I'm not implementing Clean Architecture in my Flutter app due to time constraints. Instead, I'm utilizing shared preferences for data storage. Following that, I'm retrieving values for parameters such as electricity, diesel, petrol, LPG, and others, and setting default values for them.

Screens

- addProductScreen
- detailProductScreen
- editProductScreen
- homepage

calculateCarbonFootprint Function Documentation

Overview

The calculateCarbonFootprint function calculates the carbon footprint based on the fuel type and the total weight of the emissions. It considers different emission factors for various fuel types and multiplies the total weight by the corresponding emission factor to obtain the carbon footprint.

Parameters

- fuelType (String): Specifies the type of fuel used, including options such as electricity, petrol, diesel, or LPG (liquefied petroleum gas).
- totalWeight (double): Represents the total weight of emissions produced.

Return Value

- double: Returns the calculated carbon footprint based on the provided fuel type and total weight of emissions.

Implementation Details

Variable Declaration: Declares a variable emissionFactor to store the emission factor corresponding to the provided fuel type.

Switch Statement: Determines the emission factor based on the provided fuelType using a switch statement. Different cases are provided for recognized fuel types (electricity, petrol, diesel, and LPG), with corresponding emission factors. If the provided fuelType is not recognized, a default emission factor of 1.5 is used.

Carbon Footprint Calculation: Calculates the carbon footprint by multiplying the totalWeight with the determined emissionFactor.

Return Statement: Returns the calculated carbon footprint.

Example Usage:

double carbonFootprint = calculateCarbonFootprint('petrol', 100.0);
print('Carbon Footprint: $carbonFootprint');

This will output the carbon footprint calculated based on petrol usage and a total weight of emissions equal to 100.0.

ProductBloc Documentation

Overview

The ProductBloc class manages the state of product-related operations in the application using the BLoC (Business Logic Component) architecture. It handles events such as loading products, adding, editing, deleting, and searching for products. This class integrates with SharedPreferences for data persistence.

Constructor

- ProductBloc(): Constructs a ProductBloc instance with the initial state set to ProductsLoading.

Properties

- products: A list of Product objects representing the current state of products managed by the bloc.

Methods

mapEventToState(ProductEvent event):
- Maps incoming events to corresponding states.
- Handles events such as loading products, adding, editing, deleting, and searching for products.

_mapLoadProductsToState():
- Loads products from SharedPreferences.
- Emits ProductsLoading state initially, then updates to ProductsLoaded state with loaded products.

_mapSearchProductsToState(String query):
- Searches for products based on the provided query.
- Emits ProductsLoaded state with search results.

_loadProductsFromPrefs():
- Loads products from SharedPreferences.
- Returns a list of loaded products.

_saveProductsToPrefs(List<Product> products):
- Saves products to SharedPreferences.
- Converts products to JSON and stores them.

_addProduct(Product product):
- Adds a new product to the list.
- Persists the updated list to SharedPreferences.
- Triggers a reload of products by emitting a LoadProducts event.

_editProduct(Product product):

- Edits an existing product.
- Updates the product in the list.
- Persists the updated list to SharedPreferences.
- Triggers a reload of products by emitting a LoadProducts event.

_deleteProduct(Product product):

- Deletes a product from the list.
- Persists the updated list to SharedPreferences.
- Triggers a reload of products by emitting a LoadProducts event.

Dependencies

- dart:convert: Provides encoding and decoding functionalities for JSON.
- package:bloc: Offers the BLoC pattern implementation.
- package:flutter_bloc: Provides additional functionalities for BLoC integration with Flutter.
- package:shared_preferences: Enables data persistence using SharedPreferences.