SpendWise: Personal Expense Tracker

Project Report

By

Miss Natkrittar Maswongwiwat 6688064

Miss Pundharee Puckdinukul 6688091

Miss Nichakorn Wisalkamol 6688146

Miss Praewtip Boontaweesomboon 6688185

A Report Submitted in Partial Fulfillment of

the Requirements for

ITCS259_Mobile Application Development

Faculty of Information and Communication Technology

Mahidol University

2025

Introduction and Background

Problem Statement

Many people struggle to manage a record of their daily expenditures. This lack of tracking may lead to overspending and they do not know where their money is spent. Therefore, it is difficult to analyze spending habits effectively without a simple and accessible tool.

Importance and Motivation

The motivation behind this project is to solve these financial management problems by creating an efficient mobile application. The relevance of this tool is high for students and busy users who need to track where money goes.

Use Cases

Use Case 1: Transaction Management

- **Actor:** User
- **Description:** The user interacts with the system to keep their financial records up to date by managing daily entries.
- **Steps:**
    1. The user can **add** new transactions to record both expenses and income.
    2. The user can **edit** previously recorded transactions to correct errors.
    3. The user can **delete** added transactions that are no longer needed or valid.

Use Case 2: Financial Dashboard & Monitoring

- **Actor:** User
- **Description:** The user accesses the main dashboard to get a real-time snapshot of their financial health.
- **Steps:**
    1. The user can **view** the total income and total expense calculated in real-time.
    2. The user can **view** data visualizations, specifically the **s**ummary graph and donut chart, to understand the distribution of their finances.

Use Case 3: Category Customization

- **Actor:** User
- **Description:** The user organizes their financial tracking system to fit their specific lifestyle needs by managing categories.
- **Steps:**
    1. The user can **add** custom categories to define specific income and expense types.
    2. The user can **view** the list of available categories.
    3. The user can **view** specific category bars in the transaction page.

Related Works

Existing application: Money Manager Expense & Budget

For this project, we compared our application with "Money Manager Expense & Budget" (developed by Realbyte Inc.), which is currently one of the most widely used personal financial management applications globally, with over 20 million downloads.

Money Manager is a comprehensive expense tracker that uses a double-entry bookkeeping system to ensure accurate asset management, allowing users to manually input transactions, manage various assets (cash, bank, cards), and view spending trends through dynamic graphs.

Strength

- **Detailed Visualization:** It provides extensive graphical reports, allowing users to view spending by category and changes over time.
- **Offline Capability:** Like SpendWise, it functions well without constant internet access, storing data locally on the device.
- **Customization:** Users can create their own income and expense categories rather than relying solely on defaults.

Weaknesses

- **Complexity (Learning Curve):** Due to its "double-entry" accounting nature and vast array of settings, new users often find the interface "not super intuitive" and time-consuming to learn.
- **Feature Overload:** The app includes complex features like asset transfers, credit card settlement dates, and recurrence settings that are often unnecessary for students or casual users who simply want to track daily spending.
- **Freemium Limitations:** The free version contains advertisements and limits the number of assets a user can manage.

Comparison

- **Money Manage** is more *complex* and r*equires accounting knowledge*, while **SpendWise** has a s*traightforward* system for new users.
    - **Money Management** uses a “Double-Entry Bookkeeping” system, an accounting method that is used to record every transaction in debit and credit accounts to ensure the balance of the balance sheet. However, it requires users to understand concepts like asset transfers and liability management.
    - **SpendWise** uses a simple “Single-Entry” system, which users can simply record money in and out easily. This approach does not require accounting knowledge, making it friendly and easy for people to use.

Improvements: what did we improve

1. **Easy to Use**: We designed the app to be super simple and clean. It lets users record income and expenses very quickly without confusing steps. Thip helps users track their money easily every day.
2. **Automatic Habits**: The app automatically calculates the user's “Top 5” spending categories in the donut chart. Users do not need to search through complex charts. This helps users see their spending habits immediately.
3. **Flexible Categories**: The category name can be created to fit in specific users’ lifestyle. Users will not stick with a fixed list of choices. This helps the tracking aspect of the app to be more accurate.

Methodology

System Architecture

- **Framework:** Flutter.
- **Language:** Dart.
- **Database:** SQLite.

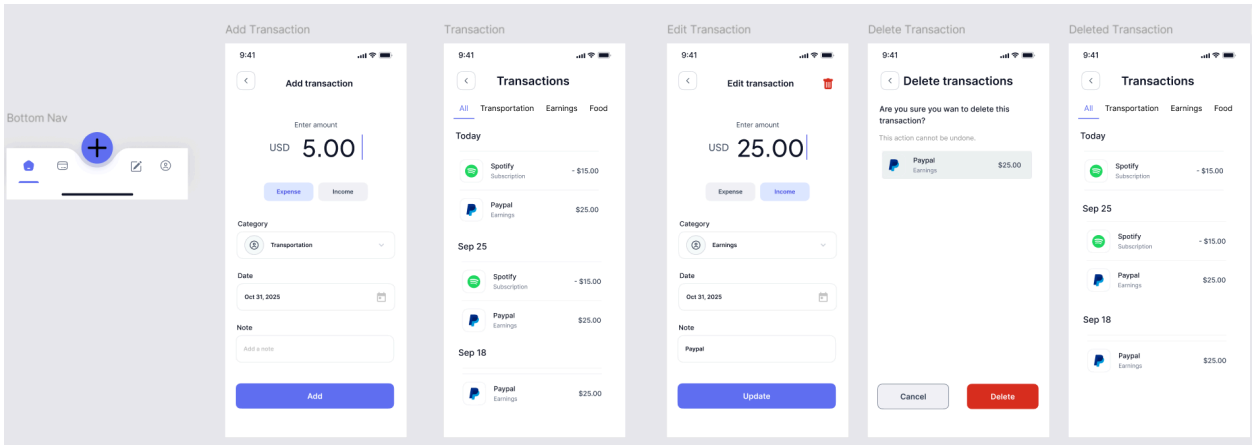
Tools and Technologies Used

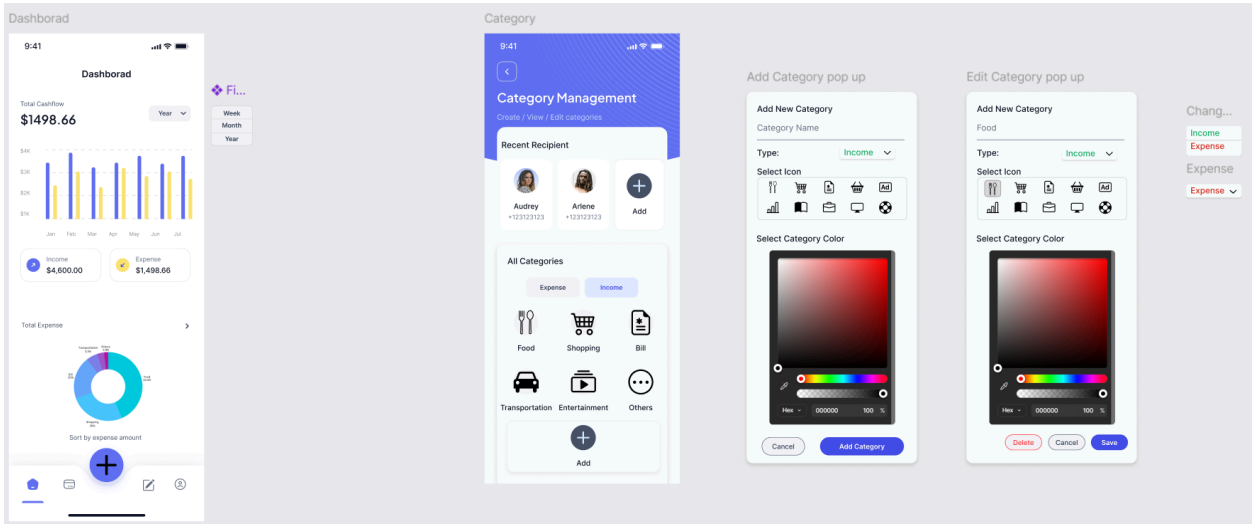
- sqflite (https://pub.dev/packages/sqflite): For local database management.
- Fl_chart (https://pub.dev/packages/fl_chart): For visualizing data through summary charts.

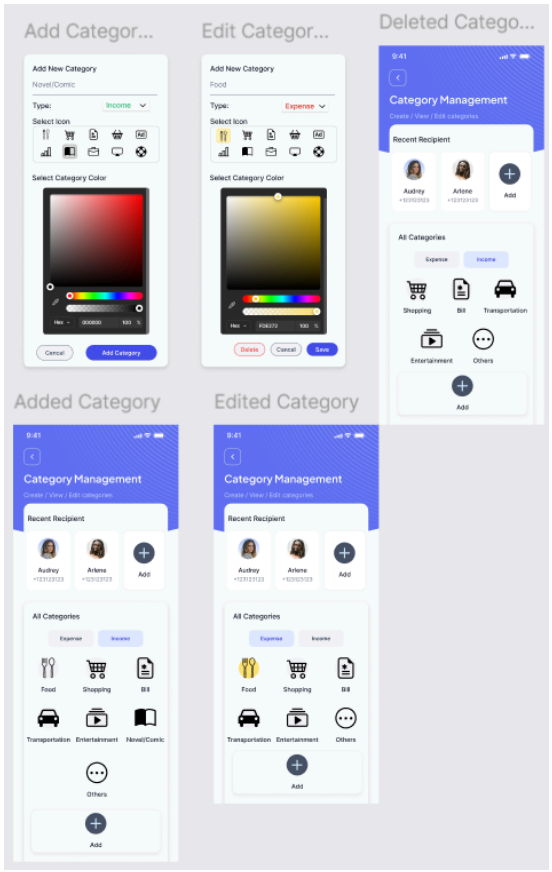
- Path package (https://pub.dev/packages/path): For joining and creating file paths, allowing the app to connect with the database file.
- path_provider (https://pub.dev/packages/path_provider): For finding the correct directories on iOS and Android where your app is allowed to store data.
- Figma: For designing the User Interface and prototype of SpendWise mobile application.

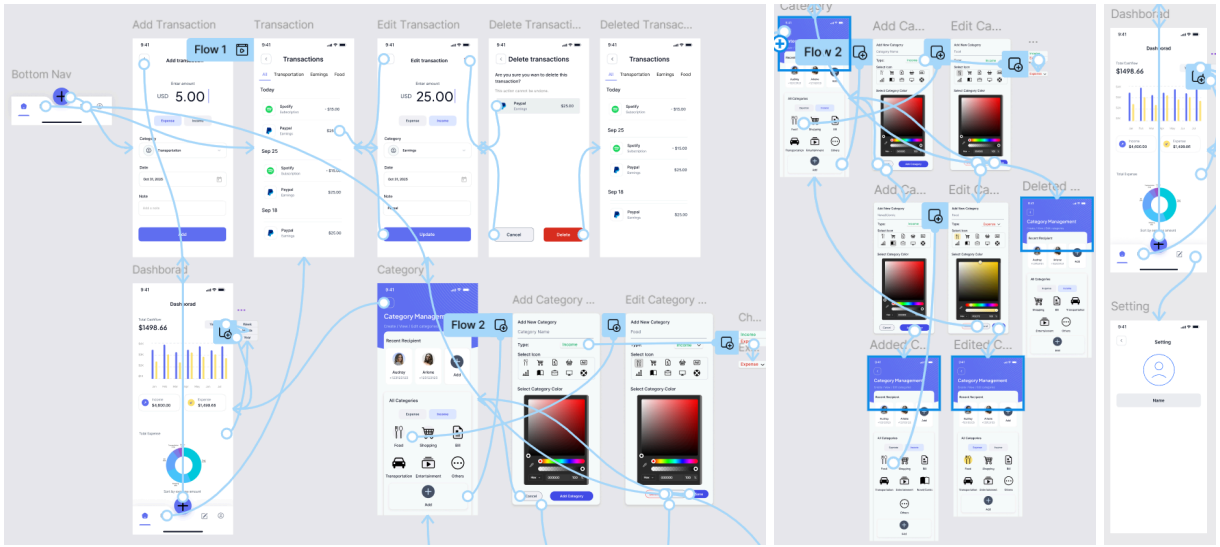
App Design and Workflow

UI Design






Add Categor...
Edit Categor...
Deleted Catego...
Added Category
Edited Category
Category Management
Recent Recipient
All Categories


Bottom Nav
Add Transaction
Transaction
Edit Transaction
Delete Transacti...
Deleted Transac...
Flow 1
Dashboard
Category
Flow 2
Add Category ...
Edit Category ...
Add Ca...
Edit Ca...
Deleted ...
Added C...
Edited C...
Setting

Workflow

Dashboard Page

a. **Input:** Users use the Dashboard or Home feature to view all Total Cashflow including yearly, monthly, daily, Dynamic bar chart (Total Income and expense) , and Top 5 expanse categories.
b. **Processing:** The app changes the graph of the Total Cashflow to yearly, monthly, or daily following the users' filter.
c. **Output:** Data is presented via a summary dashboard.

Transaction List Page

a. **Input:** Users use the Transaction feature to access all recorded incomes or expenses.
b. **Processing:** The app calculates totals and updates the financial overview in real-time.
c. **Output:** Data is presented via a summary dashboard and customizable charts.

Add Transaction Page

a. **Input:** Users use the Add Transaction feature to record incomes or expenses.
b. **Processing:** The app adds the financial overview including Amount, Category, Date, Note in real-time.
c. **Output:** Data is presented via transaction list and a summary dashboard and customizable charts.

Edit Transaction Page

a. **Input:** Users use the Edit Transaction feature to edit record user's chosen income or expenses.
b. **Processing:** The app updates the financial overview including Amount, Category, Date, Note in real-time.
c. **Output:** Data is presented via transaction list and a summary dashboard and customizable charts.

Delete Transaction Page

a. **Input:** Users use the Add Transaction feature to record incomes or expenses.
b. **Processing:** The app adds the financial overview including Amount, Category, Date, Note in real-time.
c. **Output:** Data is presented via transaction list and a summary dashboard and customizable charts.

Category Page

a. **Input:** Users use the Category feature to create, view, edit recipients or categories.
b. **Processing:** The app adds, edits, deletes the recent recipient including their name, picture and amount of transition or new category including its name, icon, and color of icon follow feature that the user chose.
c. **Output:** Data is presented via all recent recipients, category and transaction list.

Add Category Page

a. **Input:** Users use the Add Category feature to create new recipients or categories.

b. **Processing:** The app adds the recent recipient including their name, picture and amount of transition or new category including its name, icon, and color of icon.
c. **Output:** Data is presented via all recent recipients, category and transaction list.

Edit Category Page

a. **Input:** Users use the Edit Category feature to edit or delete recipients or categories.
b. **Processing:** The app updates recent recipients by changing their name, picture, and transaction amount. It also updates categories by changing their name, icon, and icon color, or deletes old categories from the Category page.
c. **Output:** Data is presented via all recent recipients, category and transaction list.

Profile page

a. **Input:** Users use the Profile feature to view and edit their profile information and settings
b. **Processing:** The app retrieves the user's current profile data, including name, picture, contact information, and preferences. When changes are made, the app updates the user's profile details and saves any modified settings, ensuring they are applied across the app.
c. **Output:** Data is presented via profile information and settings are displayed to the user, and the changes are reflected throughout all relevant features of the app.

Bottom Nav bar

a. **Input:** Users tap on 1 of 4 icons on the Bottom Nav bar to access each feature.

i. Home (First icon): Access on Dashboard Page

ii. Account book (Second icon): Access on Transaction List Page

iii. Pencil on the paper (Third icon): Access on Category Page

iv. Human in Circle (Forth icon): Access on Profile Page (Extra page)

b. **Processing:** The app accesses on each page that the user taps on the icon.

c. **Output:** Data of chosen page appears on users' phones.

Implementation Details and Code Excerpts

This section describes the key technical components and implementation logic used to develop the core features of the SpendWise application.

1. Database Implementation and Data Models

The application utilizes a local SQLite database accessed through the `sqflite` package to ensure reliable, offline data persistence, as defined in `database.dart`.

1.1 System Architecture

The data layer is structured into two main components:

1. **`DatabaseHelper` (`database.dart`):** Handles database initialization, table creation, and raw CRUD operations with SQLite.
2. **`API` (`api.dart`):** Acts as a repository layer, abstracting the SQLite calls to provide clean, domain-specific methods (e.g., `fetchTransactions`, `addCategory`) to the Flutter UI layer.

1.2 Schema Design and Initialization

The database consists of two core tables, `categories` and `transactions`. The tables are established during the initial creation of the database (`_onCreate` function).

Table	Purpose	Key Columns
categories	Stores user-defined expense/income types.	id, name, colorValue, iconCodePoint
transactions	Stores every financial record.	id, title, amount, type, categoryId (Foreign Key), date

Code Excerpt (SQLite Table Creation in database.dart):

```
Future _onCreate(Database db, int version) async {
  await db.execute('''
    CREATE TABLE categories(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
      colorValue INTEGER NOT NULL,
      defaultType TEXT NOT NULL,
      iconCodePoint INTEGER NOT NULL
    )
  ''');
  await db.execute('''
    CREATE TABLE transactions(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      title TEXT NOT NULL,
      amount REAL NOT NULL,
      type TEXT NOT NULL,
      categoryId INTEGER NOT NULL,
      date TEXT NOT NULL,
      note TEXT NOT NULL DEFAULT '',
      FOREIGN KEY (categoryId) REFERENCES categories (id)
    )
  ''');
```

2. Transaction Logging and Persistence (CRUD)

This feature enables users to record, edit, and delete financial entries.

2.1 Add Transaction (add_transaction.dart)

The _addTransaction function processes the user input, maps it to the Transaction model, and commits it to the database using the API layer.

Code Excerpt (API call in add_transaction.dart):

```
  await _api.addTransaction(newTransaction);
  globalRefreshTrigger.value++;

  if (mounted) {
    Navigator.pop(context);
  }
} catch (e) {
  print("Error adding transaction: $e");
  if (mounted) setState(() => isLoading = false);
}
```

2.2 Edit and Delete Transaction (edit_transaction.dart & delete_transaction.dart)

Updating a transaction involves retrieving the new form data, finding the corresponding categoryId, and passing the updated Transaction object to the persistence layer.

```
// 1. Find the Category ID based on the selected name
final categoryObj = _allCategories.firstWhere(
  (c) => c.name == selectedCategoryName,
  orElse: () => _allCategories.first,
);
```

3. Aggregate Financial Overview and Visualization

The dashboard relies on specialized SQL queries executed by database.dart to calculate summaries and group data for the visual components.

3.1 Aggregate Financial Overview (Net Balance)

The system calculates Total Income, Total Expense, and the final Net Balance using a single raw query that employs SQL's SUM function with CASE logic to conditionally sum income and expense amounts within the selected date range.

Code Excerpt (SQL Query for Financial Summary in database.dart):

```
final List<Map<String, dynamic>> result = await db.rawQuery('''
  SELECT
    SUM(CASE WHEN type = 'income' THEN amount ELSE 0 END) AS totalIncome,
    SUM(CASE WHEN type = 'expense' THEN amount ELSE 0 END) AS totalExpense
  FROM transactions
  WHERE date BETWEEN ? AND ?
''', [formattedStartDate, formattedEndDate]);
```

3.2 Expense Insight Generation (Top 5 Categories)

To power the pie chart, a query joins the transactions table with the categories table, filters only expense type entries, groups the results by category, and orders them by the total amount spent, limited to 5 results.

Code Excerpt (SQL Query for Top Spending Categories in database.dart):

```
final List<Map<String, dynamic>> maps = await db.rawQuery('''
  SELECT
    c.name AS categoryName,
    c.colorValue AS colorValue,
    SUM(t.amount) AS totalAmount
  FROM transactions t
  INNER JOIN categories c ON t.categoryId = c.id
  WHERE t.type = 'expense' AND t.date BETWEEN ? AND ?
  GROUP BY c.id, c.name, c.colorValue
  ORDER BY totalAmount DESC
  LIMIT ?
''', [formattedStartDate, formattedEndDate, limit]);
```

3.3 Cashflow Visualization (Bar Chart Data)

To generate the monthly bar chart (when the filter is set to 'Year'), the application uses SQL's SUBSTR function to extract the month number from the ISO 8601 date string, groups the transactions by this month, and calculates the total income and expense for each period.

Code Excerpt (SQL Query for Monthly Cashflow in database.dart):

```
final List<Map<String, dynamic>> maps = await db.rawQuery('''
  SELECT
    CAST(SUBSTR(date, 6, 2) AS INTEGER) AS month,
    SUM(CASE WHEN type = 'income' THEN amount ELSE 0 END) AS totalIncome,
    SUM(CASE WHEN type = 'expense' THEN amount ELSE 0 END) AS totalExpense
  FROM transactions
  WHERE date BETWEEN ? AND ?
  GROUP BY month
  ORDER BY month ASC
''', [formattedStartDate, formattedEndDate]);
```

This raw, processed data is then consumed by the fl_chart package in dashboard.dart to render the visualization.

4. Flexible Categorization System

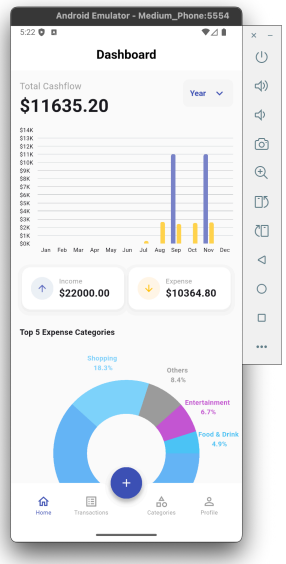
The Category management screens (add_catagory.dart, edit_catagory.dart) use models to store user-defined visual attributes:

- **colorValue:** Stores the color as an integer (Color.value) for database persistence.
- **iconCodePoint:** Stores the integer code point for the selected MaterialIcons icon.

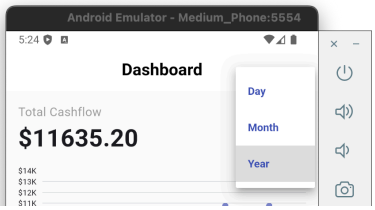
This allows the application to dynamically retrieve both the name and the corresponding visual style for every transaction, ensuring a highly customizable and visually distinct tracking experience.

Results

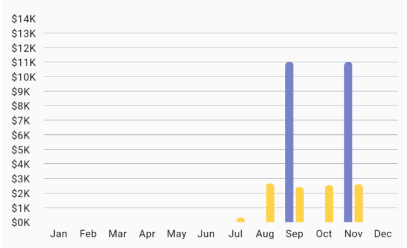
Dashboard page

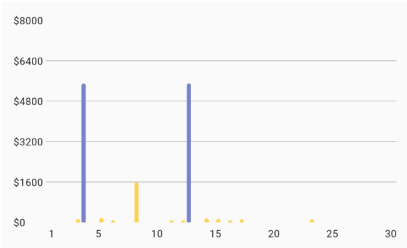


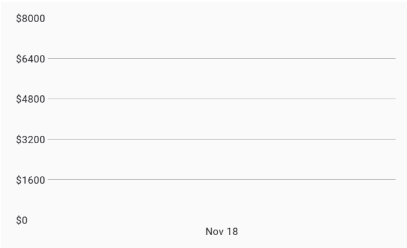
The dashboard page consists of 4 main components such as:



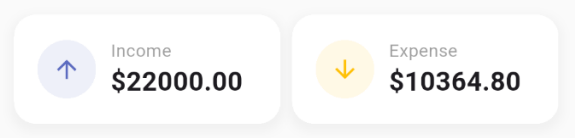
1. Financial Summary: Display the Total cashflow shows the total cashflow recorded in the database. The user can can the time period using the day/month/year dropdown in the top right of the screen.



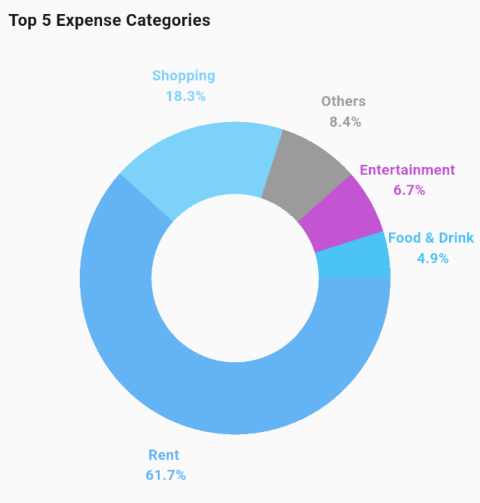




2. Dynamic Bar chart: Display the comparison of income/expense between each day/month. The user can also dynamically switch between views.

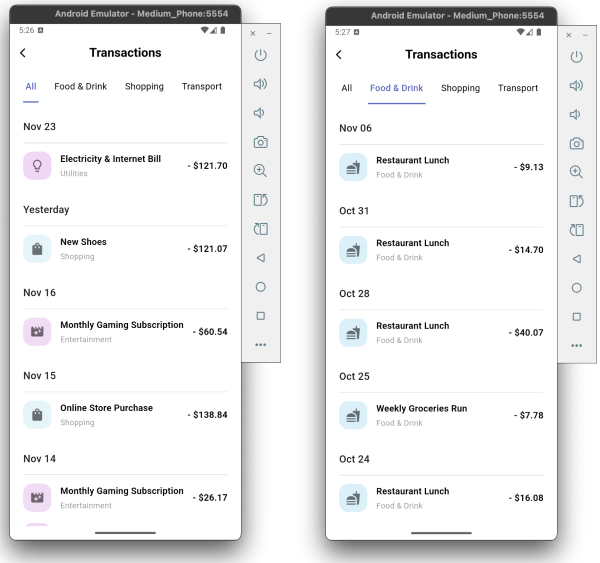


3. Expense/Income Summary: Display amount of incomes and expenses in the year.



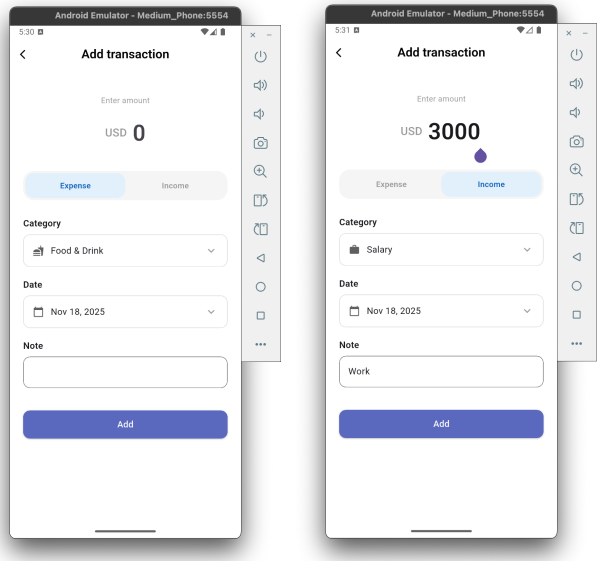
4. Top 5 expense categories: Display top 5 expense categories that users spend in the current year in percentage into Pie chart.

Transaction List page



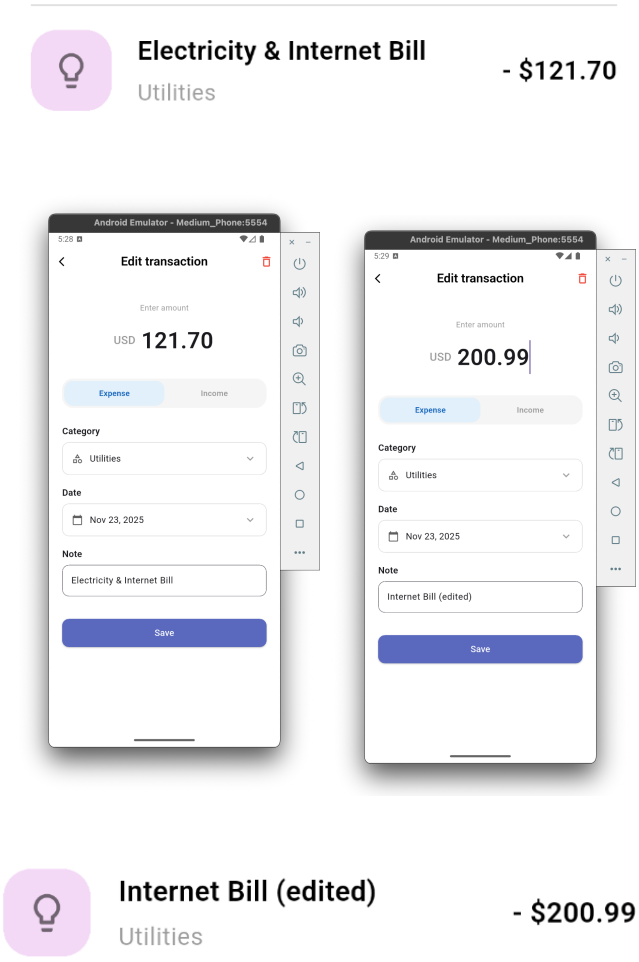
The transaction list page displays all and separated transactions which the users' created.

Add Transaction page



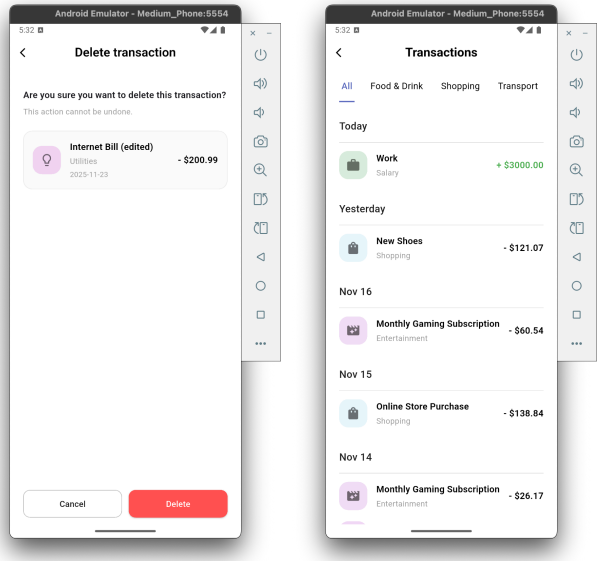
The Add Transaction page displays the Add Transaction details. Users can enter the amount of transaction, choose type of cashflow (expense or income), categories, date, and add notes to remind the user. After tapping the Add button, the recent transaction will be displayed in the Transaction page.

Edit Transaction page



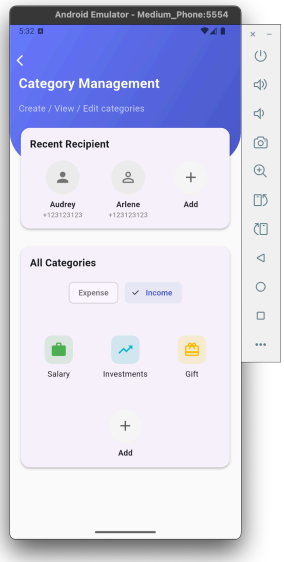
The Edit Transaction page displays the original transaction that the user has selected to edit and delete icon for reaching the Delete Transaction page. After tapping the Save button, the newly edited transaction is displayed on the Transactions page.

Delete Transaction page



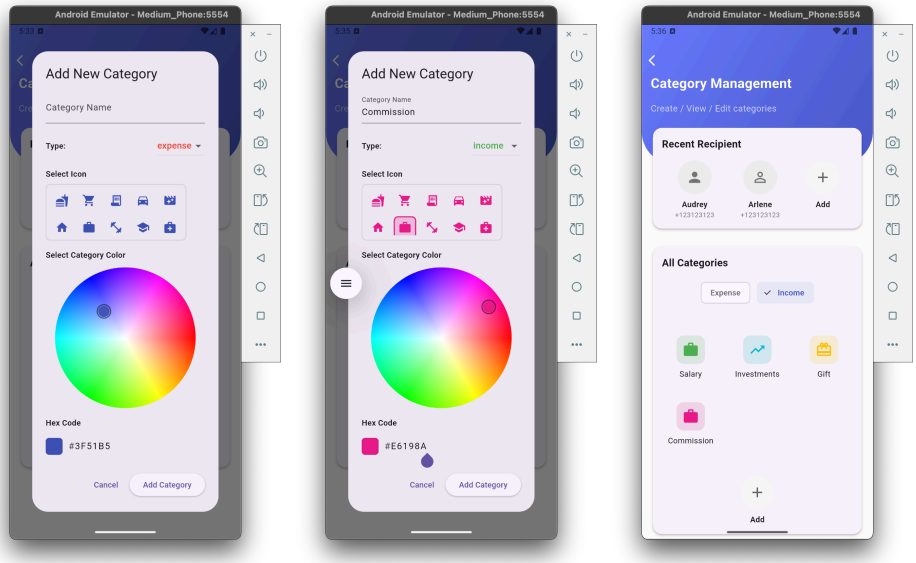
The Delete Transaction page displays the details of the transaction selected for deletion. After the user taps the Delete button, the transaction list is presented, excluding the transaction that was just deleted.

Category page



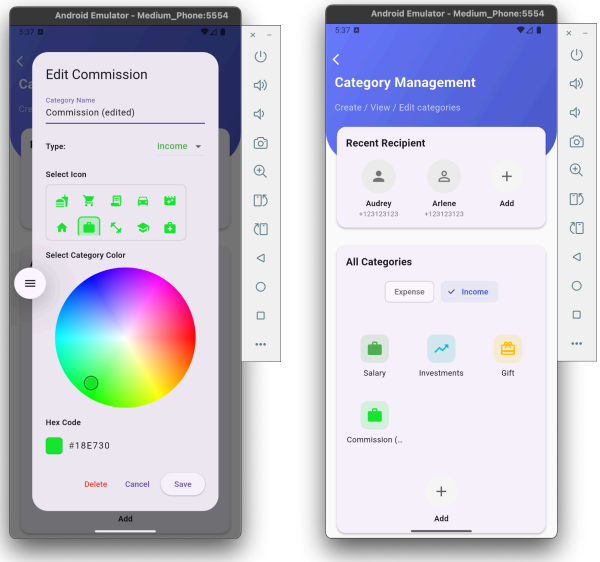
The Category page displays all recent recipients and available category options. Users may add their own categories as desired.

Add Category page



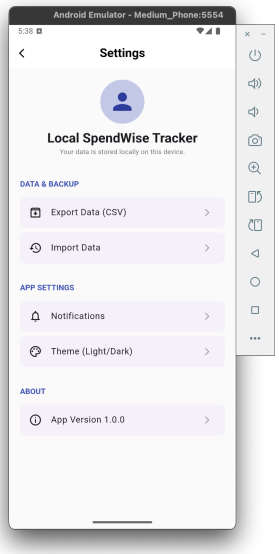
The Add Category page provides inputs for the category name, type (income or expense), icon, and color to enable users to create their own category.

Edit Category page



The Edit Category page provides the original category name, type (income or expense), icon, and color to enable users to modify the category.

Profile page



The Profile page (Extra page) serves as the user's individual settings page.

Conclusion

The ***SpendWise: Personal Expense Tracker*** project successfully achieved its primary goal: to deliver an intuitive and efficient mobile application for tracking daily finances. Built using Flutter and a local SQLite database, the application provides a robust and accessible solution for students and busy users seeking better clarity on their spending habits.

Key Achievements and Project Successes

The project delivered all planned core functionalities, such as:

- **Comprehensive Transaction Management:** Users can reliably record, edit, and delete both income and expense transactions, providing a complete financial log.
- **Flexible Categorization:** The system enables users to define and assign custom categories with unique icons and colors, ensuring flexible tracking tailored to various lifestyles.
- **Real-time Financial Overview:** The application successfully implements aggregate functions to calculate and display the total income, total expenses, and the current net balance in real-time on the dashboard.
- **Data Visualization and Insights:** Through the use of SQLite aggregation queries and the fl_chart library, the app provides critical visual tools, including:
    - A monthly/daily cashflow Bar Chart with adjustable date filtering.
    - A Pie Chart and list highlighting the Top 5 spending categories for rapid expenditure insight generation.

Lessons Learned

Throughout the development process, several key technical and collaboration lessons were learned:

- **Complexity of Data Aggregation:** Designing efficient SQLite queries for complex time-series analysis (e.g., monthly cashflow over a yearly range) was technically challenging. This required careful use of SQL functions like `SUM` and `SUBSTR` to ensure accurate and fast data grouping.
- **UI State Management:** Managing the asynchronous data flow from the local database to the numerous UI components (dashboard summary, charts, transaction lists) emphasized the necessity of using tools like the `ValueNotifier` (`globalRefreshTrigger`) to coordinate state changes efficiently across the application.
- **UI/UX Refinement:** Developing a clean and intuitive user interface required iterative testing and refinement to make the complex process of financial logging simple and accessible.

Future Improvements

While SpendWise is a fully functional product, several areas have been identified for future enhancement:

1. **Reporting and Export:** Implement the planned feature to Export Data (CSV) to allow users to move their data off-device for external analysis or backup.
2. **Authentication/Security:** Introduce password or biometric lock features to secure sensitive financial data stored locally.

3. **Advanced Budgeting:** Integrate functionality to allow users to set monthly spending limits for specific categories and provide warnings when limits are approached.

In conclusion, ***SpendWise*** successfully provides a robust, efficient, and visually informative financial tracker, meeting all project requirements and providing users with a valuable tool for monitoring and managing their personal finances.

Responsibilities of Each Member

- Ms. Natkrittar Maswongwiwat (6688064)
    - Frontend development: Transaction page, and Edit transaction page
    - Backend API
    - Testing and Debugging
    - Documentation and Presentation
- Ms. Pundharee Puckdinukul (6688091)
    - Frontend development: Dashboard page, Profile page, main page, and Add & Edit & Delete Category page
    - Database design
    - Testing and Debugging
- Ms. Nichakorn Wisalkamol (6688146)
    - Frontend development: Create & View Category page
    - UI/UX design
    - Testing and Debugging
- Ms. Praewtip Boontaweesomboon (6688185)
    - Frontend development: Add & Delete transaction page
    - UI/UX design
    - Testing and Debugging

References

[1] D. Hub, “Visualize Like a Pro in Flutter — Mastering Charts with fl_chart 📊,” Medium, Aug. 14, 2025. https://medium.com/pubdev-essentials/visualize-like-a-pro-in-flutter-mastering-charts-with-fl-chart-163fb9b4e86a

[2] “fl_chart | Flutter Package,” Dart packages. https://pub.dev/packages/fl_chart

[3] “path | Dart Package,” Dart packages. https://pub.dev/packages/path

[4] “path_provider | Flutter Package,” Dart packages. https://pub.dev/packages/path_provider

[5] “sqflite | Flutter Package,” Dart packages. https://pub.dev/packages/sqflite