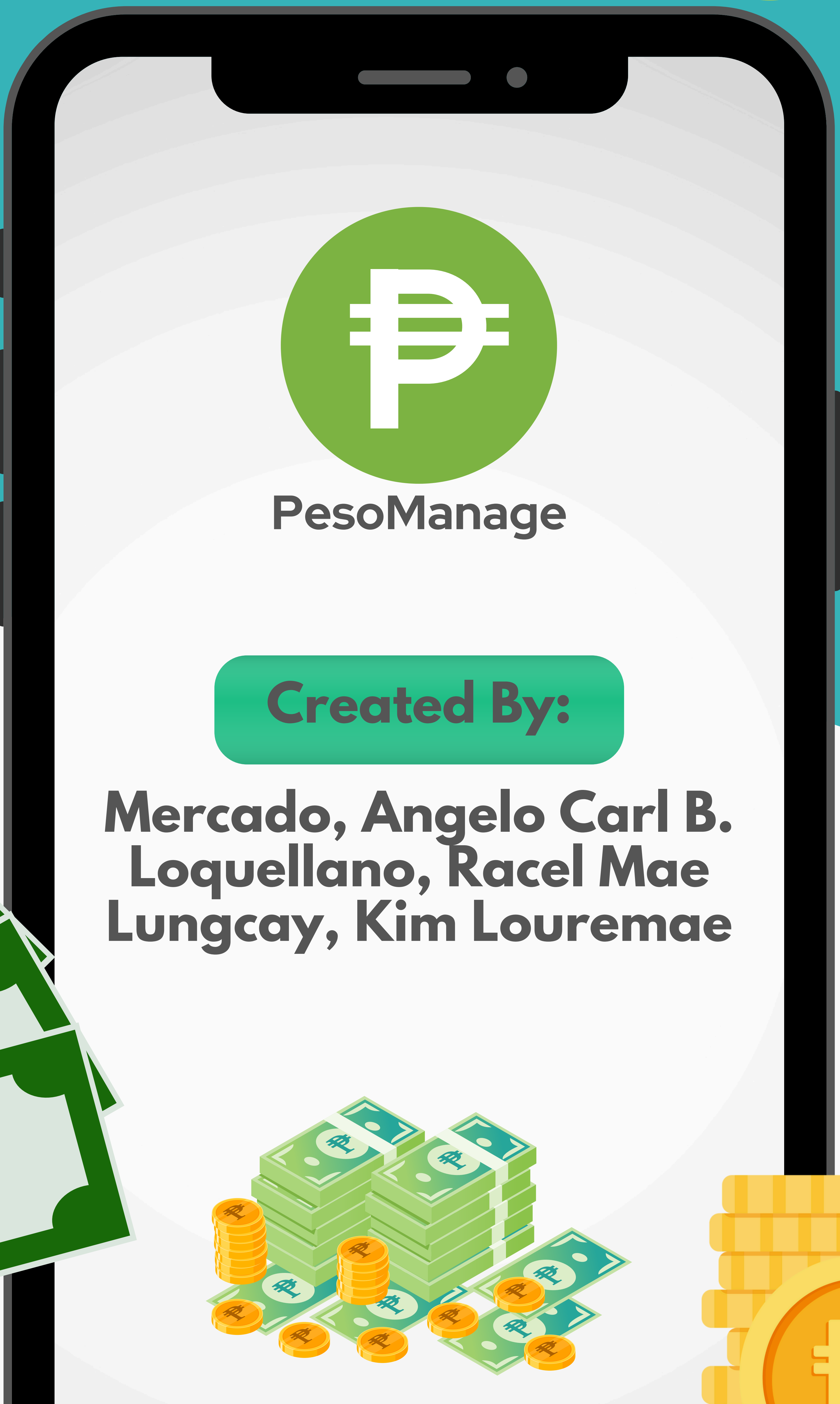
PesoManage

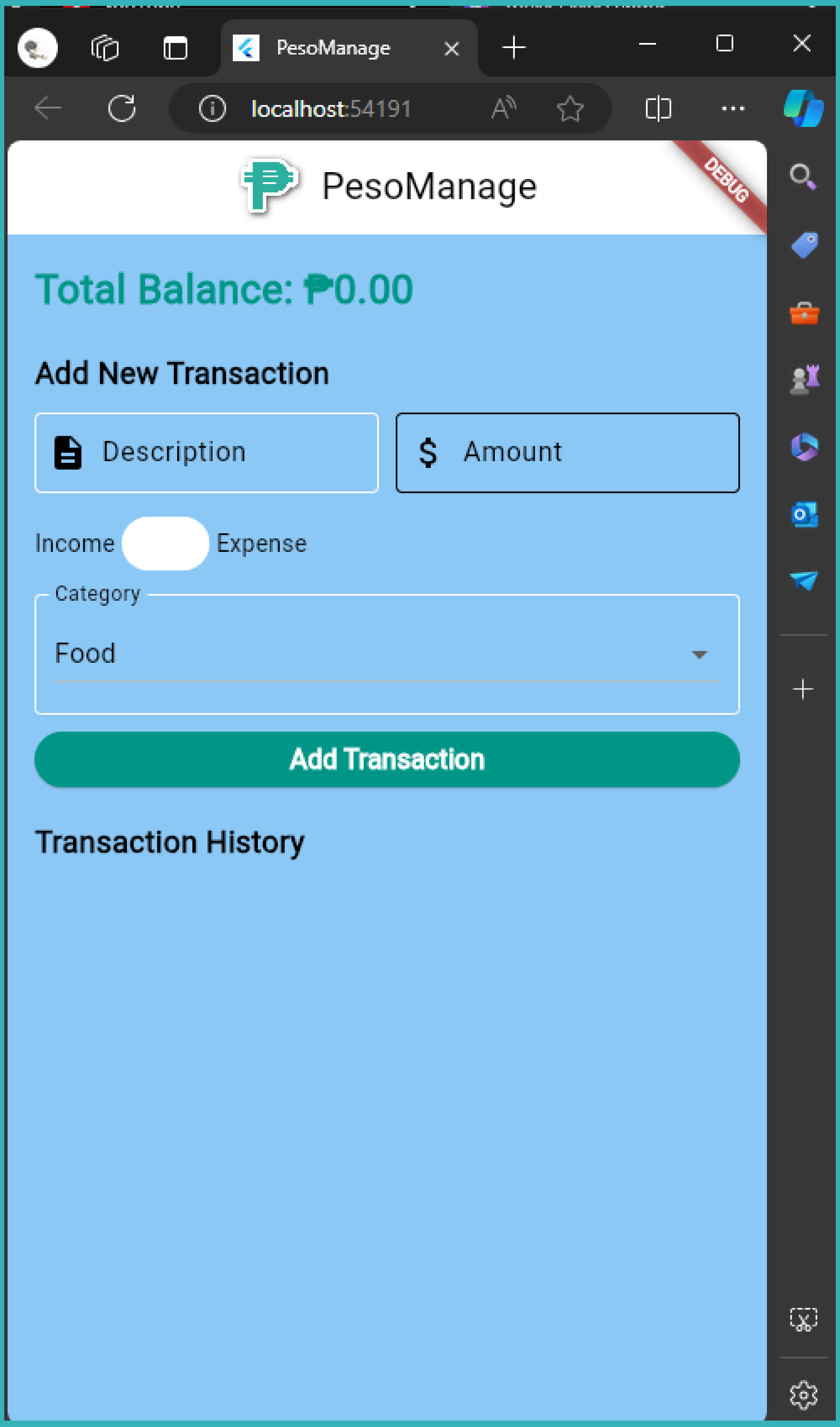
Your user-friendly budget tracking app



Created By:

Mercado, Angelo Carl B.
Loquellano, Racel Mae
Lungcay, Kim Louremae

SCREENSHOTS OF MOBILE APPLICATION



PesoManage

localhost:54191

DEBUG

Total Balance: ₱0.00

Add New Transaction

Description

Amount

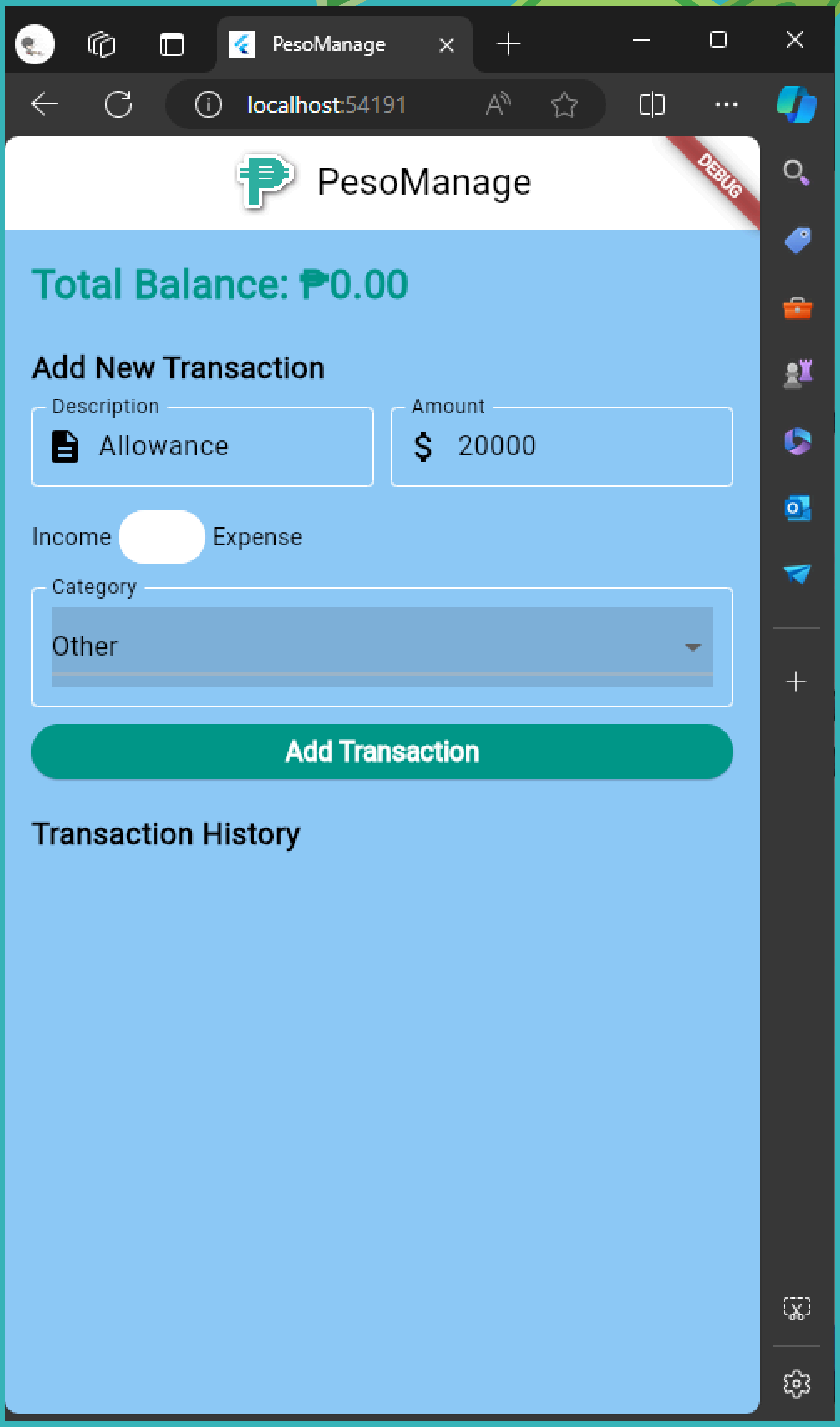
Income Expense

Category

Food

Add Transaction

Transaction History



PesoManage

localhost:54191

DEBUG

Total Balance: ₱0.00

Add New Transaction

Description: Allowance

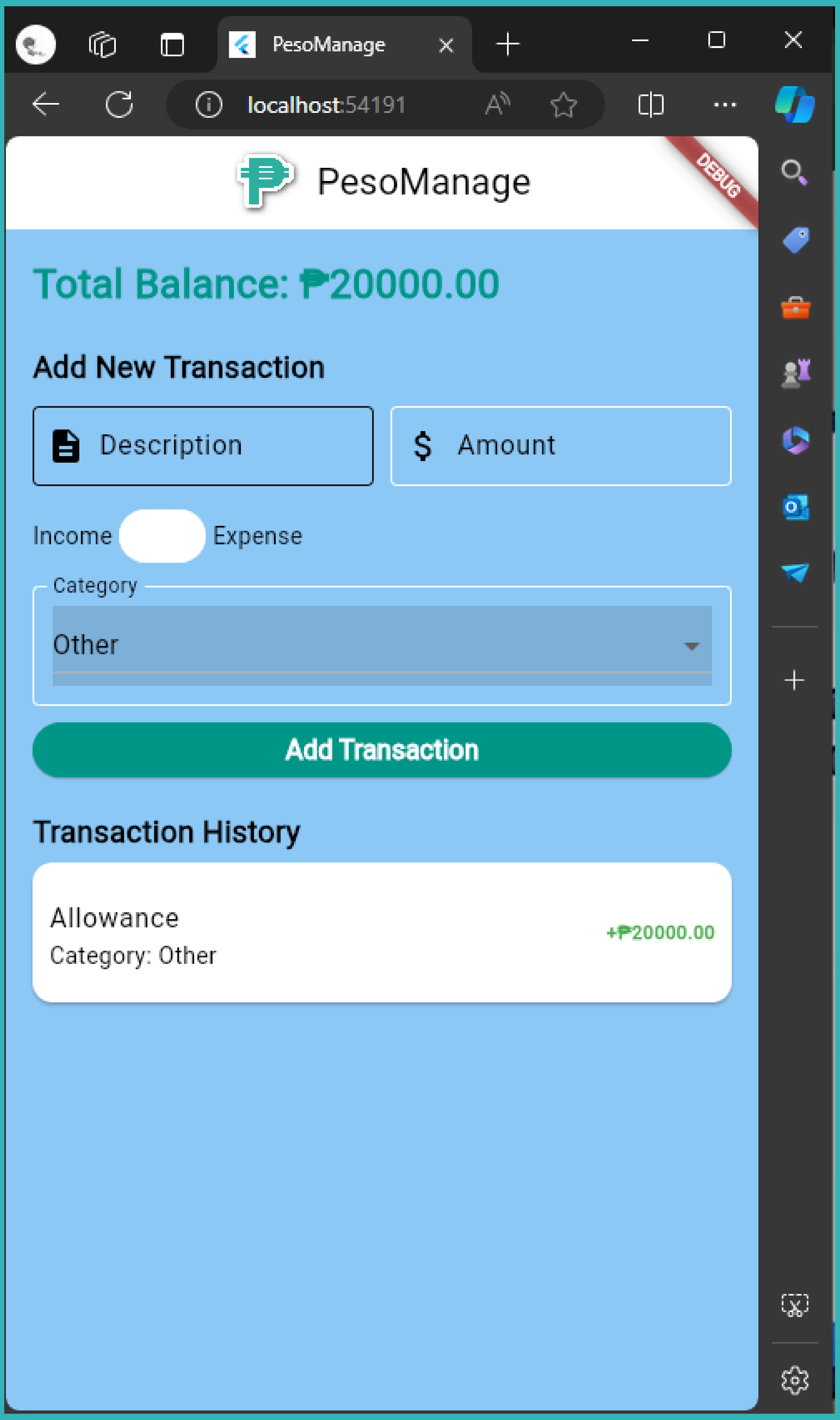
Amount: 20000

Income Expense

Category: Other

Add Transaction

Transaction History



PesoManage

Total Balance: ₱20000.00

Add New Transaction

Description | Amount

Income Expense

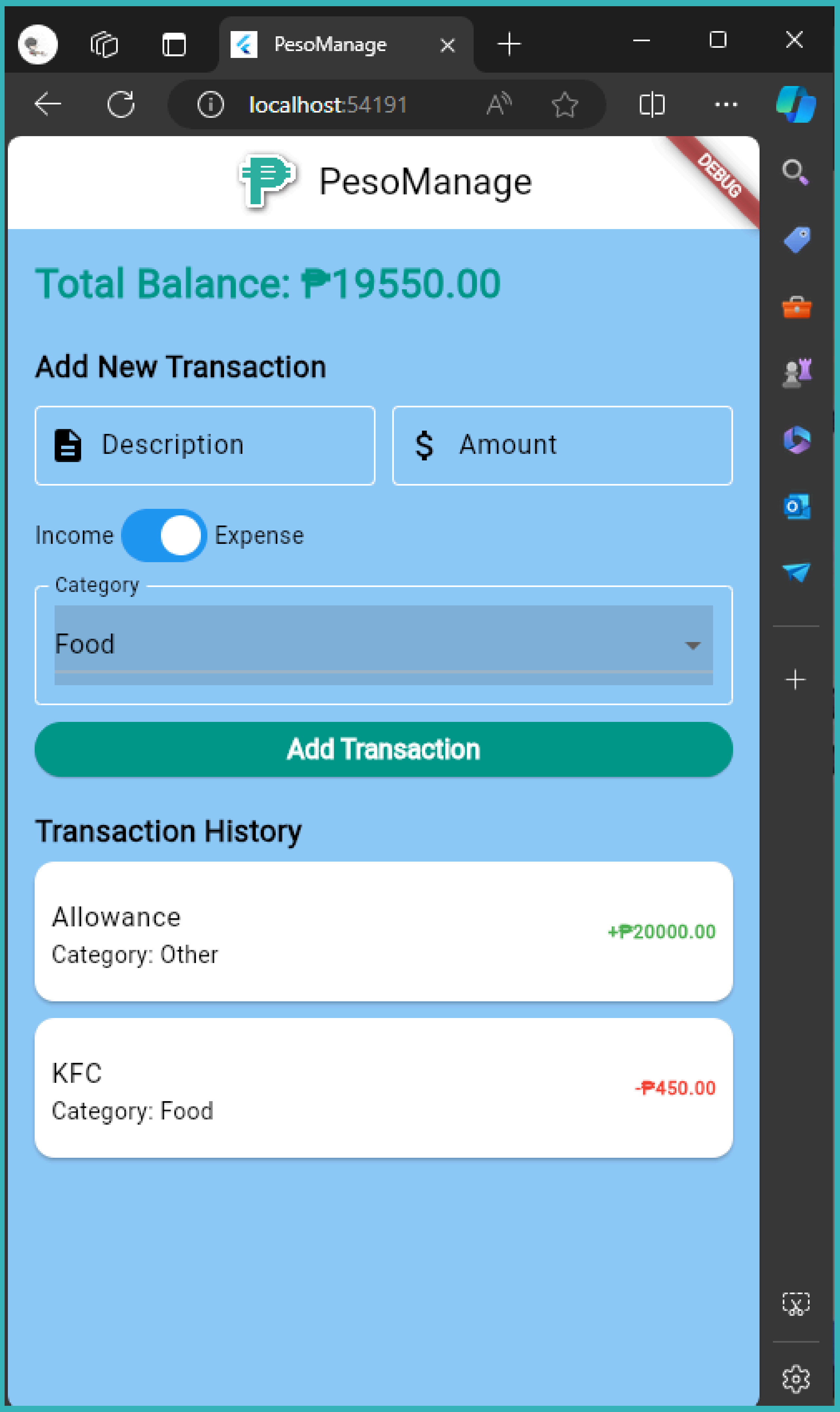
Category

Other

Add Transaction

Transaction History

Allowance
Category: Other
+₱20000.00



PesoManage

Total Balance: ₱19550.00

Add New Transaction

Description | Amount

Income Expense

Category

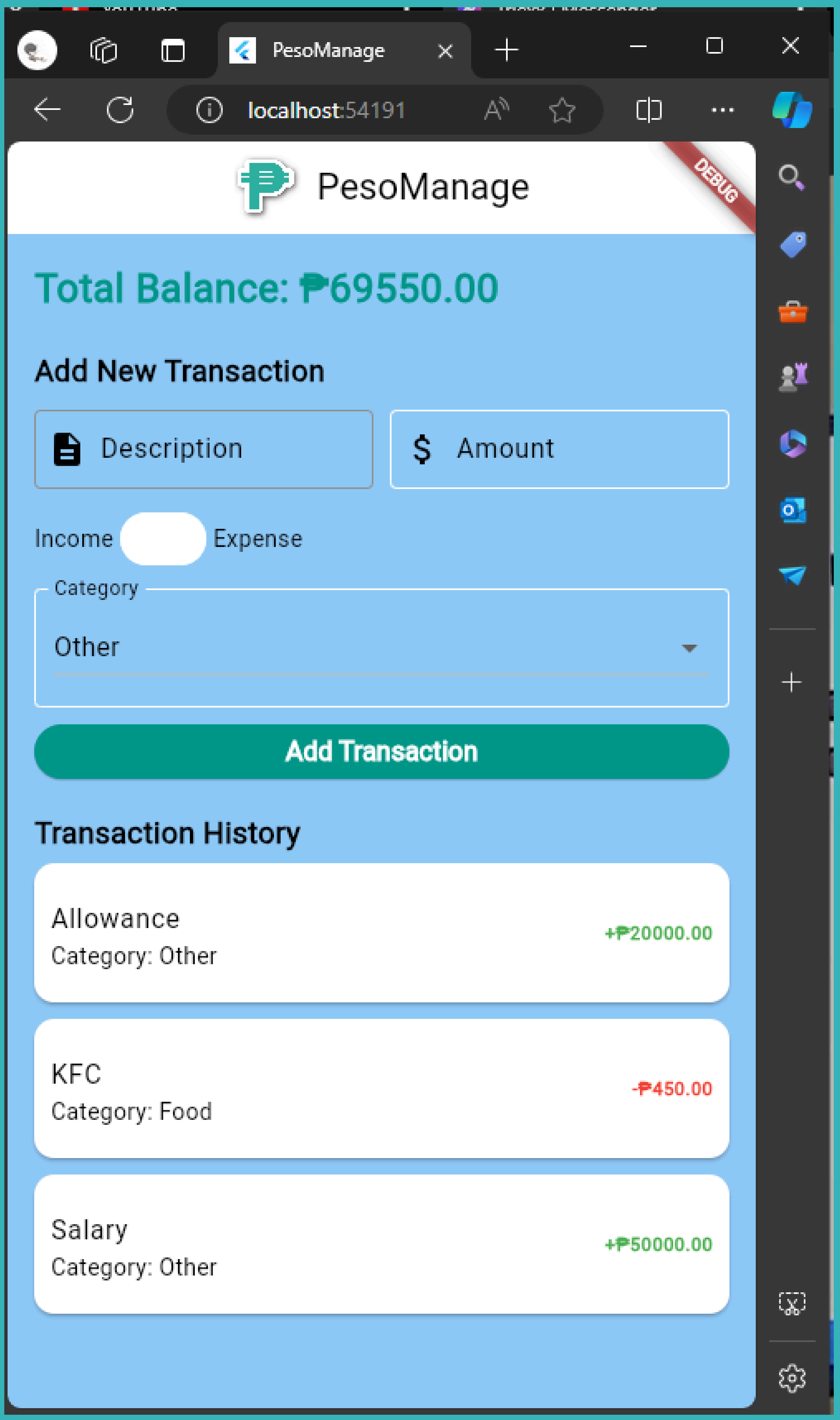
Food

Add Transaction

Transaction History

Allowance
Category: Other
+₱20000.00

KFC
Category: Food
-₱450.00



PesoManage

Total Balance: ₱69550.00

Add New Transaction

Description | Amount

Income Expense

Category

Other

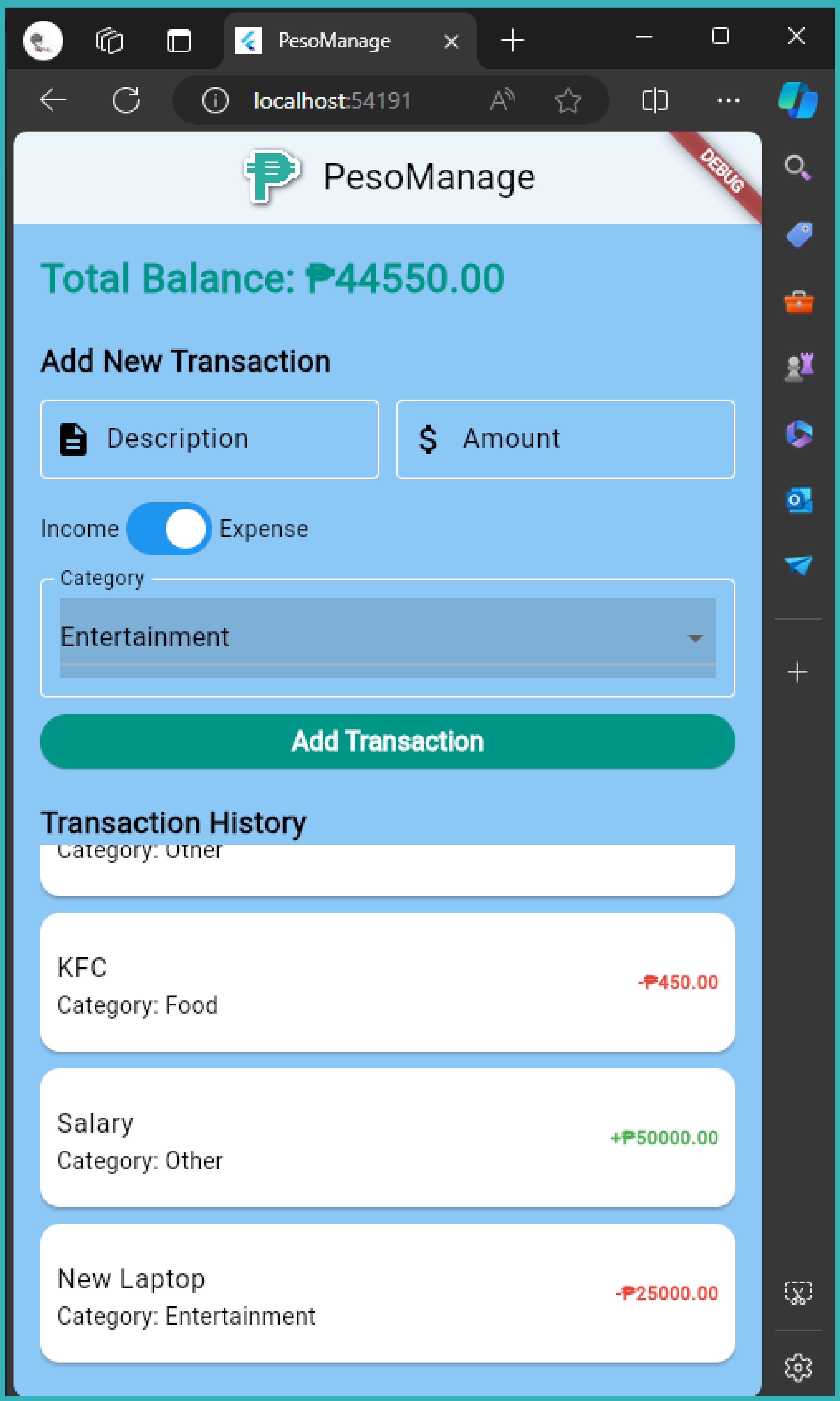
Add Transaction

Transaction History

Allowance
Category: Other
+₱20000.00

KFC
Category: Food
-₱450.00

Salary
Category: Other
+₱50000.00



PesoManage

Total Balance: ₱44550.00

Add New Transaction

Description | Amount

Income Expense

Category

Entertainment

Add Transaction

Transaction History

Category: Other

KFC
Category: Food
-₱450.00

Salary
Category: Other
+₱50000.00

New Laptop
Category: Entertainment
-₱25000.00

APPLICATION FUNCTIONS

Check Remaining Balance - user can check how much they still have for their remaining balance.

Add Income - user can add the receive amount of their income.

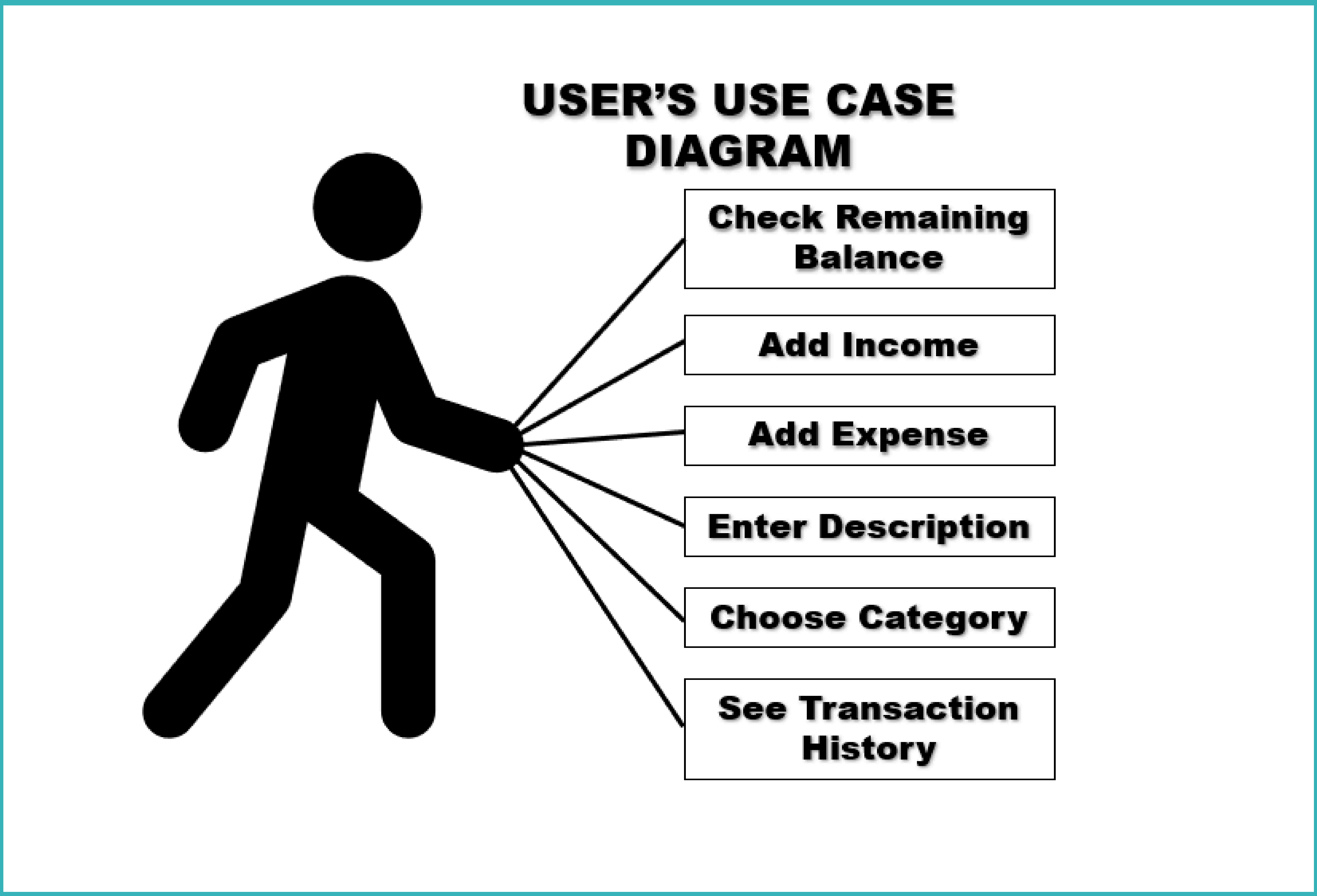
Add Expense - user's can add their expense and see how much much these can affect their balance.

Enter Description - user's can add description to add all kinds of label for its easy to determine.

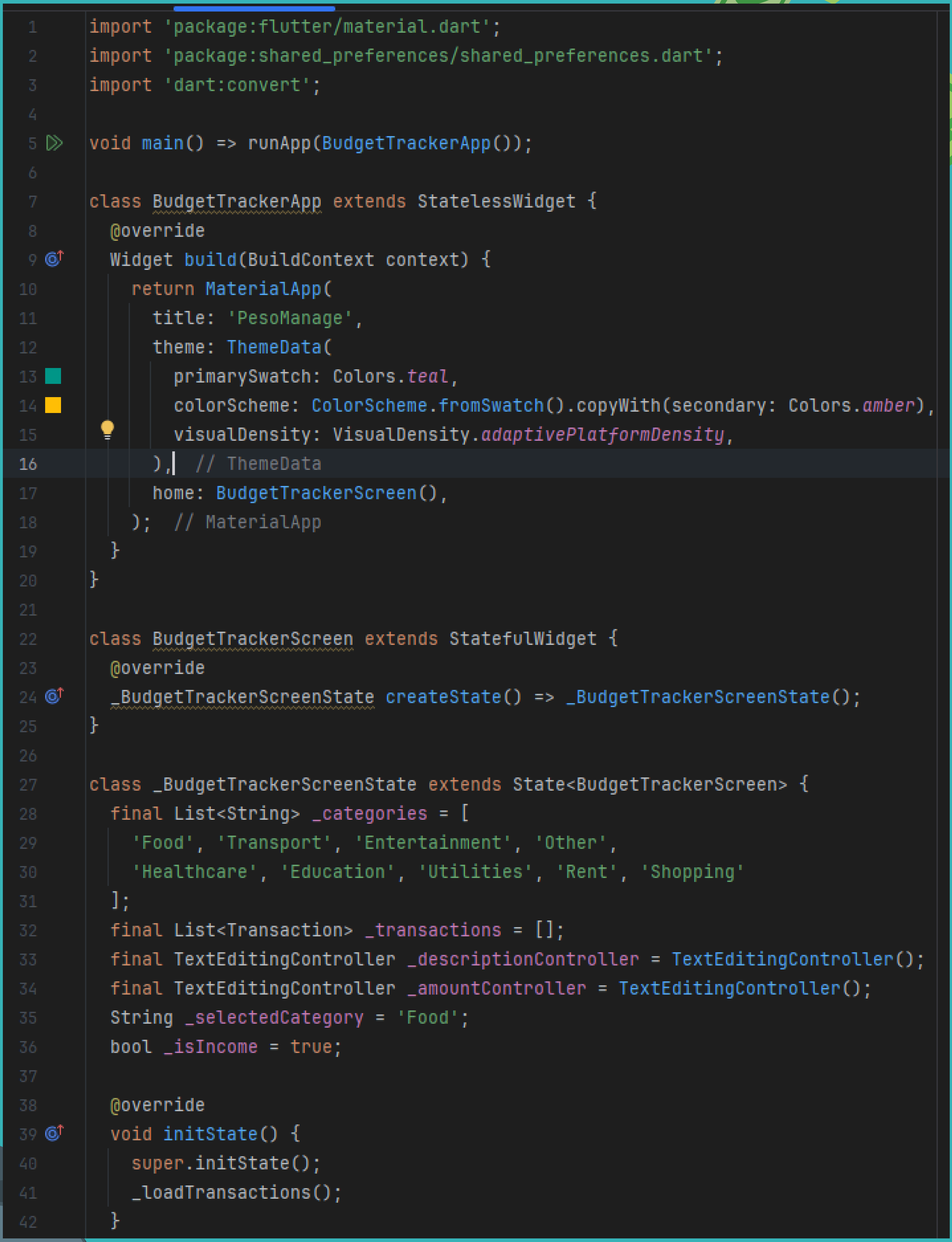
Choose Category - user's to many kind of category for ease of access.

See Transaction History - user's can see their transaction for the convenience of them being easy to see.

APPLICATION USE CASE DIAGRAM

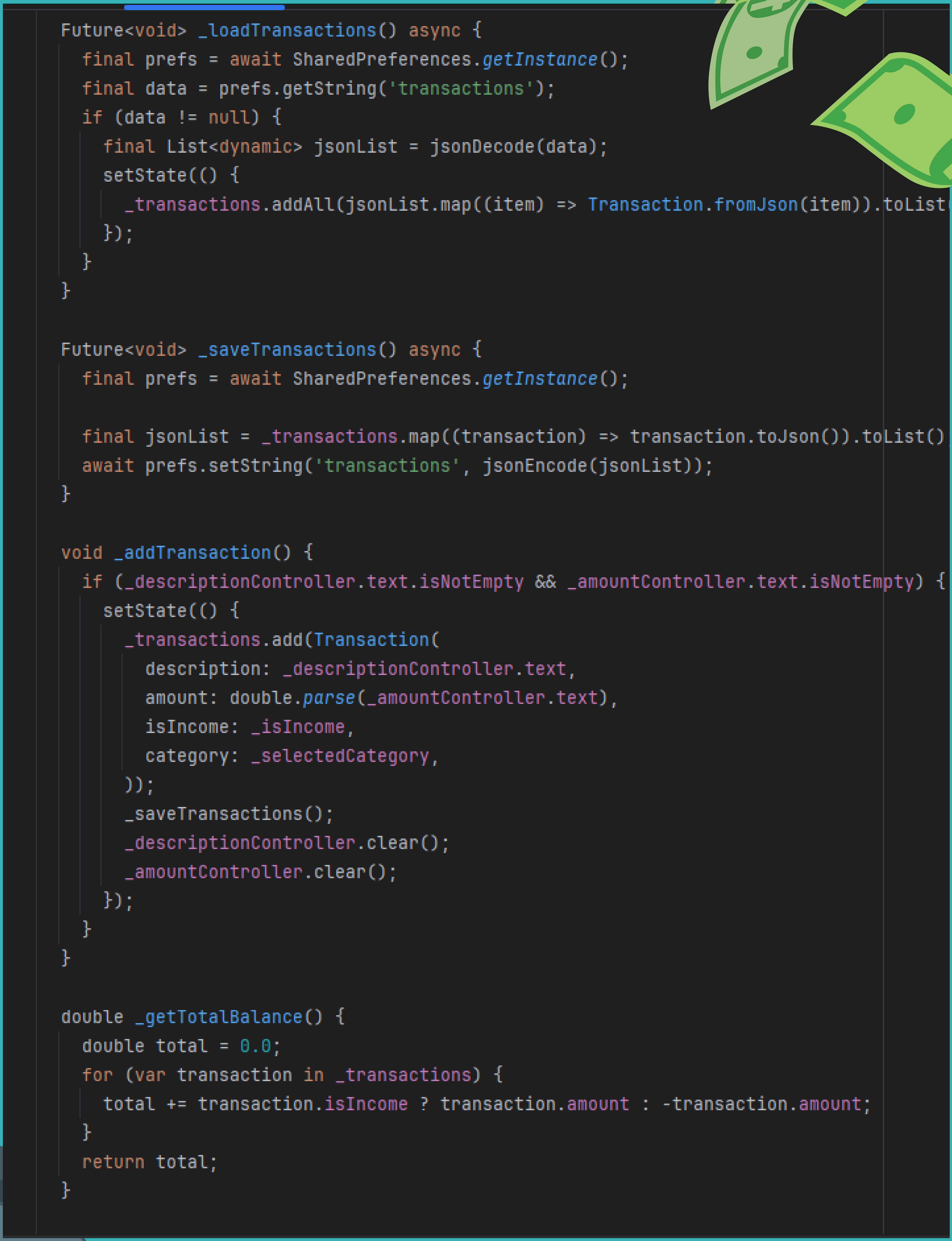


APPLICATION SOURCE CODE

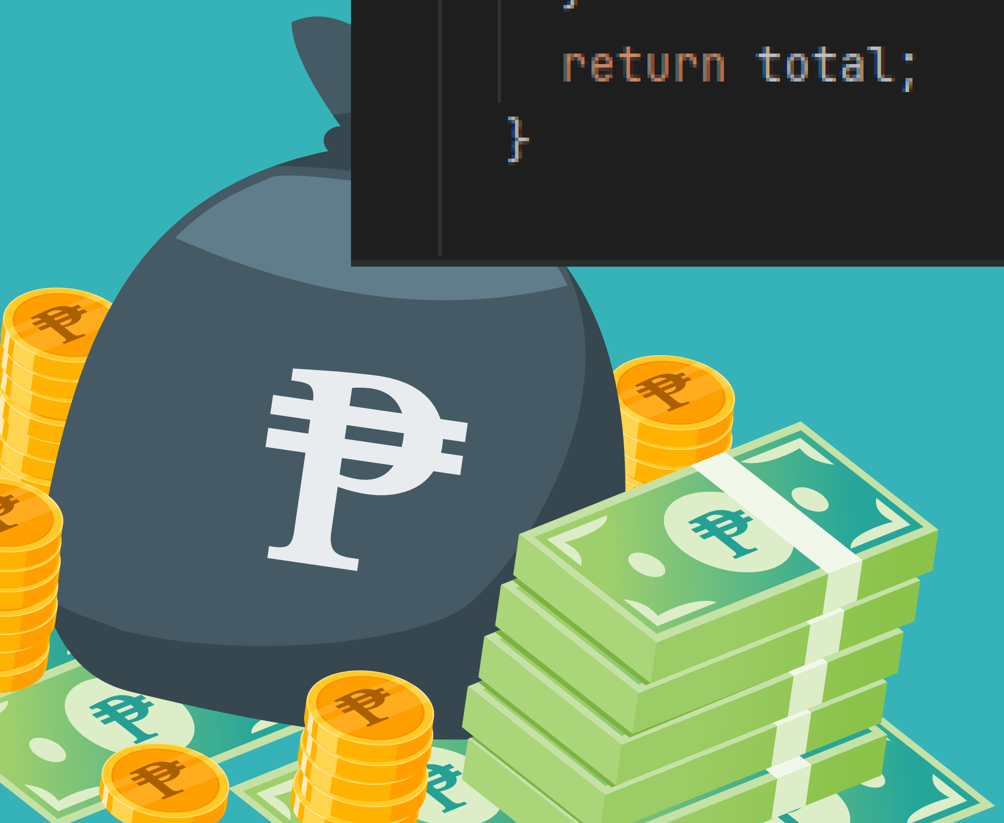


```dart
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'dart:convert';

void main() => runApp(BudgetTrackerApp());

class BudgetTrackerApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'PesoManage',
      theme: ThemeData(
        primarySwatch: Colors.teal,
        colorScheme: ColorScheme.fromSwatch().copyWith(secondary: Colors.amber),
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),  // ThemeData
      home: BudgetTrackerScreen(),
    );  // MaterialApp
  }
}

class BudgetTrackerScreen extends StatefulWidget {
  @override
  _BudgetTrackerScreenState createState() => _BudgetTrackerScreenState();
}

class _BudgetTrackerScreenState extends State<BudgetTrackerScreen> {
  final List<String> _categories = [
    'Food', 'Transport', 'Entertainment', 'Other',
    'Healthcare', 'Education', 'Utilities', 'Rent', 'Shopping'
  ];
  final List<Transaction> _transactions = [];
  final TextEditingController _descriptionController = TextEditingController();
  final TextEditingController _amountController = TextEditingController();
  String _selectedCategory = 'Food';
  bool _isIncome = true;

  @override
  void initState() {
    super.initState();
    _loadTransactions();
  }
```

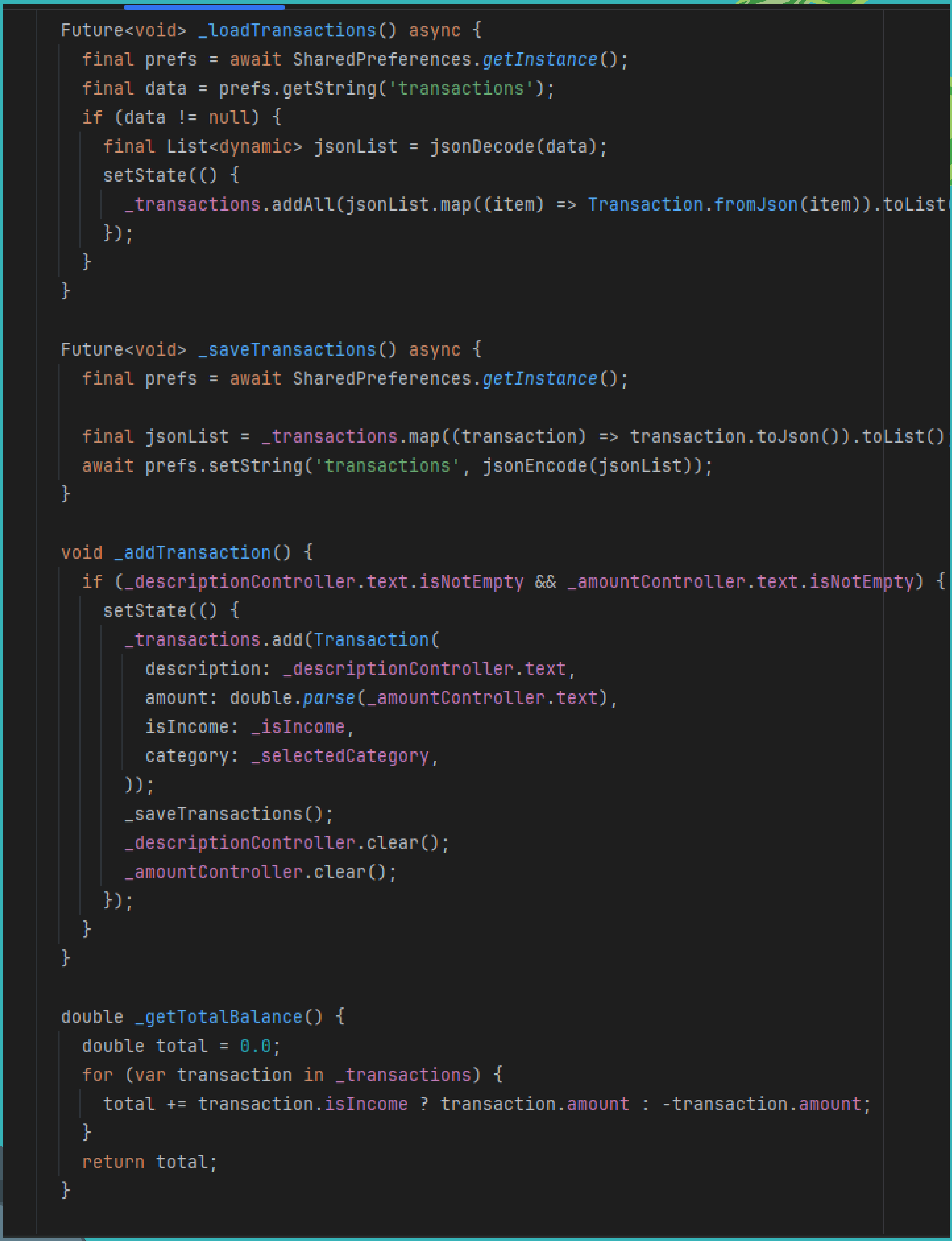
APPLICATION SOURCE CODE



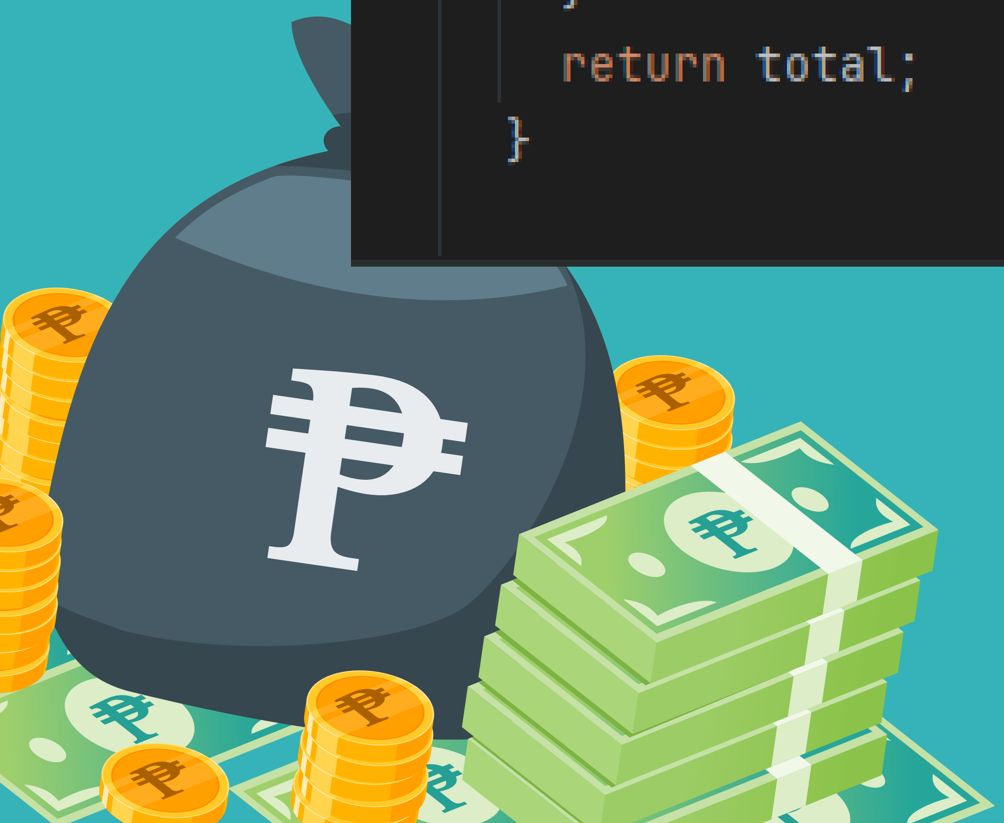
```
Future<void> _loadTransactions() async {
  final prefs = await SharedPreferences.getInstance();
  final data = prefs.getString('transactions');
  if (data != null) {
    final List<dynamic> jsonList = jsonDecode(data);
    setState(() {
      _transactions.addAll(jsonList.map((item) => Transaction.fromJson(item)).toList(
    });
  }
}

Future<void> _saveTransactions() async {
  final prefs = await SharedPreferences.getInstance();

  final jsonList = _transactions.map((transaction) => transaction.toJson()).toList()
  await prefs.setString('transactions', jsonEncode(jsonList));
}

void _addTransaction() {
  if (_descriptionController.text.isNotEmpty && _amountController.text.isNotEmpty) {
    setState(() {
      _transactions.add(Transaction(
        description: _descriptionController.text,
        amount: double.parse(_amountController.text),
        isIncome: _isIncome,
        category: _selectedCategory,
      ));
      _saveTransactions();
      _descriptionController.clear();
      _amountController.clear();
    });
  }
}

double _getTotalBalance() {
  double total = 0.0;
  for (var transaction in _transactions) {
    total += transaction.isIncome ? transaction.amount : -transaction.amount;
  }
  return total;
}
```



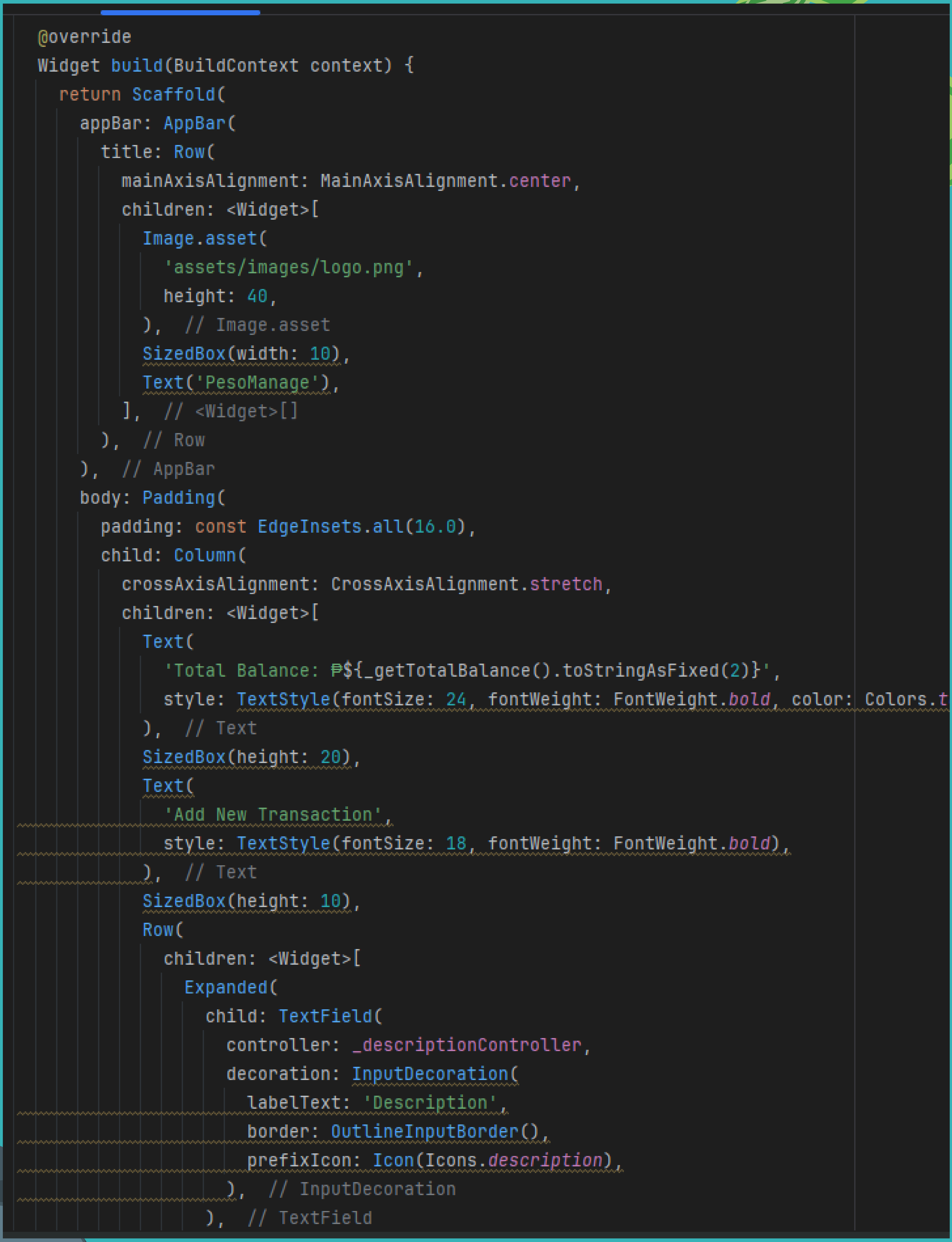
APPLICATION SOURCE CODE



```
Future<void> _loadTransactions() async {
  final prefs = await SharedPreferences.getInstance();
  final data = prefs.getString('transactions');
  if (data != null) {
    final List<dynamic> jsonList = jsonDecode(data);
    setState(() {
      _transactions.addAll(jsonList.map((item) => Transaction.fromJson(item)).toList(
    });
  }
}

Future<void> _saveTransactions() async {
  final prefs = await SharedPreferences.getInstance();

  final jsonList = _transactions.map((transaction) => transaction.toJson()).toList()
  await prefs.setString('transactions', jsonEncode(jsonList));
}

void _addTransaction() {
  if (_descriptionController.text.isNotEmpty && _amountController.text.isNotEmpty) {
    setState(() {
      _transactions.add(Transaction(
        description: _descriptionController.text,
        amount: double.parse(_amountController.text),
        isIncome: _isIncome,
        category: _selectedCategory,
      ));
      _saveTransactions();
      _descriptionController.clear();
      _amountController.clear();
    });
  }
}

double _getTotalBalance() {
  double total = 0.0;
  for (var transaction in _transactions) {
    total += transaction.isIncome ? transaction.amount : -transaction.amount;
  }
  return total;
}
```

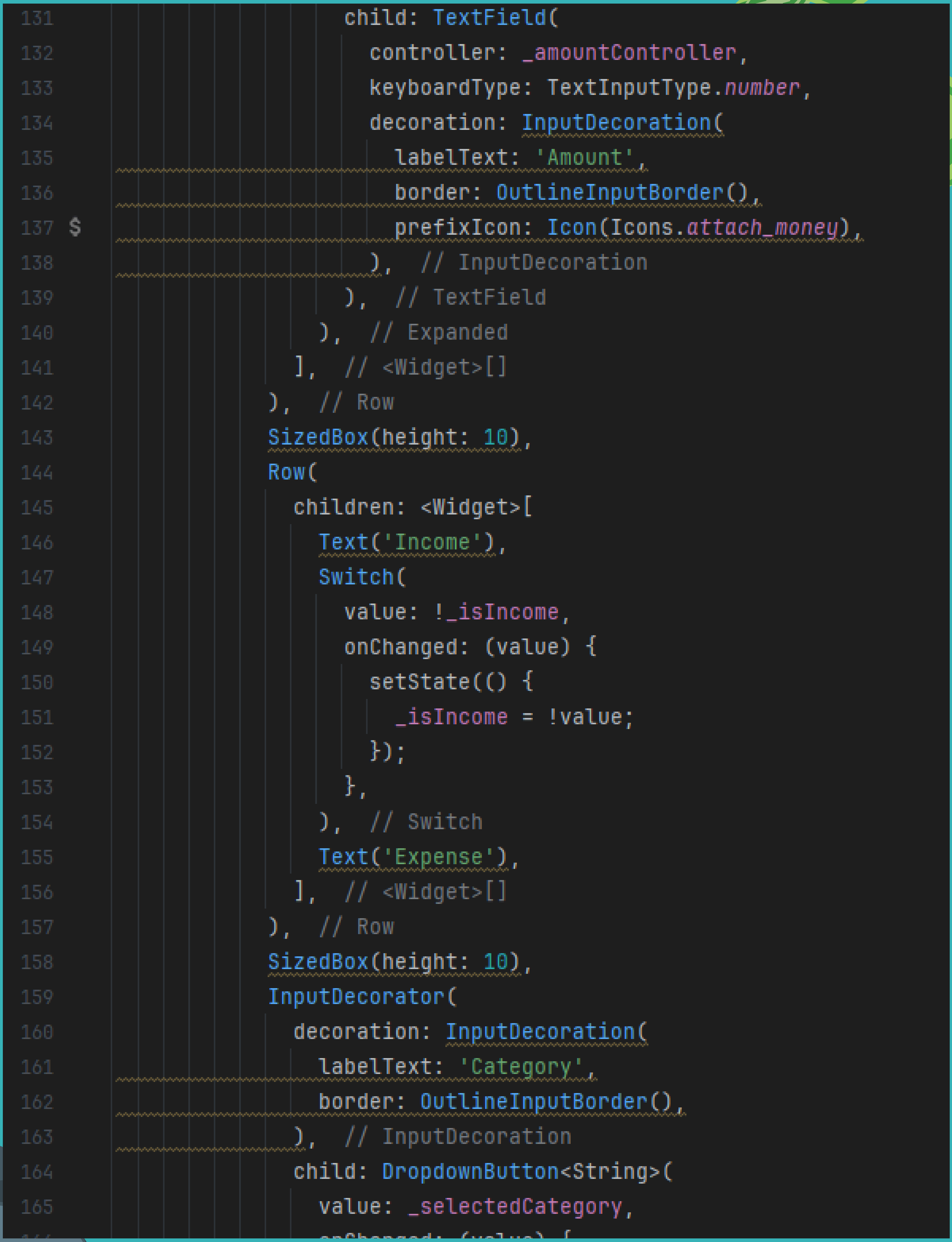


APPLICATION SOURCE CODE



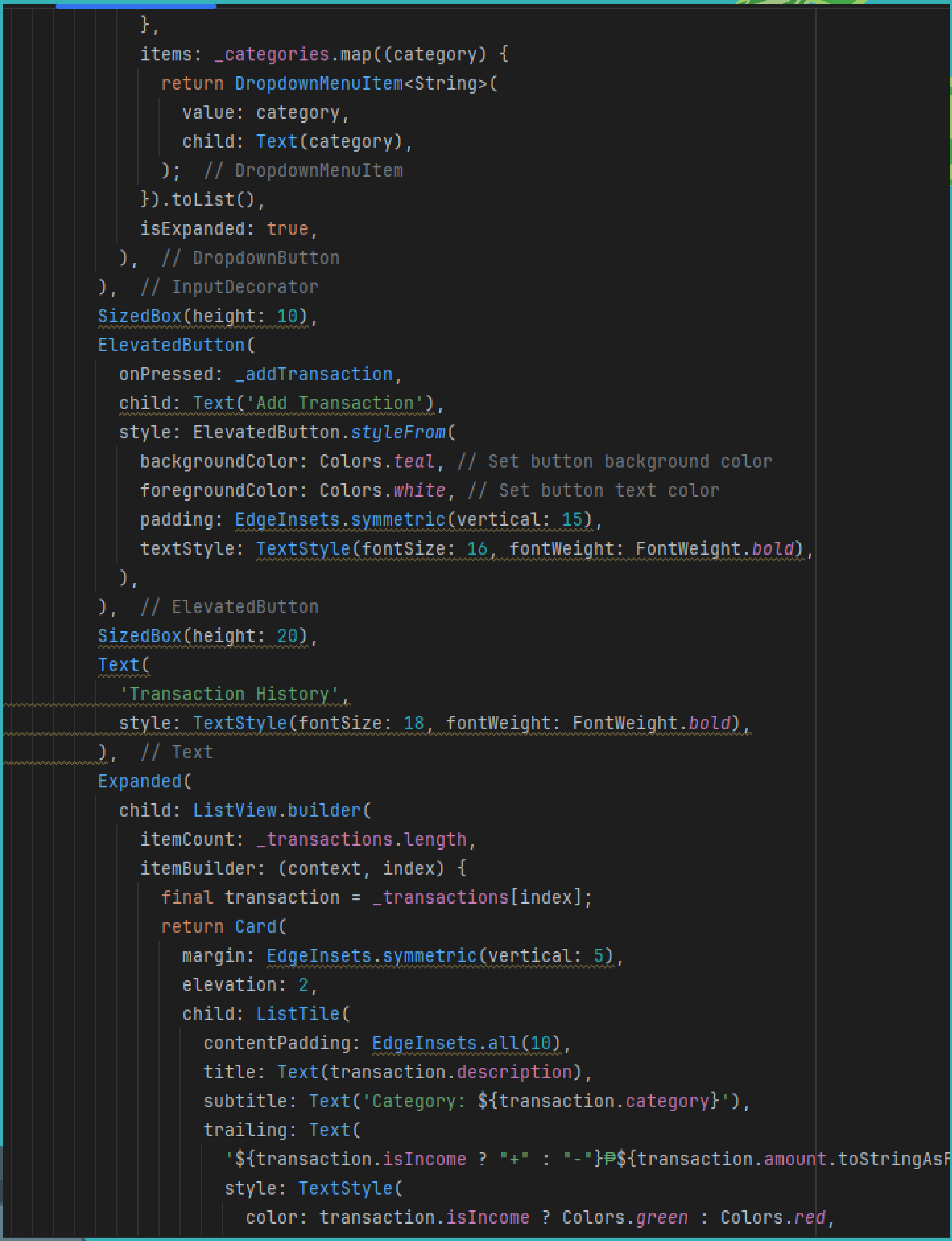
```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Image.asset(
            'assets/images/logo.png',
            height: 40,
          ),  // Image.asset
          SizedBox(width: 10),
          Text('PesoManage'),
        ],  // <Widget>[]
      ),  // Row
    ),  // AppBar
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: <Widget>[
          Text(
            'Total Balance: ₱${_getTotalBalance().toStringAsFixed(2)}',
            style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.t
          ),  // Text
          SizedBox(height: 20),
          Text(
            'Add New Transaction',
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),  // Text
          SizedBox(height: 10),
          Row(
            children: <Widget>[
              Expanded(
                child: TextField(
                  controller: _descriptionController,
                  decoration: InputDecoration(
                    labelText: 'Description',
                    border: OutlineInputBorder(),
                    prefixIcon: Icon(Icons.description),
                  ),  // InputDecoration
                ),  // TextField
```

APPLICATION SOURCE CODE



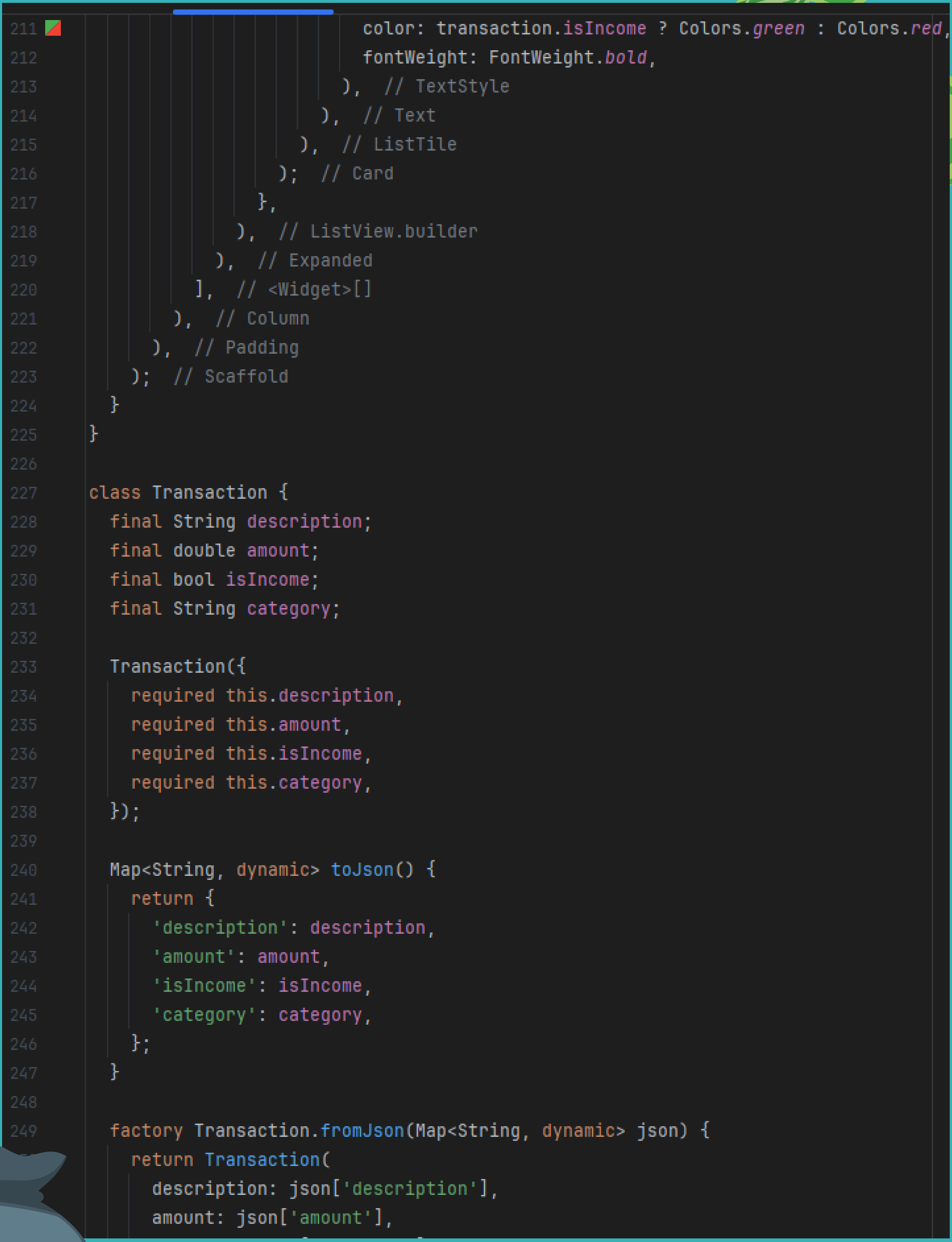
```
                  child: TextField(
                    controller: _amountController,
                    keyboardType: TextInputType.number,
                    decoration: InputDecoration(
                      labelText: 'Amount',
                      border: OutlineInputBorder(),
                      prefixIcon: Icon(Icons.attach_money),
                    ), // InputDecoration
                  ), // TextField
                ), // Expanded
              ], // <Widget>[]
            ), // Row
            SizedBox(height: 10),
            Row(
              children: <Widget>[
                Text('Income'),
                Switch(
                  value: !_isIncome,
                  onChanged: (value) {
                    setState(() {
                      _isIncome = !value;
                    });
                  },
                ), // Switch
                Text('Expense'),
              ], // <Widget>[]
            ), // Row
            SizedBox(height: 10),
            InputDecorator(
              decoration: InputDecoration(
                labelText: 'Category',
                border: OutlineInputBorder(),
              ), // InputDecoration
              child: DropdownButton<String>(
                value: _selectedCategory,
                onChanged: (value) {
```

APPLICATION SOURCE CODE



```
    },
    items: _categories.map((category) {
      return DropdownMenuItem<String>(
        value: category,
        child: Text(category),
      );  // DropdownMenuItem
    }).toList(),
    isExpanded: true,
  ),  // DropdownButton
),  // InputDecorator
SizedBox(height: 10),
ElevatedButton(
  onPressed: _addTransaction,
  child: Text('Add Transaction'),
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.teal, // Set button background color
    foregroundColor: Colors.white, // Set button text color
    padding: EdgeInsets.symmetric(vertical: 15),
    textStyle: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
  ),
),  // ElevatedButton
SizedBox(height: 20),
Text(
  'Transaction History',
  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
),  // Text
Expanded(
  child: ListView.builder(
    itemCount: _transactions.length,
    itemBuilder: (context, index) {
      final transaction = _transactions[index];
      return Card(
        margin: EdgeInsets.symmetric(vertical: 5),
        elevation: 2,
        child: ListTile(
          contentPadding: EdgeInsets.all(10),
          title: Text(transaction.description),
          subtitle: Text('Category: ${transaction.category}'),
          trailing: Text(
            '${transaction.isIncome ? "+" : "-"}₱${transaction.amount.toStringAsF
            style: TextStyle(
              color: transaction.isIncome ? Colors.green : Colors.red,
```

APPLICATION SOURCE CODE



```
                          color: transaction.isIncome ? Colors.green : Colors.red,
                          fontWeight: FontWeight.bold,
                        ),  // TextStyle
                      ),  // Text
                    ),  // ListTile
                  );  // Card
                },
              ),  // ListView.builder
            ),  // Expanded
          ],  // <Widget>[]
        ),  // Column
      ),  // Padding
    );  // Scaffold
  }
}

class Transaction {
  final String description;
  final double amount;
  final bool isIncome;
  final String category;

  Transaction({
    required this.description,
    required this.amount,
    required this.isIncome,
    required this.category,
  });

  Map<String, dynamic> toJson() {
    return {
      'description': description,
      'amount': amount,
      'isIncome': isIncome,
      'category': category,
    };
  }

  factory Transaction.fromJson(Map<String, dynamic> json) {
    return Transaction(
      description: json['description'],
      amount: json['amount'],
```