FINAL PROJECT REPORT

SEMESTER 2, ACADEMIC YEAR: 2024-2025 CT312H: MOBILE PROGRAMMING

- **Project/Application name: Expense Tracking Application**
- **GitHub link: https://github.com/24-25Sem2-Courses/ct312hm04-projecttnxk19**
- **Link Youtube demo:** https://www.youtube.com/watch?v=JhhWgMdQbPw
- **Student ID 1: B2111985**
- **Student Name 1: Tran Nguyen Xuan Khanh**
- **Student ID 2: B2111963**
- **Student Name 2: Ho Kim Trong**
- **Class/Group Number: CT312HM04**

I. Introduction

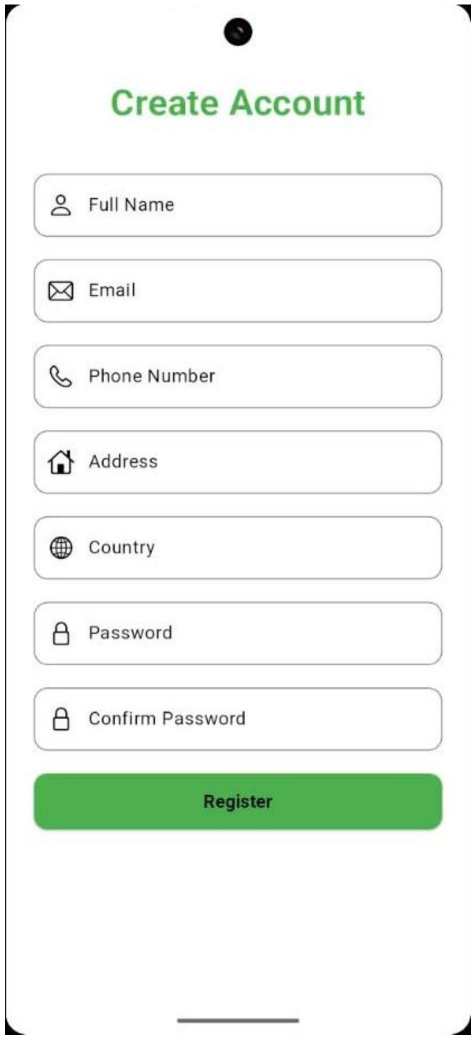
- Aapplication description: Our Expense Tracking App helps users manage their finances efficiently. With features to add, edit, and track expenses, income and display the current balance. Therefore, users can easily monitor their financial health.
- A task assignment sheet for each member if working in groups.

Member	Tasks
Tran Nguyen Xuan Khanh	UI and function for main pages
Ho Kim Trong	Features for registration, login, and database connection with PocketBase.

-

II. Details of implemented features 1.

Application page 1: Registration



- **Description:** What is this feature/application page? What is it used for?
- **Screenshots:** some screenshots of this feature/application page.
- **Implementation details:** students should answer the following questions:

  + List the widgets used for this feature/page. Is there any special widget (not introduced in the lesson) used? If so, state them.

  + Does the feature use any libraries or plugins? If so, state them and state the role of those libraries/plugins.

  + Does this feature use a shared state management solution? If so state briefly how your solution works and describe the code architecture.

  + Does this feature read or store any data? Locally or remotely? If so, state the data table structure. If you are using a REST API, briefly describe that API (how to call, input, output).

The User Registration feature allows new users to create an account by entering their personal details such as name, email, phone, address, country, and password.

The app securely stores this information in a PocketBase database, enabling authentication and user-specific data management.
Implementation Details:

Widgets Used:
Form : used for form validation and submission).
TextFormField: Input fields for name, email, phone, address, country, password, and password confirmation.
ElevatedButton: Button for submitting the registration form.
ScaffoldMessenger: Displays success or error messages.
CircularProgressIndicator: Shows a loading indicator when the request is processing.

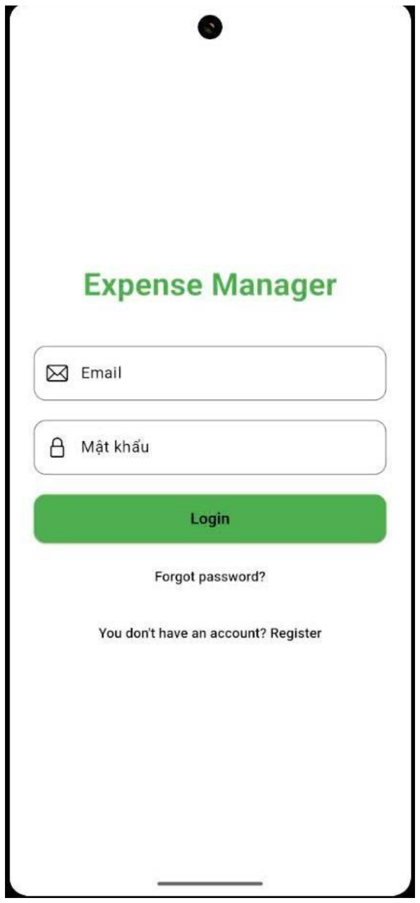
Libraries/Plugins Used:
http: Sends API requests to PocketBase. **flutter_secure_storage:** Stores authentication tokens securely. **Provider:** Manages user authentication state**.**

State Management Solution – UserProvider:
The UserProvider class is implemented using ChangeNotifier, allowing the app to manage the user's authentication state efficiently.

How it Works:
The **_user** variable holds the currently authenticated user. **setUser(User user):** Updates the _user object and notifies all listeners (e.g., UI updates).
clearUser(): Resets _user to null when the user logs out, triggering UI updates.

Code Architecture:
UserProvider is registered at the app's root (MultiProvider in main.dart)
Other widgets can access UserProvider using Provider.of<UserProvider>(context).
This ensures a **global user state** across the app.

2. Feature / Application page 2: Login



The User Registration feature allows new users to create an account by entering their personal details such as name, email, phone, address, country, and password. The app securely stores this information in a PocketBase database, enabling authentication and user-specific data management.

Widgets Used:
TextFormField: Input fields for name, email, phone, address, country, password, and password confirmation.
ElevatedButton: Button for submitting the registration form.
ScaffoldMessenger: Displays success or error messages.
CircularProgressIndicator: Shows a loading indicator when the request is processing.
Form: used for form validation and submission.

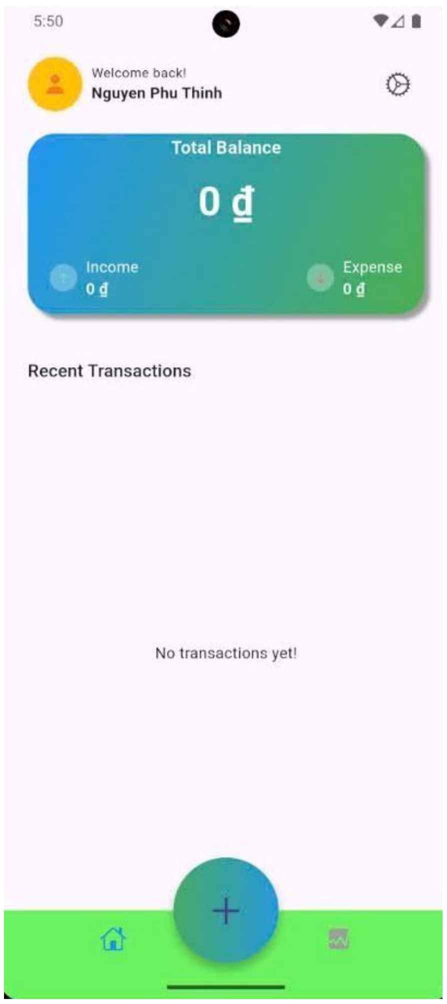
Libraries/Plugins Used:
http: Sends API requests to PocketBase. **flutter_secure_storage**: Stores authentication tokens securely. **Provider**: Manages user authentication state
State Management Solution – UserProvider

The UserProvider class is implemented using ChangeNotifier, allowing the app to manage the user's authentication state efficiently.

How it Works:
The **_user** variable holds the currently authenticated user. **setUser(User user):** Updates the **_user** object and notifies all listeners (e.g., UI updates).
clearUser(): Resets **_user** to **null** when the user logs out, triggering UI updates.

Code Architecture:
UserProvider is registered at the app's root (MultiProvider in main.dart). Other widgets can access UserProvider using Provider.of<UserProvider>(context) This ensures a global user state across the app.

3. Feature / Application page 3: Home page



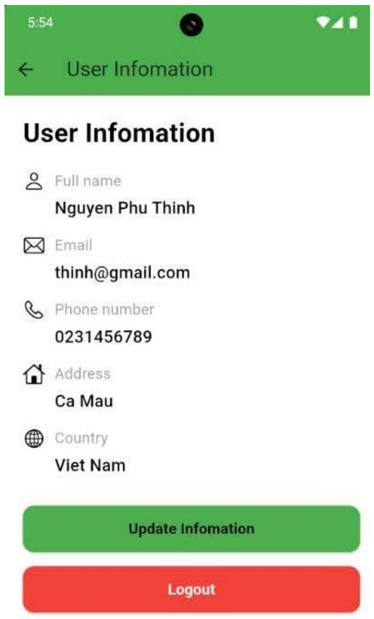
The Home Screen is the main dashboard of the expense tracking application. It provides an overview of the user's financial summary, including total balance, income, and expenses. The screen also displays recent transactions and allows users to navigate to different sections of the app**.**

Implementation Details:
Widgets Used:
Scaffold: Provides the main structure of the screen.
SafeArea: Ensures UI elements do not overlap with system status bar.
Column & Row: Used for layout structuring.
CircleAvatar: Displays user profile icon.
Text: Displays user name and financial information.
Container: Used for styling the balance card.
ListView: Displays recent transactions.
BottomNavigationBar: Provides navigation options.
FloatingActionButton: Triggers the add transaction modal.
ListTile: Used in the modal for selecting transaction type.
Gradient Container: Uses a gradient background for the balance card. **ClipRect:** Used for rounded corners in UI elements.

Libraries & Plugins Used
Provider: Manages the state of income and expense transactions. **intl:** Used for currency formatting and date formatting. **cupertino_icons:** Provides iOS-style icons**.**

State Management Solution
This feature uses the Provider package for state management:
ExpenseProvider: Manages expense transactions.
IncomeProvider: Manages income transactions.
Each provider maintains a list of transactions and notifies listeners whenever changes occur.

4. **Feature / Application page 4: View user information**



User Infomation

Full name
Nguyen Phu Thinh

Email
thinh@gmail.com

Phone number
0231456789

Address
Ca Mau

Country
Viet Nam

The "User Information" page in this application is used to display and manage the user's profile details, including full name, email, phone number, address, and country. The page also provides options to update the user's information and log out from the application.

Implementation Details:

Widgets Used:

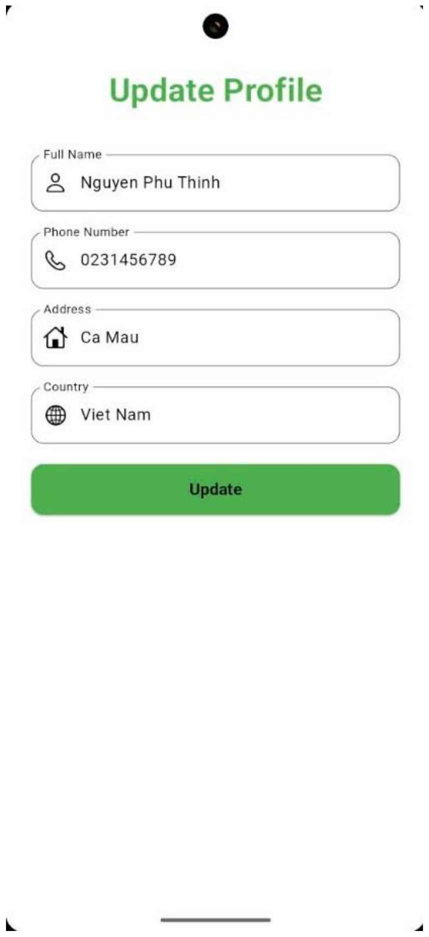
- **Scaffold:** Provides the structure for the screen.
- **AppBar:** Displays the title "User Information" and a back button.
- **Column:** Arranges user details vertically.
- **ListTile:** Displays each user detail with an icon and label.
- **ElevatedButton:** Used for "Update Information" and "Logout" actions.
- **Icon:** Represents different user attributes (e.g., email, phone, address).

Libraries/Plugins Used:

- **Provider**: Used for state management.
- **flutter/material.dart:** Provides core UI components.

- **State Management Solution:**
- The feature uses Provider for managing user state.
- UserProvider holds user details and updates state when information is modified.
- The UI listens to UserProvider using Consumer<UserProvider>.

5. Feature / Application page 5: Update Profile


Update Profile

Full Name: Nguyen Phu Thinh

Phone Number: 0231456789

Address: Ca Mau

Country: Viet Nam

The "Update Profile" feature allows users to edit and update their personal information, including their full name, phone number, address, and country. This feature is essential for keeping user data up-to-date and ensuring accurate information is stored within the system.

Implementation details:

Widgets Used:
Scaffold for the page layout.

SafeArea to ensure UI elements are positioned correctly on different devices. **Column** and **SingleChildScrollView** for structuring the form **TextFormField** for user input fields with initial values.

ElevatedButton for the update action.

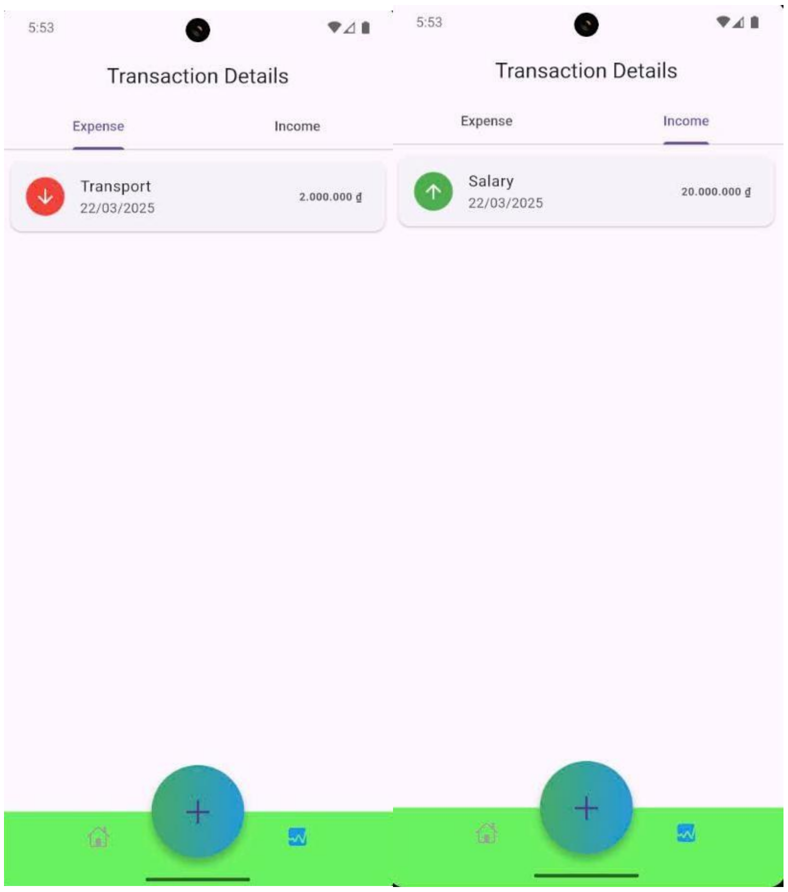
SnackBar for displaying success or error messages.
CircularProgressIndicator for showing a loading state during updates.
CupertinoIcons for better UI aesthetics with icons.

Libraries/Plugins Used:
flutter/cupertino.dart for additional UI components (icons).
flutter/material.dart for standard Flutter UI components.

State Management:
This feature uses internal state management (**setState**) within
_UpdateScreenState to track user input changes and loading state.
The **AuthService** class is responsible for handling updates to user information.

Data Handling:
Data Source: User information is retrieved from an external authentication system and displayed in the form.
Data Update: When users submit the form, the updated data is sent to AuthService.updateUser() for remote storage.

6. Feature / Application page 6: View Transaction



The Transaction Detail page is designed to display details of individual transactions, including both expenses and income. This page allows users to review, edit, or delete specific transactions.

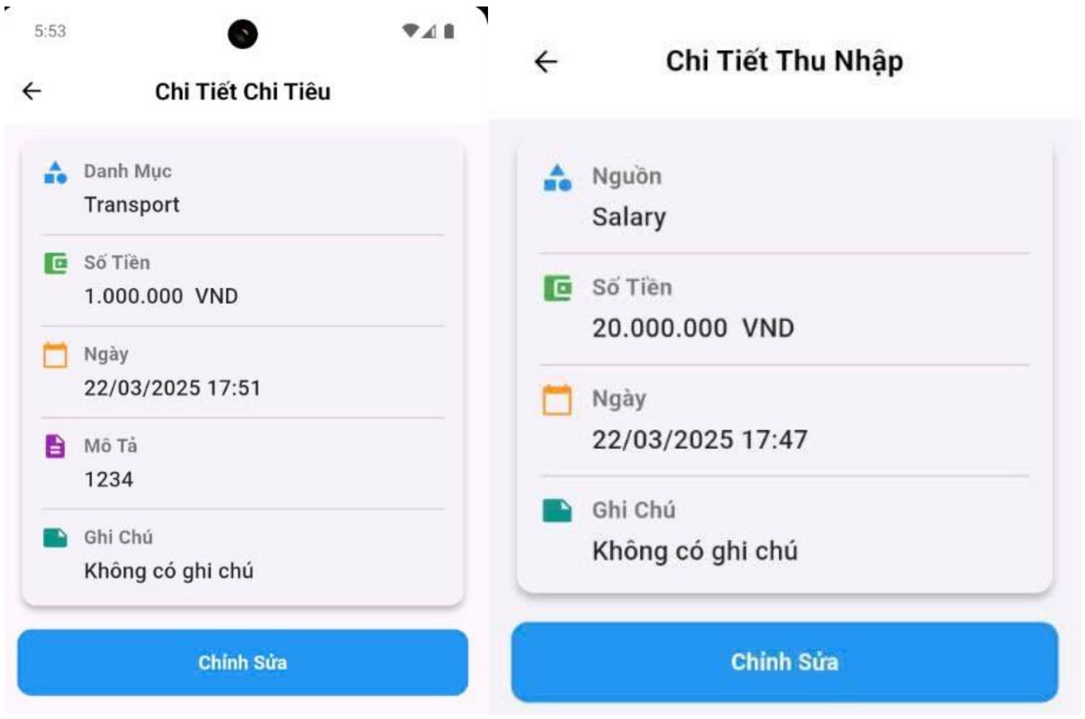
Widgets Used:
Scaffold: Provides the basic structure of the page.
AppBar: Displays the page title and back button.
TabBar & TabBarView: Allows switching between Expense and Income transactions.
ListView: Displays the list of transaction details.
Card: Used to present each transaction in a visually distinct format. **Text & TextStyle:** Displays transaction details such as amount, date, and category.
FloatingActionButton: Provides options to edit or delete a transaction.
IconButton: Used for delete and edit actions.
Consumer (from Provider package): Used for state management to retrieve transaction details.

ConfirmDialog (Custom Dialog Widget): Used to confirm deletion before removing a transaction.

Libraries & Plugins Used
Provider: Manages the state of transactions.
Intl: Formats date and currency values.
PocketBase (for remote storage, if applicable): Retrieves transactions from a backend service.

State Management Solution
This feature uses Provider for state management. The structure is as follows:
ExpenseProvider: Manages the list of expenses.
IncomeProvider: Manages the list of incomes.
The **TransactionDetailScreen** listens to these providers using **Consumer** to get the latest transaction data.

7. Feature / Application page 7: Expense and Income Details


5:53

Chi Tiết Chi Tiêu

Danh Mục
Transport

Số Tiền
1.000.000 VND

Ngày
22/03/2025 17:51

Mô Tả
1234

Ghi Chú
Không có ghi chú

Chỉnh Sửa

Chi Tiết Thu Nhập

Nguồn
Salary

Số Tiền
20.000.000 VND

Ngày
22/03/2025 17:47

Ghi Chú
Không có ghi chú

Chỉnh Sửa

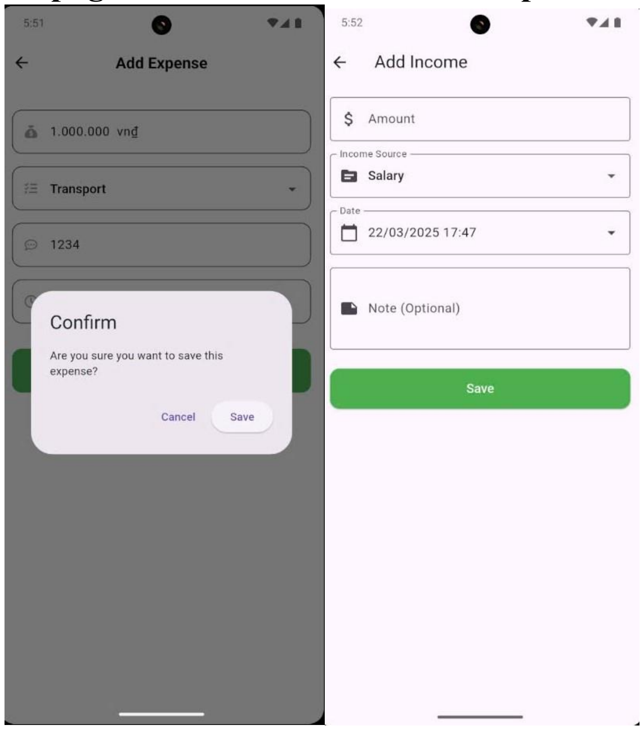
This screen displays detailed information about a specific income transaction. Users can view the source of income and expense, the amount, the date, and any notes related to the transaction. Additionally, users can navigate to the edit screen by clicking the **Edit** button.

Widgets Used: Column, Row, Expanded, SizedBox, Card, Divider, Icon, Text, TextButton, and Padding.
SingleChildScrollView to handle scrolling.
DraggableScrollableSheet for displaying the detail screen as a bottom sheet.

Libraries/Plugins Used:
Intl: Used for formatting dates (DateFormat) and currency (NumberFormat)
flutter/material.dart: Standard Flutter UI components.

State Management:
The screen does not actively manage state but relies on Navigator.pop(context, true) to return a value when the income is edited
When the Edit button is clicked, the screen navigates to the EditIncome and Edit Expense screen. If the edit is successful, it pops back with true to refresh the data.

8. Feature / Application page 8: Add income and add Expense



The Add Expense and Add Income screens allow users to record their financial transactions. These pages enable users to enter necessary details such as amount, category/source, date, and notes. The purpose of these screens is to facilitate easy tracking of income and expenses within the application.

Widgets Used:

Both screens share a similar structure and use the following Flutter widgets: •

TextField: For user input (amount, note)

- **DropdownButton:** To select the category (Expense) or source (Income)
- **DatePicker**: For selecting the transaction date
- **ElevatedButton:** To submit the form
- **Consumer** (from Provider package): To interact with the state management solution

Libraries & Plugins

- **provider**: For state management
- **intl**: To format dates correctly
- **flutter/material.dart**: Core UI components

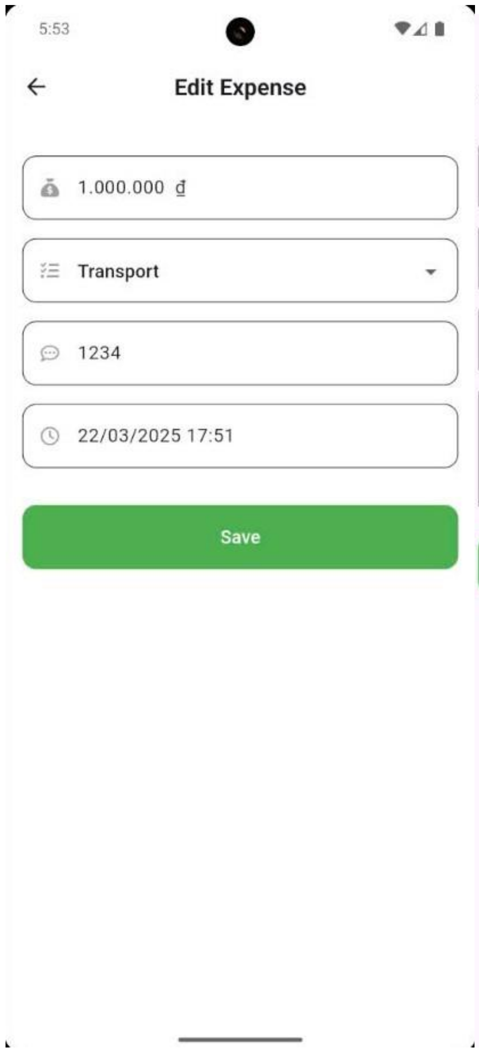
State Management

State management for these screens is handled using **Provider**:

- ExpenseProvider manages the state for expenses.
- IncomeProvider manages the state for income.
- When a user submits a transaction, the respective provider updates the app state and notifies listeners.

Data Storage: PocketBase is used to store finalized transactions.

9. Feature / Application page 9: Edit income and Edit Expense


5:53

Edit Expense

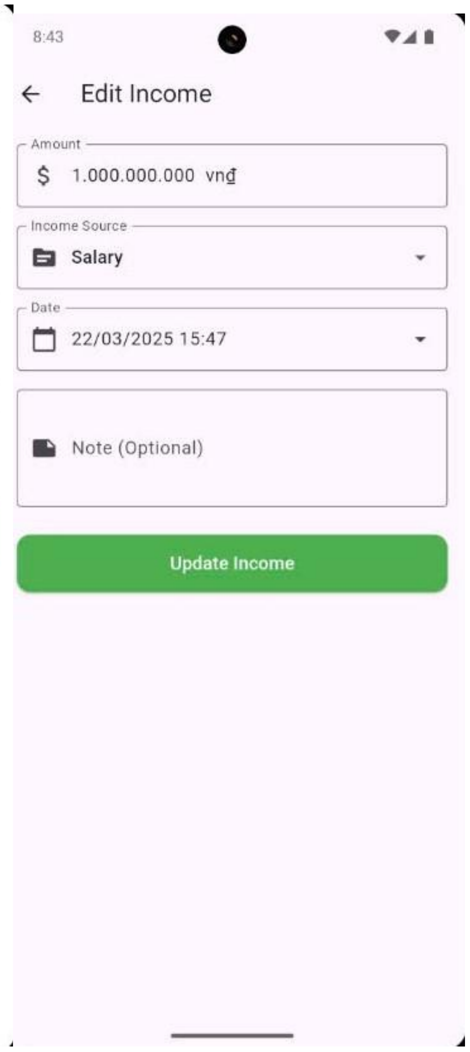
1.000.000 ₫

Transport

1234

22/03/2025 17:51

Save


8:43

Edit Income

Amount: $ 1.000.000.000 vnđ

Income Source: Salary

Date: 22/03/2025 15:47

Note (Optional)

Update Income

The Edit Expense and Edit Income features allow users to modify existing expense and income records in the application. These features provide a user-friendly interface where users can update transaction details such as amount, category, date, and notes.

Implementation Details

Widgets Used

- TextFormField: Used for input fields (amount, notes, etc.).
- DropdownButtonFormField: Used for selecting categories (expense categories or income sources).
- DatePicker: Used for selecting the transaction date.
- ElevatedButton: Used for the save/update button.

- Consumer (from Provider package): Used for accessing state management.
- Scaffold, Column, Padding, SizedBox, etc., for UI layout.
- showDatePicker: A Flutter built-in function for date selection.
- DropdownButtonFormField: Used to create dropdown selections for categories and income sources.

Libraries/Plugins Used

- provider: Used for state management of expense and income data.
- intl: Used for date formatting.

State Management Solution

This feature uses **Provider** for state management.

- ExpenseProvider: Manages expense-related state.
- IncomeProvider: Manages income-related state.
- When the edit screen is opened, it fetches the selected expense/income data from the provider and pre-fills the form fields.
- When the user updates a record, the provider updates the respective entry in the SQLite database and refreshes the transaction list.

Data Storage: The transaction data is stored and updated in Pocketbase