Mobile Application Project Documentation

1. Title Page

Project Title: Expense Tracker Mobile Application
Student Name: Hamza Bander Ali Mahdi
University ID: 2023030524
Department: Artificial Intelligence
Course: Mobile Application Development
Submission Date: 31 January 2026

---

2. Abstract

This project presents a Flutter-based mobile application designed to help users manage and track their daily expenses efficiently. The application focuses on simplicity, usability, and offline functionality by relying entirely on local storage. Through structured state management and a clean architectural approach, the app allows users to add, edit, review, and delete expenses while maintaining data integrity and a smooth user experience.

---

3. Introduction

Personal expense management is an essential aspect of daily life. Many users require a lightweight and offline solution that allows them to track their spending without relying on internet connectivity. The Expense Tracker application addresses this need by providing a fully local mobile solution built using Flutter.

The main objectives of this project are:

- Applying Flutter development best practices
- Implementing proper state management
- Using local databases effectively
- Structuring code in a clean and maintainable way
- Delivering a complete academic mobile application

---

4. Application Overview

The Expense Tracker application allows users to:

- Log in locally
- Add, edit, and delete expenses
- View expenses in organized lists
- Filter expenses by date ranges
- Manage personal profile information
- Store all data locally on the device

The application works entirely offline and does not depend on external servers or cloud services.

---

5. State Management Explanation

The application uses **Provider** as the state management solution.

Reasons for choosing Provider:

- Simple and lightweight
- Easy integration with Flutter widgets
- Clear separation between UI and business logic
- Improves performance by minimizing unnecessary rebuilds

Two main providers are implemented:

- **ExpenseProvider**: Manages expense-related operations such as adding, updating, deleting, and retrieving expenses from the database.
- **ProfileProvider**: Manages user profile data including username and profile image.

---

6. Navigation & Screens

The application consists of multiple screens with proper navigation flow:

- **Splash Screen**: Displays the application logo while checking initial app state.
- **Login Screen**: Handles local user login.
- **Home Screen**: Displays a summary dashboard of expenses.
- **Expenses List Screen**: Shows a detailed list of recorded expenses with filtering options.
- **Add Expense Screen**: Allows users to add or edit expense entries.
- **Settings Screen**: Enables profile management, logout, and account deletion.

Navigation is handled using Flutter's built-in navigation system.

---

7. Data Storage & Database Design

Local Storage

The application uses two forms of local storage:

- **SharedPreferences**
  - Stores simple user-related data such as login status and profile information.
- **Sqflite (SQLite Database)**
  - Stores expense records persistently.

Database Structure

The expense table includes:

- Expense ID
- Title
- Amount
- Date
- Category

CRUD operations are implemented through a dedicated database helper class.

---

8. Project Structure

The project follows a well-organized and layered folder structure derived directly from the actual source files and their import statements. The structure reflects a clear separation between UI, logic (state management), and data layers, with further subdivision inside each main folder.

Detailed Folder Structure

```
lib/
|— main.dart                // Application entry point
|
├— data/
|   ├— database/
|   |   └— db_helper.dart    // SQLite database helper and CRUD operations
|   |
|   └— models/
|       └— expense_model.dart // Expense data model
|
├— logic/
|   └— providers/
|       ├— expense_provider.dart  // Expense state management
|       └— profile_provider.dart  // User profile state management
|
├— ui/
|   ├— screens/
|   |   ├— splash_screen.dart
|   |   ├— login_screen.dart
|   |   ├— home_screen.dart
|   |   ├— expenses_list_screen.dart
|   |   ├— add_expense_screen.dart
|   |   └— settings_screen.dart
|   |
|   └— helpers/
|       └— common_ui_helpers.dart // Reusable UI components and helpers
```

Structure Explanation

- **main.dart**: Initializes the application, providers, and root widget.
- **data layer**:
  - models/: Contains data models used across the app.
  - database/: Handles local SQLite database operations.
- **logic layer**:
  - providers/: Contains Provider classes responsible for business logic and state management.
- **ui layer**:
  - screens/: Contains all application screens and pages.
  - helpers/: Contains reusable UI utilities and helper widgets.

This hierarchical structure improves readability, scalability, and maintainability, and clearly reflects the architectural decisions used in the project.

---

9. Input Validation & User Feedback

The application includes:

- Validation for empty fields and invalid inputs
- Confirmation dialogs for critical actions
- Loading indicators where applicable

These features enhance usability and prevent incorrect data entry.

---

10. Security Considerations

- All data is stored locally on the user's device
- No sensitive information is transmitted externally
- Users can permanently delete their data
- Proper separation between UI and data layers reduces logical vulnerabilities

---

11. Results

The application successfully delivers:

- A fully functional offline expense tracking system
- Clean UI and smooth navigation
- Persistent local data storage
- Organized and scalable code structure

---

12. Future Work & Recommendations

Possible future enhancements include:

- Cloud synchronization
- Data encryption
- Expense analytics and charts
- Exporting reports (PDF / Excel)
- AI-based expense categorization

---

13. Conclusion

This project demonstrates the practical application of Flutter development concepts, including state management, local storage, and clean architecture. The Expense Tracker application meets all technical and documentation requirements and serves as a strong foundation for future expansion.

---

14. References

- Flutter Official Documentation
- Dart Language Documentation
- SQLite Documentation
- Course Materials
- AI tools (used for understanding and analysis only)