfassung

Die Footprint Hero App stellt eine innovative Möglichkeit dar, Benutzern bewusstere Entscheidungen im Verkehrsbereich zu ermöglichen. Mit einer klaren

Architektur, die das MVVM-Muster nutzt, bietet die App eine benutzerfreundliche Benutzeroberfläche, die einfache CO2-Berechnung und -Verfolgung ermöglicht. Die Integration von Sensordaten zur Bewegungserkennung und die Verwendung von Benachrichtigungen tragen zur Steigerung der Benutzereinbindung bei. Durch die Förderung nachhaltigerer Verkehrsgewohnheiten kann die App einen positiven Beitrag zur Umwelt leisten.