DIGT2107: Practice of Software Development

Project Iteration 1.2:

Detailed User Stories and Requirements

Project Name:

SmartBudget – Personal Finance Tracker

Team Number:

Team #3

Team Members:

Muhammed Ahmed, Mohsen Pasdar, Alia Hagi-Dhaffe

Date:

October 05, 2025

Table of Contents

1 Introduction

This document defines the **requirements**, **use cases**, and **user stories** for SmartBudget, a desktop, offline-first personal finance tracker. It focuses on a realistic MVP for two semesters, written in clear, user-centric language and aligned with the object-oriented design practices covered in class.

2 Requirements

2.1 Functional Requirements

- **FR1 – Transactions:** The system must let the user record, edit, and delete income and expense transactions (amount, date, category, optional note), and persist them locally so they remain available across sessions.
- **FR2 – Browsing & Filters:** The system must let the user view transactions and apply filters (e.g., by date range and category) to review specific spending.
- **FR3 – Budgets & Alerts:** The system must let the user set a monthly spending limit per category and notify the user when spending reaches or exceeds that limit.
- **FR4 – Summaries & Charts:** The system must provide a period summary (income, expenses, balance) and at least one visual breakdown of spending by category.

2.2 Non-Functional Requirements

- **Usability**: The system should present a simple, consistent JavaFX interface with clear labels, readable fonts, and sensible navigation between Transactions, Budgets, and Reports.

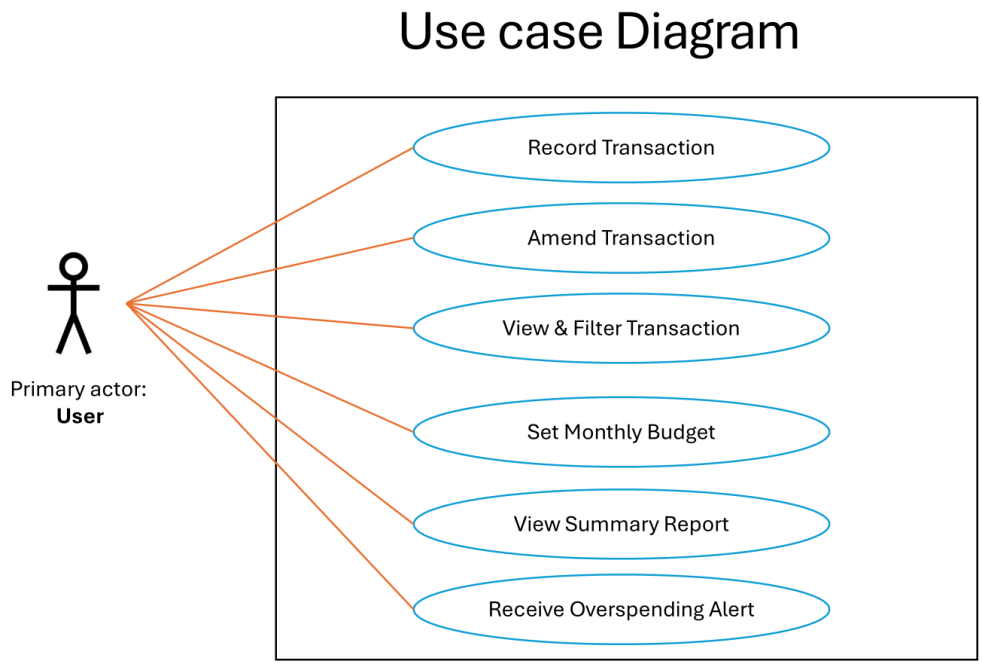
- **Performance**: The system should load typical datasets quickly and remain responsive during common operations.
- **Reliability**: The system should save changes immediately to local storage to minimize data loss and handle invalid inputs gracefully with clear feedback.
- **Privacy/Offline**: The system should operate fully offline and avoid sending data over a network.
- **Portability**: The system should run on Windows, macOS, and Linux with a supported Java runtime.
- **Maintainability**: The system should adopt a modular structure (e.g., MVC) and include unit tests for core business logic to enable straightforward updates.

3 Use Cases

3.1 Actors

- **User** – an individual tracking personal finance.

3.2 Use-Case Diagram



3.3 Use-Case Descriptions

3.3.1 UC-1: Record Transaction

- **Actor**: User
- **Goal**: Add a new income or expense.
- **Preconditions**: App is running.
- **Main Flow**: (1) User initiates adding a transaction. (2) System asks for type, amount, date, category, optional note. (3) User confirms. (4) System saves and updates totals.
- **Alternates**: Missing/invalid fields → system requests correction.
- **Postconditions**: Transaction is persisted; summaries reflect it.

3.3.2 UC-2: Amend Transaction

- **Actor**: User
- **Goal**: Modify or delete an existing entry.
- **Preconditions**: At least one transaction exists.

- **Main Flow**: (1) User selects a transaction. (2) System shows details. (3) User edits or deletes. (4) System saves and refreshes totals.
- **Alternates**: Entry not found → system informs user.
- **Postconditions**: History and aggregates reflect the change.

3.3.3 UC-3: View & Filter Transactions

- **Actor**: User
- **Goal**: Review past entries by period/category.
- **Main Flow**: (1) User requests transaction list. (2) System displays entries. (3) User applies filters. (4) System narrows results and updates totals for the selection.
- **Alternates**: No matches → show empty results and zero totals.

3.3.4 UC-4: Set Monthly Budget

- Actor: User
- **Goal**: Define or adjust a per-category spending limit for the current month.
- **Main Flow**: (1) User selects category. (2) System requests limit. (3) User confirms. (4) System stores limit and shows remaining allowance.
- **Alternates**: Invalid limit → system asks for a valid amount.
- **Postconditions**: Budgets available to alerts and reports.

3.3.5 UC-5: View Summary Report

- **Actor**: User
- **Goal**: See totals and category breakdowns for a period.
- **Main Flow**: (1) User selects a period. (2) System computes totals and breakdown. (3) System presents figures and at least one chart.
- **Alternates**: No data in period → system shows zeros and a note.

3.3.6 UC-6: Receive Overspending Alert

- **Actor**: User
- **Goal**: Be notified when a category exceeds its limit.
- **Preconditions**: At least one budget is defined.
- **Main Flow**: (1) System compares spending to limits as transactions are added/viewed. (2) System signals when a limit is reached or exceeded.
- **Alternates**: No budgets defined → no alert.
- Postconditions: User is aware of budget status.

4 User Stories

4.1 User Story 1: Record a Transaction

- **Description**: *As a user, I want to record an expense or income so that my daily finances are accurately tracked.*
- **Traceability**: FR1, NFR – Reliability, Usability
- **Preconditions:** The application is open and the user has access to the "Add Transaction" feature.
- **Acceptance Criteria:**
  - The user can enter transaction details (amount, type, date, and category).
  - The transaction is saved immediately after confirmation and appears in the list of transactions.
  - The total income/expense summary updates automatically.
  - The system provides feedback confirming successful entry.
- **Postconditions:** Transaction is stored locally and persists when the application restarts.
- **Priority:** High

4.2 User Story 2: Edit or Delete a Transaction

- **Description**: *As a user, I want to edit or delete a transaction so that my financial records stay accurate.*
- **Traceability:** FR1, NFR – Reliability, Usability
- **Preconditions:** At least one transaction exists in the system.
- **Acceptance Criteria:**
  - The user can open an existing transaction and modify any editable field.
  - The user can delete an entry after a confirmation prompt.
  - Updated or deleted transactions reflect immediately in totals and summaries.
  - The system prevents accidental deletion through a confirmation step.
- **Postconditions:** Transaction list and totals are updated to reflect the change.
- **Priority:** High

4.3 User Story 3: View and Filter Transactions

- **Description**: *As a user, I want to view and filter transactions by date and category so that I can easily review my spending history.*
- **Traceability:** FR2, NFR – Usability, Performance
- **Acceptance Criteria:**
  - The user can view all past transactions in a table or list.
  - The user can filter transactions by category, date range, or keyword.
  - The filtered list updates immediately after criteria are applied.
  - The user can clear filters to return to the full view.

- **Postconditions:** Filtered or full lists remain visible until the user changes the view.
- **Priority:** High

4.4 User Story 4: Set a Monthly Budget

- **Description**: *As a user, I want to set a monthly budget for each spending category so that I can manage my expenses more effectively.*
- **Traceability:** FR3, NFR – Usability, Reliability
- **Preconditions:** User has at least one spending category defined.
- **Acceptance Criteria:**
  - The user can set or update a monthly spending limit for a selected category.
  - The budget amount is saved and displayed for the current month.
  - The system visually shows remaining budget as transactions are added.
  - Invalid inputs (e.g., negative values or empty fields) are rejected with feedback.
- **Postconditions:** Budget is stored and available for alert and reporting features.
- **Priority:** Medium

4.5 User Story 5: Receive Overspending Alerts

- **Description**: *As a user, I want to receive an alert when spending exceeds my budget so that I can adjust my spending habits early.*
- **Traceability:** FR3, NFR – Reliability, Usability
- **Preconditions:** At least one active budget is set for the month.
- **Acceptance Criteria:**

- The system checks total spending against each category limit.
- When spending meets or exceeds the limit, the system displays a clear visual alert.
- The alert indicates the category and amount over budget.
- The alert remains visible until acknowledged or budget is reset.

- **Postconditions:** User is aware of overspending status and can adjust budgets accordingly.
- **Priority:** Medium

4.6 User Story 6: View Monthly Summary

- **Description**: *As a user, I want to view a summary of my income, expenses, and remaining balance so that I can assess my financial status.*
- **Traceability:** FR4, NFR – Usability, Performance
- **Acceptance Criteria:**
  - The user can select a reporting period (e.g., month or custom range).
  - The system generates totals for income, expenses, and balance.
  - The summary is displayed in an easy-to-read format.
  - When no data is available, a clear "no records found" message is shown.
- **Postconditions:** Generated summaries can be revisited without re-entry until the user changes the period.
- **Priority:** High

4.7 User Story 7: View Spending Chart

- **Description**: *As a user, I want to see a chart of my spending by category so that I can identify trends and make informed financial decisions.*
- **Traceability:** FR4, NFR – Usability, Performance
- **Acceptance Criteria:**
  - The system presents at least one visual chart (pie or bar) showing spending by category.
  - Charts update automatically when transactions or filters change.
  - The visualization is easy to interpret with clear labels and legends.
- **Postconditions:** Chart view remains synchronized with current transaction data.
- **Priority:** Low

4.8 User Story 8: Load Saved Data on Startup

- **Description**: *As a user, I want my saved transactions and budgets to load automatically when I open the app so that I can continue seamlessly.*
- **Traceability:** FR1, FR3, NFR – Reliability, Performance
- **Preconditions:** Previously saved data exists locally.
- **Acceptance Criteria:**
  - The system loads all stored transactions and budgets automatically at startup.
  - The system notifies the user if no data is found.
  - Loaded data appears accurately within corresponding views (Transactions, Budgets, Reports).

- **Postconditions:** User resumes from previous state with no manual loading required.
- **Priority:** High

5 Iteration Plan and Backlog

The SmartBudget project will be developed through multiple iterations across the Fall 2025 and Winter 2026 terms. Each iteration builds upon previous work, aligning with Agile and incremental development principles.

5.1 Iteration Plan

5.1.1 Iteration 1 (Weeks 1–2 – Vision and Planning)

- Deliverable 1.1 – Vision Document: Submitted.

5.1.2 Iteration 2 (Weeks 3–5 – Requirements and Use Cases)

- Deliverable 1.2 – Requirements, Use Cases, and User Stories: Current deliverable.

5.1.3 Iteration 3 (Weeks 6–9 – Initial Development and Testing)

- Deliverable 2.1 – Testing Document: Unit testing setup, prototype development, and initial test results.

5.1.4 Iteration 4 (Weeks 10–12 – Mid-Project Peer Review)

- Deliverable 2.2 – Full Documentation, Prototype, and Presentation

5.1.5 Iteration 5 (Weeks 1–2 – Advanced Software Design)

- Deliverable 3.1 – Design Documentation: Class, sequence, and activity diagrams.

5.1.6 Iteration 6 (Weeks 3–4 – GUI Development and Testing)

- Deliverable 3.1 (Update): GUI implementation and initial testing.

5.1.7 Iteration 7 (Weeks 6–9 – Refactoring and Implementation)

- Deliverable 3.2 – Completed Implementation: Refactored, optimized codebase with all core features finalized.

5.1.8 Iteration 8 (Weeks 10–11 – Final Testing and Documentation)

- Deliverable 3.3 – Final Documentation Preparation

5.1.9 Iteration 9 (Week 12 – Block Week – Final Presentation and Submission)

- Deliverable 3.3 – Final Project Submission

5.2 Updated Backlog

- **User Story 1** – Record a Transaction (Core functionality: enables basic data entry.)
- **User Story 2** – Edit or Delete a Transaction (Ensures data accuracy and integrity.)

- **User Story 3** – View and Filter Transactions (Viewing is essential for MVP; filtering can be expanded later.)
- **User Story 8** – Load Saved Data on Startup (Allows continuity between sessions.)
- **User Story 6** – View Monthly Summary (Provides immediate feedback on totals and balances.)
- **User Story 4** – Set a Monthly Budget (Introduces budget tracking.)
- **User Story 5** – Receive Overspending Alerts (Enhances awareness and control over spending.)
- **User Story 7** – View Spending Chart (Adds visualization for deeper insights.)

6 Conclusion

This deliverable defines a clear, achievable MVP for ***SmartBudget*** that emphasizes user-centred design, minimal complexity, and iterative refinement.

The scope—transactions, filtering, budgeting, alerts, and summaries—is well balanced for two semesters and aligns with Agile best practices and object-oriented design foundations.