DIGT2107: Practice of Software Development

Project Deliverable 2.2:

Mid Project Report Submission

Project Name:

SmartBudget – Personal Finance Tracker

Team Number:

Team #3

Team Members:

Mohsen Pasdar, Muhammed Ahmed, Alia Hagi-Dhaffe

Date:

November 30, 2025

Table of Contents

1 Introduction

This document represents Deliverable 2.2 for the DIGT2107: Practice of Software Development course. As part of Iteration 2.2, we bring together the updated requirements, the complete testing documentation, the full user manual, and the overview of the SmartBudget prototype. This deliverable builds on the earlier work completed in Deliverables 1.1, 1.2, and 2.1, where we defined the project vision, described the system requirements, created detailed use cases and user stories, and prepared the initial testing plan.

The main goal of this deliverable is to present a clear and organized view of the project at the mid-point of development. We refined the functional and non-functional requirements based on feedback, improved the use cases, and expanded the testing documentation with full traceability. This ensures that each requirement is linked to a corresponding test case and prepared for validation once implementation begins.

We also included the UX/UI prototype created in Figma. This prototype is not a Java desktop application and does not include any backend logic. Instead, it serves as a visual guide for how the final system should look and behave. It shows the layout, navigation structure, and user flows for key features such as recording transactions, managing budgets, and viewing reports. This design will guide the upcoming development work and help keep the implementation aligned with the requirements.

This document is structured to show the progress made up to this point. It begins with the updated requirements and use cases, followed by the complete testing document. It then presents the user manual and the prototype overview. Together, these components form a solid foundation for the next development stages, where we will begin implementing the backend, JavaFX interface, building core features, and executing test cases as the system evolves.

2 Updated 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.

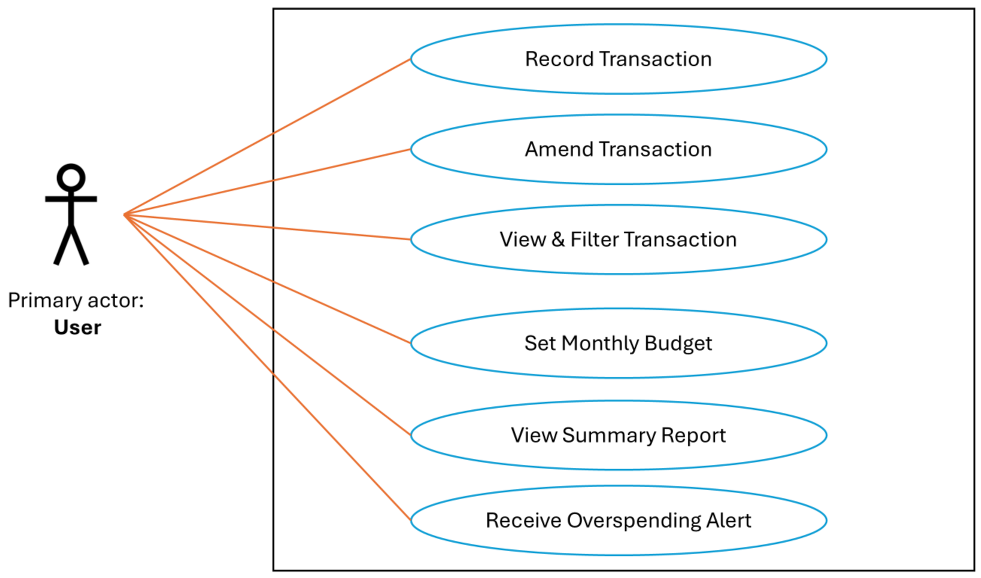
3 Updated Use Cases

3.1 Actors

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

3.2 Use-Case Diagram

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 Updated 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 Test Plan

5.1 Testing Objectives and Scope

The main goal of this testing phase is to make sure that each part of the SmartBudget system works as expected and meets the requirements defined in earlier deliverables. In this iteration, the focus is on preparing the groundwork for testing by defining clear test cases and linking them to each requirement. The testing will mainly cover the system's core functions such as recording transactions, filtering data, setting budgets, and viewing summaries. Although the tests are not yet executed, this plan ensures that every feature is ready for validation in later stages of development and that the testing process will be organized, consistent, and easy to track.

5.2 Testing Approach (Unit Testing Focus for Iteration 3)

The testing approach for this iteration follows a unit testing strategy, where each function or module of the SmartBudget system will be tested individually. This helps identify issues early and ensures that small parts of the program work correctly before being combined into a larger system. The testing plan focuses on designing tests using JUnit in Java to verify calculations, data handling, and interface responses. Each test case will check one small piece of functionality based on the requirements and user stories. Future iterations will expand this to include integration and GUI testing once the main components are implemented.

5.3 Requirement-to-Test Case Traceability Matrix

This section shows how each system requirement is linked to one or more test cases. The traceability matrix helps make sure that all functional and non-functional requirements are covered by at least one planned test. It provides a clear view of what will be tested, how it will be verified, and ensures that no part of the system is left untested. This also helps track future changes and maintain consistency between requirements and test documentation.

Table 1: Requirement-to-Test Case Traceability Matrix

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
FR-001	Record, edit, delete income/expense transactions; persist locally across sessions	TC-001	Add expense with valid inputs (amount, date, category, optional note)	
FR-001	— same as above —	TC-002	Add income with valid inputs	
FR-001	— same as above —	TC-003	Reject invalid amount (non-numeric / negative)	
FR-001	— same as above —	TC-004	Reject missing required fields (amount, date, type) with clear feedback	
FR-001	— same as above —	TC-005	Edit an existing transaction and verify totals update	

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
FR-001	— same as above —	TC-006	Delete transaction with confirmation prompt; totals update	
FR-001	— same as above —	TC-007	Cancel delete preserves the transaction	
FR-001	— same as above —	TC-008	Data persists across app restart (transactions remain available)	
FR-002	View transactions and apply filters (date range, category)	TC-009	View all transactions in list/table	
FR-002	— same as above —	TC-010	Filter by category (single category)	
FR-002	— same as above —	TC-011	Filter by date range (start–end)	
FR-002	— same as above —	TC-012	Combined filters (date range + category) narrow results and update totals	
FR-002	— same as above —	TC-013	No matches → empty results + zero totals shown	
FR-002	— same as above —	TC-014	Clear filters restores full transaction list	
FR-002	— same as above —	TC-015	Keyword search narrows list	
FR-003	Set monthly spending limit per category and alert on reaching/exceeding	TC-016	Set monthly budget (valid positive amount)	
FR-003	— same as above —	TC-017	Reject invalid budget inputs (negative/blank) with feedback	
FR-003	— same as above —	TC-018	Update an existing budget; remaining allowance recalculates	
FR-003	— same as above —	TC-019	Alert triggers when spending equals the limit	
FR-003	— same as above —	TC-020	Alert triggers when spending exceeds the limit; shows overage amount	

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
FR-003	— same as above —	TC-021	No budgets defined → no alert appears	
FR-004	Provide period summary (income, expenses, balance) + at least one chart	TC-022	Monthly summary totals compute correctly (income, expenses, balance)	
FR-004	— same as above —	TC-023	Custom range summary computes correctly	
FR-004	— same as above —	TC-024	No data in selected period → zeros + “no records found” note	
FR-004	— same as above —	TC-025	Category chart (pie/bar) reflects data accurately with clear labels	
FR-004	— same as above —	TC-026	Chart and summary auto-update when transactions or filters change	
NFR-Usability	Simple, consistent JavaFX UI; clear labels; readable fonts; sensible navigation	TC-027	Heuristic usability review of labels, font sizes, and navigation between Transactions/Budgets/Reports	
NFR-Performance	Responsive under typical datasets; quick loads	TC-028	Load 1–5k transactions; ensure UI remains responsive (no noticeable lag on list/filter)	
NFR-Reliability	Immediate save to local storage; graceful handling of invalid input with clear feedback	TC-029	Immediate persistence after create/edit/delete; verify storage writes and recovery on restart	
NFR-Privacy/Offline	Full offline operation; no network transmission	TC-030	Run with network disabled; verify all functions work and no network calls are made	
NFR-Portability	Runs on Windows, macOS, Linux (supported Java runtime)	TC-031	Smoke run on 3 platforms; application starts and core flows (add/view) work	

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
NFR-Maintainability	Modular (e.g., MVC) structure; unit tests for core business logic	TC-032	Project structure review + presence of unit tests for core model/services	

5.4 Mapping to Use Cases and User Stories

This section connects each test case to the related use cases and user stories. It ensures that all user actions and scenarios described by the users are properly represented in the testing plan. By mapping test cases to these elements, the document confirms that the system will be tested from both technical and user perspectives, covering real-life usage and expected workflows.

Table 2: Mapping Test Cases to Use Cases

Use Case ID	Use Case Goal	Test Case ID(s)	Coverage
UC-1: Record Transaction	Add income/expense and update totals	TC-001, TC-002, TC-003, TC-004, TC-022, TC-026	
UC-2: Amend Transaction	Modify or delete an existing entry; refresh totals	TC-005, TC-006, TC-007, TC-022, TC-026	
UC-3: View & Filter Transactions	Review by period/category; empty results path	TC-009, TC-010, TC-011, TC-012, TC-013, TC-014, TC-015	
UC-4: Set Monthly Budget	Define/adjust per-category limit	TC-016, TC-017, TC-018	
UC-5: View Summary Report	Select period; show totals and chart; no-data path	TC-022, TC-023, TC-024, TC-025	
UC-6: Receive Overspending Alert	Notify on reach/exceed budget; precondition = budgets defined	TC-019, TC-020, TC-021	

Table 3: Mapping Test Cases to User Stories

Use Story ID	Story Summary	Related FR	Test Case ID(s)	Coverage
US-1: Record a Transaction	Add expense or income; immediate save; totals update; success feedback	FR-001	TC-001, TC-002, TC-004, TC-022	
US-2: Edit or Delete a Transaction	Edit/delete; confirmation; totals/summaries update	FR-001	TC-005, TC-006, TC-007, TC-022	
US-3: View & Filter Transactions	View all; filter by category/date/keyword; clear filters	FR-002	TC-009, TC-010, TC-011, TC-012, TC-014, TC-015	
US-4: Set a Monthly Budget	Set/update budget; show remaining; reject invalid	FR-003	TC-016, TC-017, TC-018	
US-5: Receive Overspending Alerts	Trigger at ≥ limit; show category and overage; remains visible	FR-003	TC-019, TC-020, TC-021	
US-6: View Monthly Summary	Select period; totals computed; clear message for no data	FR-004	TC-022, TC-024	
US-7: View Spending Chart	Present chart; auto-updates; clear labels/legend	FR-004	TC-025, TC-026	
US-8: Load Saved Data on Startup	Auto-load transactions/budgets at startup; notify if none	FR-001 / FR-003	TC-008	

5.5 Detailed Test Case Descriptions

This section provides detailed descriptions of all planned test cases that will be used to verify the ***SmartBudget*** system. Each test case includes its ID, related requirement, preconditions, steps, and expected results. These details will guide future testing activities, making it easy for developers and testers to perform and track tests systematically once it is implemented.

5.5.1 TC-001

Table 4: TC-01 test case description

Test Case ID	**TC-01**
Title	Add Expense – Valid Inputs
Requirement	FR-001; UC-1; US-1
Preconditions	App is running; "Transactions" view available
Steps	1. Open Transactions; Click "Add Transaction"; Select Expense. 2. Enter valid amount (e.g., 25.50), pick a valid date, choose category (e.g., Food), add optional note. 3. Click "Save".
Expected Result	Expense row added; expense increases and balance totals decreases; success feedback; persisted.
Actual Result	
Status	
Priority	High

5.5.2 TC-002

Table 5: TC-02 test case description

Test Case ID	**TC-02**
Title	Add Income – Valid Inputs
Requirement	FR-001; UC-1; US-1
Preconditions	App is running; "Transactions" view available
Steps	1. Open Transactions; Click "Add Transaction"; Select Income. 2. Enter valid amount (e.g., 25.50), pick a valid date, choose category (e.g., Salary), add optional note. 3. Click "Save"
Expected Result	Income row added; income and balance totals increase; success feedback; persisted.
Actual Result	
Status	
Priority	High

5.5.3 TC-003

Table 6: TC-03 test case description

Test Case ID	**TC-03**
Title	Add Transaction – Reject Invalid Amount
Requirement	FR-001; UC-1; US-1
Preconditions	App is running; “Transactions” view available
Steps	1. Open Transactions; Click “Add Transaction”; Select Income/Expense. 2. Enter non-numeric (e.g., “abc”) or negative amount. 3. Click “Save”.
Expected Result	Save blocked; clear validation message; no record created; no totals change.
Actual Result	
Status	
Priority	High

5.5.4 TC-004

Table 7: TC-04 test case description

Test Case ID	**TC-04**
Title	Add Transaction – Reject Missing Required Fields
Requirement	FR-001; UC-1; US-1
Preconditions	App is running; “Transactions” view available
Steps	1. Open Transactions; Click “Add Transaction”; Select Income/Expense. 2. Leave amount or date or type empty. 3. Click “Save”.
Expected Result	Save blocked; specific error message; no changes persisted.
Actual Result	
Status	
Priority	High

5.5.5 TC-005

Table 8: TC-05 test case description

Test Case ID	**TC-05**
Title	Edit Transaction – Totals Recalculate
Requirement	FR-001; UC-2; US-2
Preconditions	At least one existing transaction
Steps	1. Open an existing transaction. 2. Change amount (e.g., 20 → 30). 3. Click “Save”.
Expected Result	Row updates; affected totals/summary recalculates; success feedback; persisted.
Actual Result	
Status	
Priority	High

5.5.6 TC-006

Table 9: TC-06 test case description

Test Case ID	**TC-06**
Title	Delete Transaction – Confirmation and Totals Update
Requirement	FR-001; UC-2; US-2
Preconditions	At least one existing transaction
Steps	1. Open an existing transaction. 2. Click delete. 3. Confirm deletion.
Expected Result	Row removed; totals/summary recalculates; confirmation used to prevent accidental deletion; persisted removal.
Actual Result	
Status	
Priority	High

5.5.7 TC-007

Table 10: TC-07 test case description

Test Case ID	**TC-07**
Title	Delete Transaction – Cancel Preservation
Requirement	FR-001; UC-2; US-2
Preconditions	At least one existing transaction
Steps	1. Open an existing transaction. 2. Click delete. 3. In confirmation, click Cance
Expected Result	No deletion; list and totals unchanged
Actual Result	
Status	
Priority	Medium

5.5.8 TC-008

Table 11: TC-08 test case description

Test Case ID	**TC-08**
Title	Persistence – Transactions Survive App Restart
Requirement	FR-001; US-8
Preconditions	At least one existing transaction
Steps	1. Close app. 2. Reopen app. 3. Go to Transactions.
Expected Result	Previously saved transactions load automatically; no manual import required.
Actual Result	
Status	
Priority	High

5.5.9 TC-009

Table 12: TC-09 test case description

Test Case ID	**TC-09**
Title	View All Transactions – Default List/Table
Requirement	FR-002; UC-3; US-3
Preconditions	At least one existing transaction
Steps	1. Open Transactions view.
Expected Result	All transactions visible in a readable table/list with key columns; totals visible.
Actual Result	
Status	
Priority	High

5.5.10 TC-010

Table 13: TC-10 test case description

Test Case ID	**TC-10**
Title	Filter by Category (Single Category)
Requirement	FR-002; UC-3; US-3
Preconditions	Transactions exist across multiple categories.
Steps	1. Apply category filter (e.g., Food).
Expected Result	Only rows with category = "Food" shown; totals/summary reflect filtered subset.
Actual Result	
Status	
Priority	High

5.5.11 TC-011

Table 14: TC-11 test case description

Test Case ID	**TC-11**
Title	Filter by Date Range
Requirement	FR-002; UC-3; US-3

Preconditions	Transactions exist across multiple dates.
Steps	1. Set start and end date.
Expected Result	Only transactions within range shown; totals reflect range.
Actual Result	
Status	
Priority	High

5.5.12 TC-012

Table 15: TC-12 test case description

Test Case ID	**TC-12**
Title	Combined Filters – Date Range + Category
Requirement	FR-002; UC-3; US-3
Preconditions	Data spans categories and dates.
Steps	1. Apply date range. 2. Apply category.
Expected Result	List narrows to intersection; totals update accordingly.
Actual Result	
Status	
Priority	High

5.5.13 TC-013

Table 16: TC-13 test case description

Test Case ID	**TC-13**
Title	Filter results – No Matches
Requirement	FR-002; UC-3; US-3
Preconditions	Data set present
Steps	1. Apply filters that yield zero results.
Expected Result	Empty list message + zero totals displayed clearly.
Actual Result	
Status	
Priority	Medium

5.5.14 TC-014

Table 17: TC-14 test case description

Test Case ID	TC-14
Title	Clear Filters – Restore Full List
Requirement	FR-002; UC-3; US-3
Preconditions	Active filter(s) applied.
Steps	1. Click “Clear Filters”
Expected Result	All transactions visible again; totals reflect full data set
Actual Result	
Status	
Priority	Medium

5.5.15 TC-015

Table 18: TC-15 test case description

Test Case ID	TC-15
Title	Keyword Search – Narrows List
Requirement	FR-002; UC-3; US-3
Preconditions	Transactions with varied notes/payee/category names exist
Steps	1. Enter keyword (e.g., “coffee”) in search.
Expected Result	Only matching rows shown (by configured fields: note, payee, category); totals update; clearing search restores list.
Actual Result	
Status	
Priority	Medium

5.5.16 TC-016

Table 19: TC-16 test case description

Test Case ID	TC-16
Title	Set Monthly Budget – Valid Positive Amount

Requirement	FR-003; UC-4; US-4
Preconditions	Category exists (e.g., Food).
Steps	1. Open Budgets. 2. Select “Food”. 3. Enter 200.00. 4. Save
Expected Result	Budget stored for current month; remaining allowance visible.
Actual Result	
Status	
Priority	Medium

5.5.17 TC-017

Table 20: TC-17 test case description

Test Case ID	**TC-17**
Title	Set Monthly Budget – Reject Invalid Inputs
Requirement	FR-003; UC-4; US-4
Preconditions	Budgets view open
Steps	1. Enter negative or blank budget. 2. Save
Expected Result	Save blocked; clear error message; no budget created/changed
Actual Result	
Status	
Priority	Medium

5.5.18 TC-018

Table 21: TC-18 test case description

Test Case ID	**TC-18**
Title	Update Existing Budget – Recalculate Remaining
Requirement	FR-003; UC-4; US-4
Preconditions	Budget already set for category; some spending exists
Steps	1. Edit “Food” budget 200 → 250.

	2. Save
Expected Result	New limit stored; remaining allowance recalculated and displayed.
Actual Result	
Status	
Priority	Medium

5.5.19 TC-019

Table 22: TC-19 test case description

Test Case ID	**TC-19**
Title	Overspending Alert – Trigger at Equal to Limit
Requirement	FR-003; UC-6; US-5
Preconditions	Budget = 100 for Food; current spending = 90
Steps	1. Add Food expense of 10.
Expected Result	Alert appears indicating limit reached (category shown); remains visible until acknowledged/reset
Actual Result	
Status	
Priority	Medium

5.5.20 TC-020

Table 23: TC-20 test case description

Test Case ID	**TC-20**
Title	Overspending Alert – Trigger when Exceeding Limit
Requirement	FR-003; UC-6; US-5
Preconditions	Budget = 100 for Food; current spending = 95
Steps	1. Add Food expense of 10 (total 105).
Expected Result	Alert shows category and overage amount (e.g., +5); persists until acknowledged/reset
Actual Result	
Status	
Priority	Medium

5.5.21 TC-021

Table 24: TC-21 test case description

Test Case ID	**TC-21**
Title	No Budget Defined – No Alert
Requirement	FR3; UC-6; US-5
Preconditions	No budgets set.
Steps	1. Add multiple expenses in any category.
Expected Result	No budget alert appears.
Actual Result	
Status	
Priority	Medium

5.5.22 TC-022

Table 25: TC-22 test case description

Test Case ID	**TC-22**
Title	Monthly Summary – Correct Totals
Requirement	FR-004; UC-5; US-6
Preconditions	Month has known set of incomes/expenses
Steps	1. Open Reports. 2. Select "This Month".
Expected Result	Income, expenses, and balance compute correctly and match dataset.
Actual Result	
Status	
Priority	High

5.5.23 TC-023

Table 26: TC-23 test case description

Test Case ID	**TC-23**
Title	Custom Range Summary – Correct Totals

Requirement	FR-004; UC-5; US-6
Preconditions	Data across multiple months
Steps	1. Choose custom start/end dates across months.
Expected Result	Totals reflect only the selected range; displayed clearly.
Actual Result	
Status	
Priority	Medium

5.5.24 TC-024

Table 27: TC-24 test case description

Test Case ID	**TC-24**
Title	Summary – No Data Path
Requirement	FR-004; UC-5; US-7
Preconditions	Select a period with no transactions
Steps	1. Open Reports
Expected Result	Zeros shown for totals; clear “no records found” note
Actual Result	
Status	
Priority	Medium

5.5.25 TC-025

Table 28: TC-25 test case description

Test Case ID	**TC-25**
Title	Category Chart – Accurate Visualization
Requirement	FR-004; UC-5; US-7
Preconditions	Transactions across multiple categories.
Steps	1. Open Reports with chart (pie/bar).
Expected Result	Chart values/percentages match underlying totals; labels/legend are clear.
Actual Result	
Status	

Priority	Low

5.5.26 TC-026

Table 29: TC-26 test case description

Test Case ID	**TC-26**
Title	Reports/Charts Auto-Update on Data/Filter Change
Requirement	FR-004; UC-5; US-7
Preconditions	Reports open with visible chart; baseline totals noted
Steps	1. Add/edit/delete a transaction OR change filters
Expected Result	Summary and chart refresh immediately to reflect the change.
Actual Result	
Status	
Priority	Medium

5.5.27 TC-027

Table 30: TC-27 test case description

Test Case ID	**TC-27**
Title	Usability Heuristics – Labels, Fonts, Navigation
Requirement	NFR-Usability
Preconditions	UI available (Transactions, Budgets, Reports)
Steps	1. Inspect labels, font sizes, contrast. 2. Walk through navigation paths between key views.
Expected Result	Clear labels; readable fonts; predictable navigation; consistent UI components
Actual Result	
Status	
Priority	Medium

5.5.28 TC-028

Table 31: TC-28 test case description

Test Case ID	**TC-28**
Title	Performance – Typical Dataset Responsiveness
Requirement	NFR- Performance
Preconditions	Seed transactions locally
Steps	1. Launch app and open Transactions. 2. Apply filters/search repeatedly
Expected Result	Loads and interactions complete without noticeable lag; list/filters remain responsive
Actual Result	
Status	
Priority	Medium

5.5.29 TC-029

Table 32: TC-29 test case description

Test Case ID	**TC-29**
Title	Reliability – Immediate Save & Graceful Invalid Handling
Requirement	NFR- Reliability
Preconditions	App running
Steps	1. Add/edit/delete; immediately kill and relaunch app. 2. Attempt invalid inputs on add/edit.
Expected Result	Most recent changes present after relaunch (immediate save); invalid inputs yield clear feedback; no data corruption
Actual Result	
Status	
Priority	High

5.5.30 TC-030.

Table 33: TC-30 test case description

Test Case ID	**TC-30**

Title	Privacy/Offline – No Network Dependency
Requirement	Privacy/Offline
Preconditions	Device offline (disable network)
Steps	1. Launch app. 2. Perform add/view/filter/report flows
Expected Result	All features function offline; no network calls attempted
Actual Result	
Status	
Priority	High

5.5.31 TC-031

Table 34: TC-31 test case description

Test Case ID	**TC-31**
Title	Portability – Run on Windows/macOS/Linux
Requirement	NFR-Portability
Preconditions	Build packaged with supported Java runtime
Steps	1. Start app on each OS. 2. Smoke test add/view flows.
Expected Result	App launches and core flows work on all three platforms
Actual Result	
Status	
Priority	Medium

5.5.32 TC-032

Table 35: TC-32 test case description

Test Case ID	**TC-32**
Title	Maintainability – Modular Structure & Unit Tests Present
Requirement	NFR-Maintainability
Preconditions	Repo accessible
Steps	1. Review project structure (e.g., MVC separation. 2. Verify presence of unit tests for core model/services.

Expected Result	Clear modular separation; baseline unit tests exist; naming/packaging consistent.
Actual Result	
Status	
Priority	Medium

6 User Manual

6.1 Introduction

SmartBudget is a personal finance tracking application designed to help users record daily transactions, manage monthly budgets, view summaries, and analyze spending patterns. The application operates fully offline, ensures immediate data saving, and supports Windows, macOS, and Linux systems. This manual explains how to use all features of SmartBudget, including:

- Recording, editing, and deleting income or expense transactions
- Browsing and filtering financial history
- Setting category-based monthly budgets
- Receiving automatic overspending alerts
- Viewing summaries and category charts
- Understanding how data is loaded and saved

6.2 Getting Started

6.2.1 System Requirements

- **Operating Systems:** Windows, macOS, Linux
- **Java Runtime:** Any supported Java runtime compatible with JavaFX
- **Network:** Not required (SmartBudget functions fully offline)

6.2.2 First Launch

On startup, SmartBudget will automatically:

- Load saved transactions (if any)

- Load monthly budgets (if previously set)
- Display your most recent financial data

If no prior data exists, SmartBudget will inform you that your finance database is empty.

6.3 Navigating SmartBudget

The SmartBudget interface consists of four main sections:

- Dashboard - Summary of Total Income, Expenses, Budgets & Recent Transactions
- Transactions – Record, view, filter, edit, and delete income or expense entries.
- Budgets – Set monthly spending limits per category.
- Reports – View summary totals and category charts.

6.4 Transactions

6.4.1 Adding a Transaction

You can record either an expense or income.

Steps:

1. Open the Transactions tab.
2. Click Add Transaction.
3. Choose Expense or Income.
4. Enter the required details.
5. Click Save.

6.4.2 Editing a Transaction

Steps:

1. Select a transaction from the list.
2. Click Edit.
3. Modify amount, category, date, or note.
4. Save changes.

6.4.3 Deleting a Transaction

Steps:

1. Open the transaction you want to delete.
2. Click Delete.

3. Confirm deletion in the confirmation prompt.

6.4.4 Viewing Transactions

Displays a table of all entries with totals for income, expenses, and balance.

6.4.5 Filtering Transactions

Filter by:

- Category
- Date range
- Combined filters
- Keyword search

Clear Filters resets the list.

6.5 Budgets

6.5.1 Setting a Budget

Steps:

1. Open Budgets tab.
2. Select a category.
3. Enter a monthly limit.
4. Save.
5. Updating a Budget
   Modify the limit and save.

6.6 Overspending Alerts

Alerts appear when spending reaches or exceeds the budget limit for a category.

6.7 Reports

6.7.1 Summary Report

Steps:

1. Open the Reports ab
2. Select reporting period:

- This Month
- Last Month
- Custom Range

Reports show totals for income, expenses, and balance.

6.7.2 Category Spending Chart

SmartBudget displays at least one visual chart (pie or bar) broken down by category.

Chart Behaviour:

- Automatically updates when transactions change
- Labels and legends clearly identify categories
- Values correspond exactly to underlying data

6.8 Data Persistence

All changes save instantly and load automatically on startup.

6.9 Offline Mode and Privacy

SmartBudget works 100% offline. All data remains local.

6.10 Performance & Responsiveness

Optimized for large datasets with fast loading and filtering.

6.11 Cross-Platform Support

Works uniformly on Windows, macOS, and Linux.

6.12 Maintenance & Updates

Modular architecture, clean code structure, and unit test support.

6.13 Frequently Asked Questions

Question 1: Does SmartBudget work offline?

Yes, completely offline.

Question 2: Is data synced?

Not in the current version. Data is stored locally on each device.

Question 3: How to back up data?

Manually copy the app's storage folder to another location or device.

6.14 Summary

SmartBudget provides a clean, reliable, and intuitive way to:

- Track financial transactions
- Monitor category budgets
- Receive warnings about overspending
- Visualize spending in charts
- Operate securely and offline

This user manual should help you navigate and use all features confidently.

7 Prototype Overview

This section describes the interactive prototype created for the SmartBudget – Personal Finance Tracker. The prototype was built using Figma and focuses on demonstrating the full user experience of the final application. It does not include backend logic or functional code or even the final UI of the final application. Instead, it presents the visual layout, navigation flow, and screen structure that will guide the development of the real application.

The prototype includes all major features defined in our requirements, use cases, and user stories. It gives a clear view of how users will move through the app and how each feature will look when implemented.

7.1 Purpose of the Prototype

The main goal of this prototype is to let users understand the planned interface early in the project. It shows:

- How each feature will appear to the user
- The visual style, layout, and structure of the app
- Main navigation paths between sections
- The expected flow for recording transactions, viewing reports, and setting budgets

7.2 Feature Coverage

The prototype includes screens and flows for all key requirements defined in the earlier deliverables:

7.2.1 Record, Edit, and Delete Transactions (FR1)

- “Add Transaction” screen with fields for amount, type, category, date, and notes
- Transaction list showing existing entries
- Edit and delete options displayed within each transaction row

7.2.2 Viewing and Filtering Transactions (FR2)

- A Transactions screen with a clean list layout
- Filter button for selecting category or date range
- Search field for keyword-based filtering
- Clear indicators when filters are active

7.2.3 Monthly Budgets and Alerts (FR3)

- A Budgets screen for setting monthly spending limits per category
- Editable limit field for each category
- Visual indicator for remaining budget
- Alert designs showing overspending warnings

7.2.4 Summary and Charts (FR4)

- Summary screen showing total income, expenses, and balance
- Visual chart (pie or bar) of spending by category
- Option to select monthly or custom periods

7.3 Navigation Flow

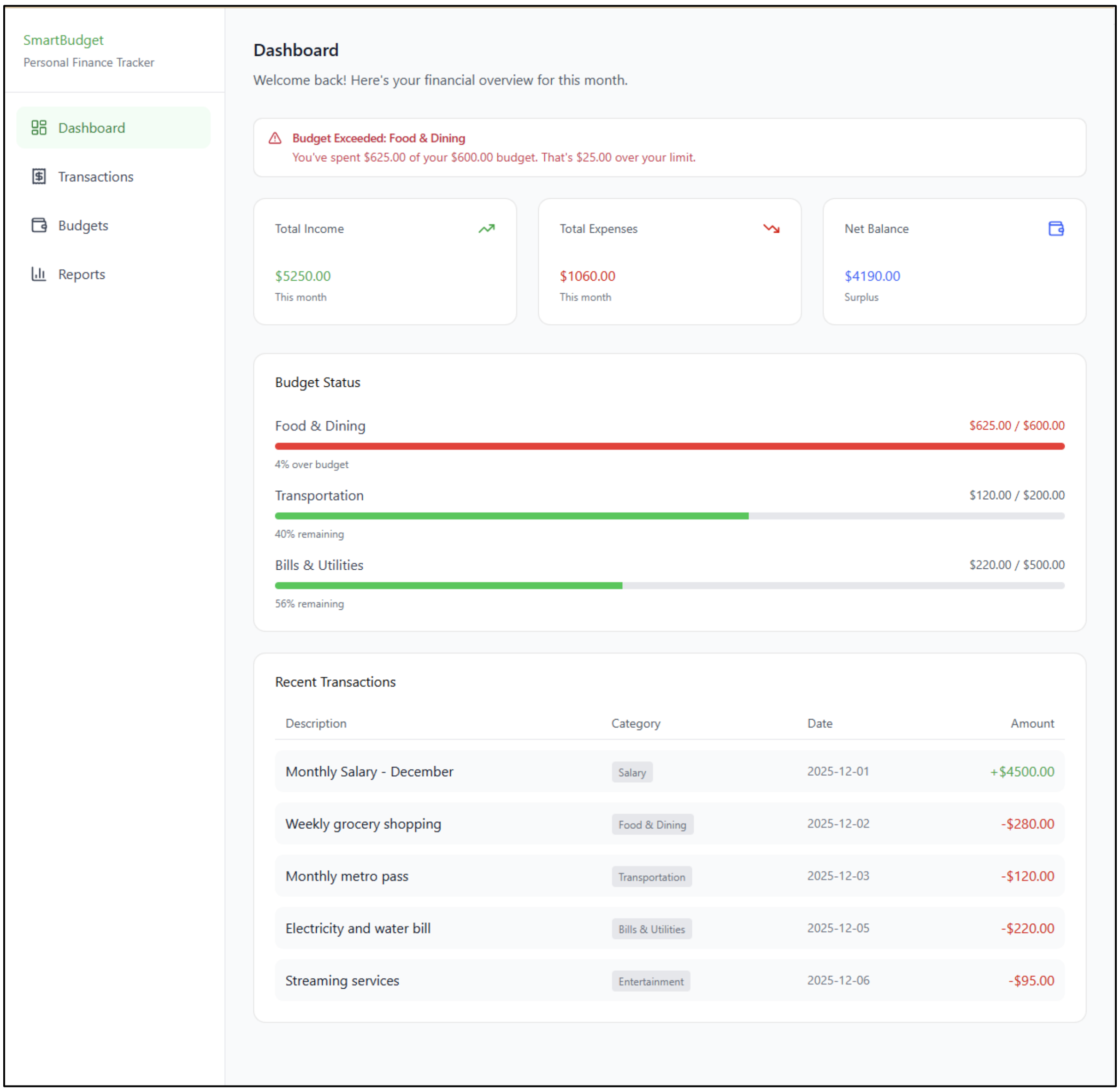
The prototype follows a simple and intuitive navigation model:
A left navigation bar linking:

- **Dashboard**
- **Transactions**
- **Budgets**
- **Reports**

This supports the non-functional requirement for usability and ensures easy movement between main features.

7.4 Prototype Screenshots

7.4.1 Dashboard view



Dashboard

Welcome back! Here's your financial overview for this month.

Budget Exceeded: Food & Dining
You've spent $625.00 of your $600.00 budget. That's $25.00 over your limit.

Total Income	Total Expenses	Net Balance
$5250.00	$1060.00	$4190.00
This month	This month	Surplus

Budget Status

Food & Dining — $625.00 / $600.00 — 4% over budget

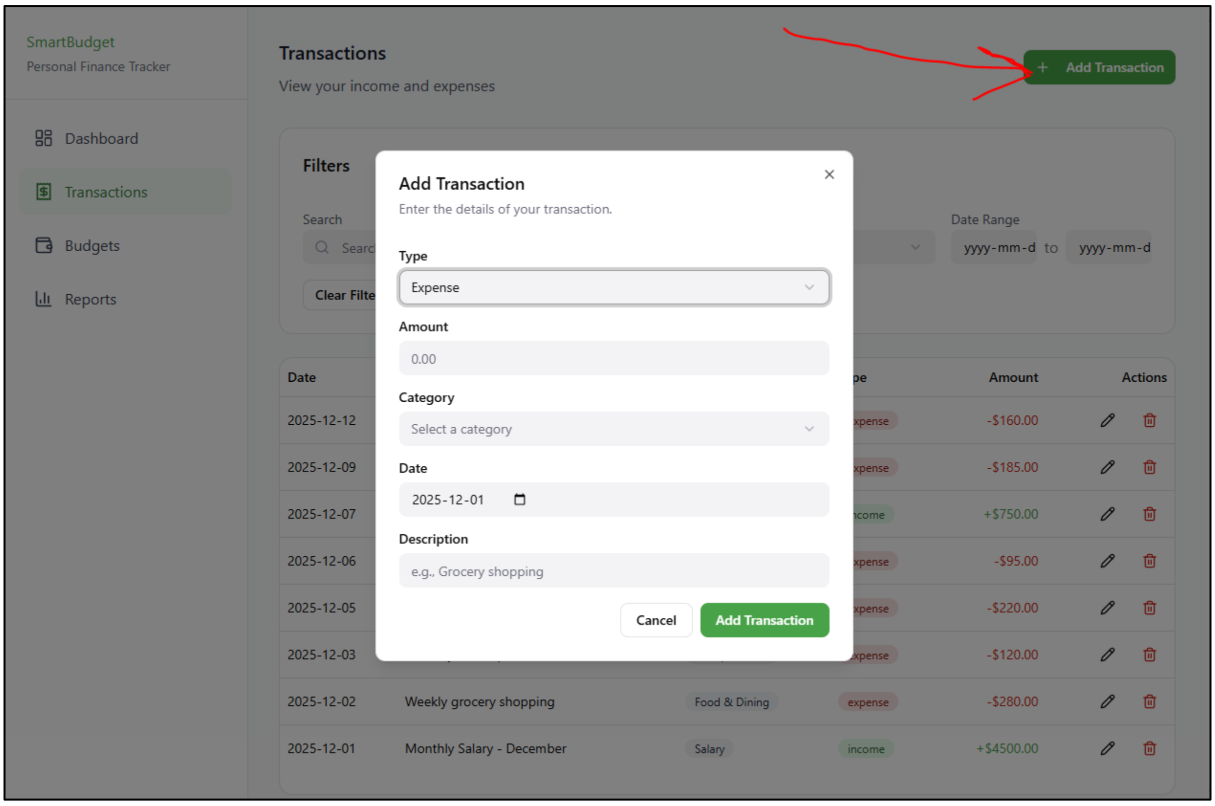
Transportation — $120.00 / $200.00 — 40% remaining

Bills & Utilities — $220.00 / $500.00 — 56% remaining

Recent Transactions

Description	Category	Date	Amount
Monthly Salary - December	Salary	2025-12-01	+$4500.00
Weekly grocery shopping	Food & Dining	2025-12-02	-$280.00
Monthly metro pass	Transportation	2025-12-03	-$120.00
Electricity and water bill	Bills & Utilities	2025-12-05	-$220.00
Streaming services	Entertainment	2025-12-06	-$95.00

7.4.2 Add Transaction form



SmartBudget
Personal Finance Tracker

- Dashboard
- Transactions
- Budgets
- Reports

Transactions
View your income and expenses

+ Add Transaction

Filters

Search
Searc

Clear Filte

Date Range
yyyy-mm-d to yyyy-mm-d

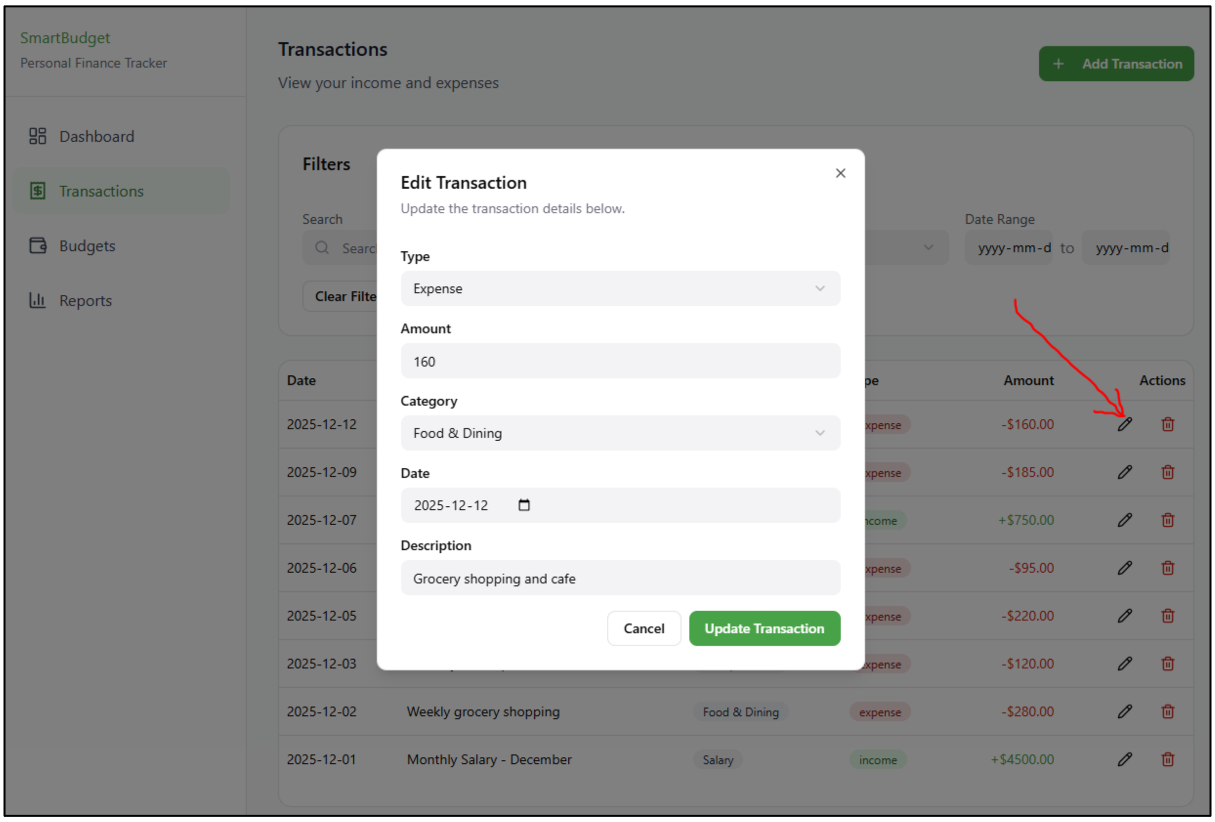
Add Transaction
Enter the details of your transaction.

Type	Expense
Amount	0.00
Category	Select a category
Date	2025-12-01
Description	e.g., Grocery shopping

Cancel | Add Transaction

Date			pe	Amount	Actions
2025-12-12			xpense	-$160.00	
2025-12-09			xpense	-$185.00	
2025-12-07			come	+$750.00	
2025-12-06			xpense	-$95.00	
2025-12-05			xpense	-$220.00	
2025-12-03			xpense	-$120.00	
2025-12-02	Weekly grocery shopping	Food & Dining	expense	-$280.00	
2025-12-01	Monthly Salary - December	Salary	income	+$4500.00	

7.4.3 Edit Transaction form



SmartBudget
Personal Finance Tracker

- Dashboard
- Transactions
- Budgets
- Reports

Transactions
View your income and expenses

+ Add Transaction

Filters

Search
Searc

Clear Filte

Date Range
yyyy-mm-d to yyyy-mm-d

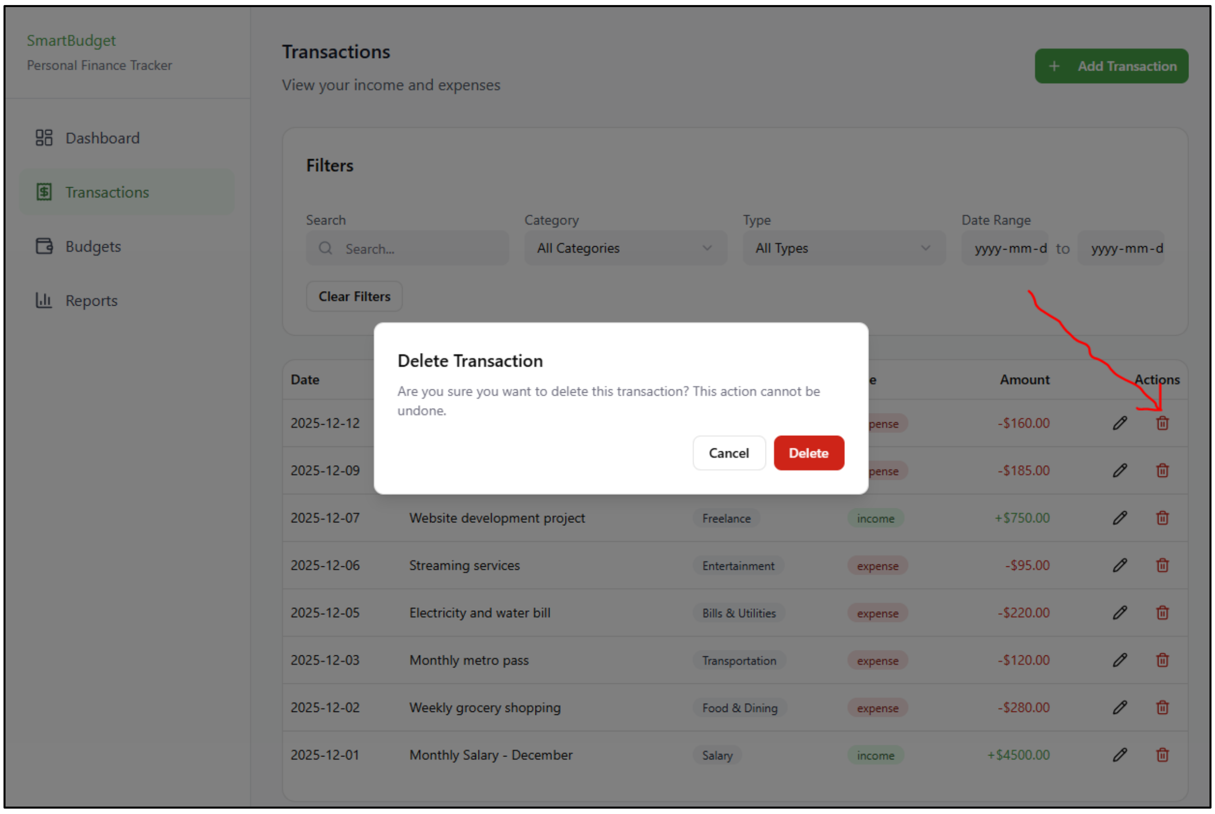
Edit Transaction
Update the transaction details below.

Type	Expense
Amount	160
Category	Food & Dining
Date	2025-12-12
Description	Grocery shopping and cafe

Cancel | Update Transaction

Date			pe	Amount	Actions
2025-12-12			xpense	-$160.00	
2025-12-09			xpense	-$185.00	
2025-12-07			come	+$750.00	
2025-12-06			xpense	-$95.00	
2025-12-05			xpense	-$220.00	
2025-12-03			xpense	-$120.00	
2025-12-02	Weekly grocery shopping	Food & Dining	expense	-$280.00	
2025-12-01	Monthly Salary - December	Salary	income	+$4500.00	

7.4.4 Delete Transaction form


SmartBudget
Personal Finance Tracker

Dashboard
Transactions
Budgets
Reports

Transactions
View your income and expenses

+ Add Transaction

Filters

Search	Category	Type	Date Range
Search...	All Categories	All Types	yyyy-mm-d to yyyy-mm-d

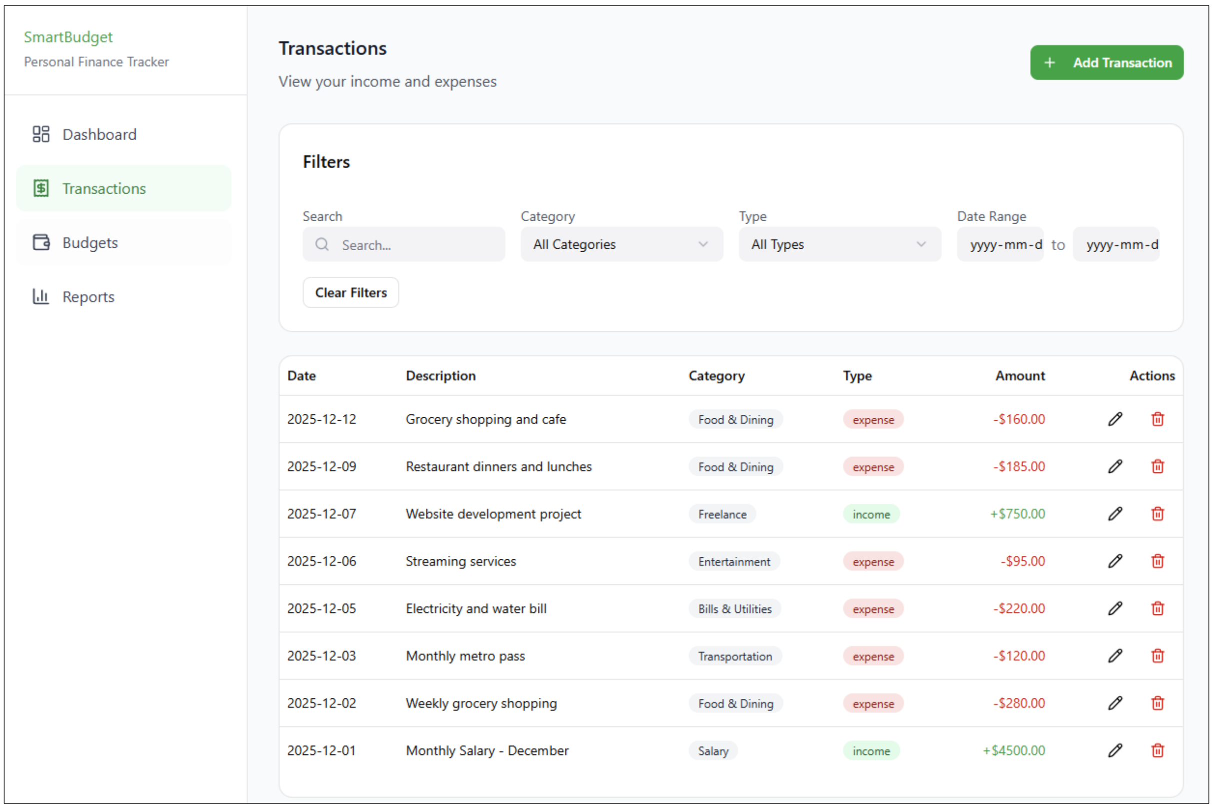
Clear Filters

Delete Transaction
Are you sure you want to delete this transaction? This action cannot be undone.

Cancel | Delete

Date			e	Amount	Actions
2025-12-12			pense	-$160.00	
2025-12-09			pense	-$185.00	
2025-12-07	Website development project	Freelance	income	+$750.00	
2025-12-06	Streaming services	Entertainment	expense	-$95.00	
2025-12-05	Electricity and water bill	Bills & Utilities	expense	-$220.00	
2025-12-03	Monthly metro pass	Transportation	expense	-$120.00	
2025-12-02	Weekly grocery shopping	Food & Dining	expense	-$280.00	
2025-12-01	Monthly Salary - December	Salary	income	+$4500.00	

7.4.5 Filters and search UI


SmartBudget
Personal Finance Tracker

Dashboard
Transactions
Budgets
Reports

Transactions
View your income and expenses

+ Add Transaction

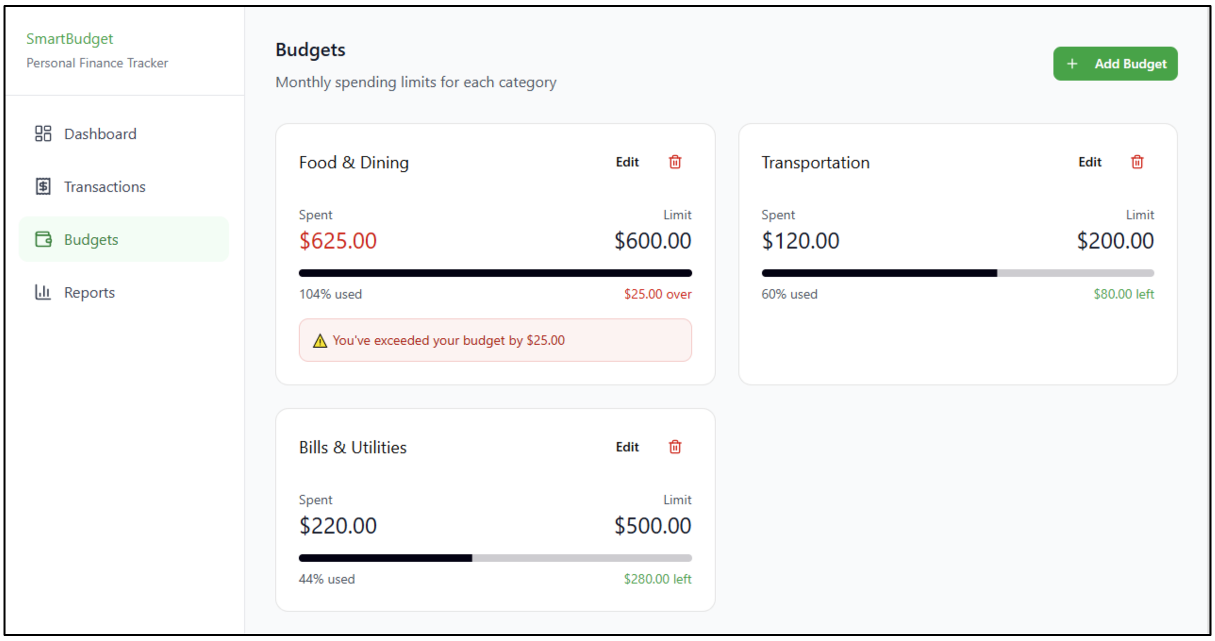
Filters

Search	Category	Type	Date Range
Search...	All Categories	All Types	yyyy-mm-d to yyyy-mm-d

Clear Filters

Date	Description	Category	Type	Amount	Actions
2025-12-12	Grocery shopping and cafe	Food & Dining	expense	-$160.00	
2025-12-09	Restaurant dinners and lunches	Food & Dining	expense	-$185.00	
2025-12-07	Website development project	Freelance	income	+$750.00	
2025-12-06	Streaming services	Entertainment	expense	-$95.00	
2025-12-05	Electricity and water bill	Bills & Utilities	expense	-$220.00	
2025-12-03	Monthly metro pass	Transportation	expense	-$120.00	
2025-12-02	Weekly grocery shopping	Food & Dining	expense	-$280.00	
2025-12-01	Monthly Salary - December	Salary	income	+$4500.00	

7.4.6 Budgets screen and Overspending alert sample



SmartBudget
Personal Finance Tracker

Dashboard
Transactions
Budgets
Reports

Budgets
Monthly spending limits for each category

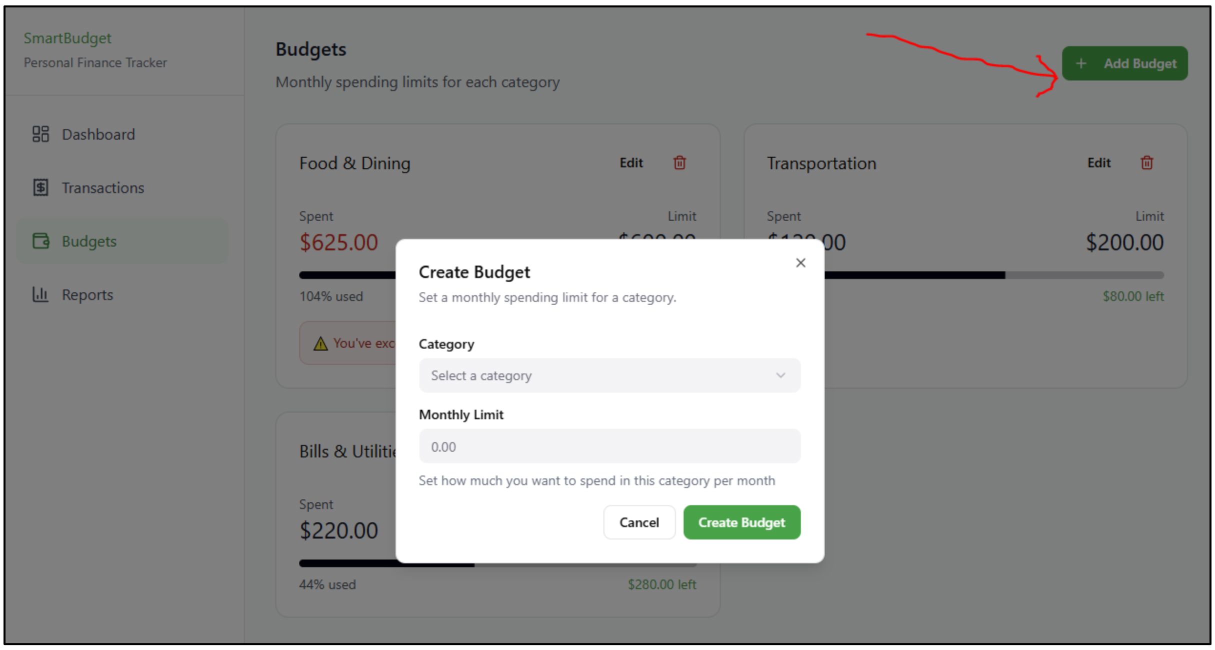
+ Add Budget

Food & Dining — Edit
Spent $625.00 — Limit $600.00
104% used — $25.00 over
⚠ You've exceeded your budget by $25.00

Transportation — Edit
Spent $120.00 — Limit $200.00
60% used — $80.00 left

Bills & Utilities — Edit
Spent $220.00 — Limit $500.00
44% used — $280.00 left

7.4.7 Add Budget form



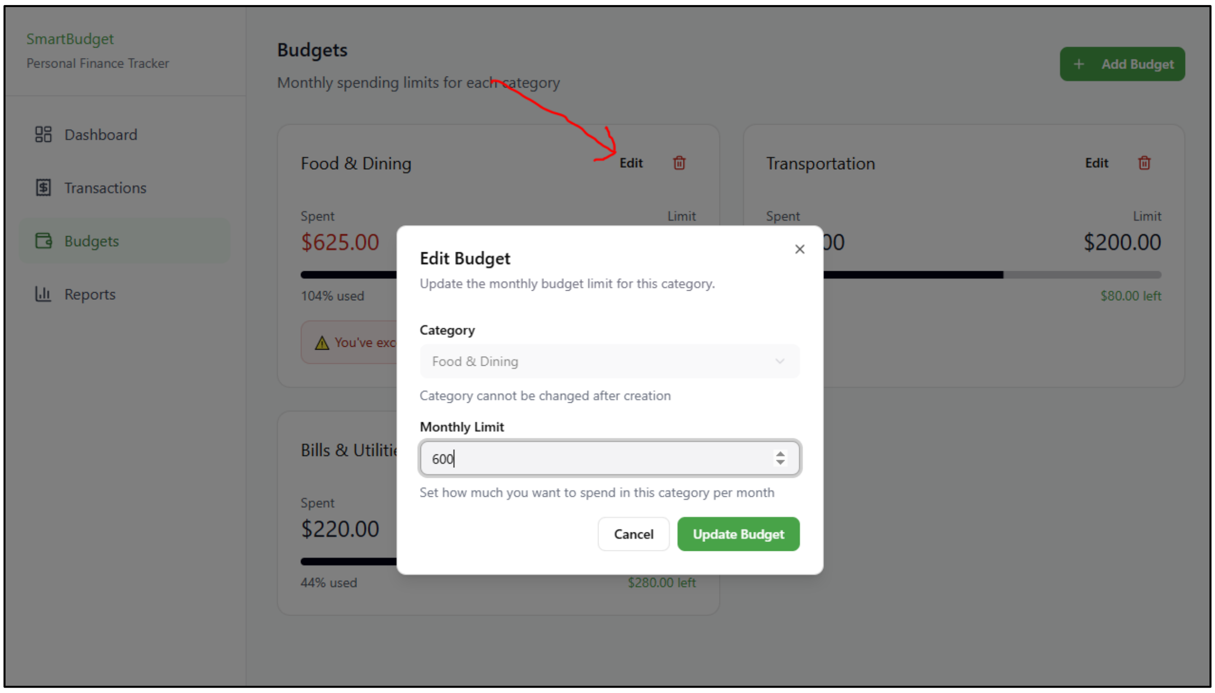
Create Budget
Set a monthly spending limit for a category.

Category
Select a category

Monthly Limit
0.00
Set how much you want to spend in this category per month

Cancel
Create Budget

7.4.8 Edit Budget form


SmartBudget
Personal Finance Tracker

Dashboard
Transactions
Budgets
Reports

Budgets
Monthly spending limits for each category

+ Add Budget

Food & Dining — Edit
Spent $625.00 — Limit
104% used
⚠ You've exc

Transportation — Edit
Spent 00 — Limit $200.00
$80.00 left

Bills & Utilitie
Spent $220.00
44% used — $280.00 left

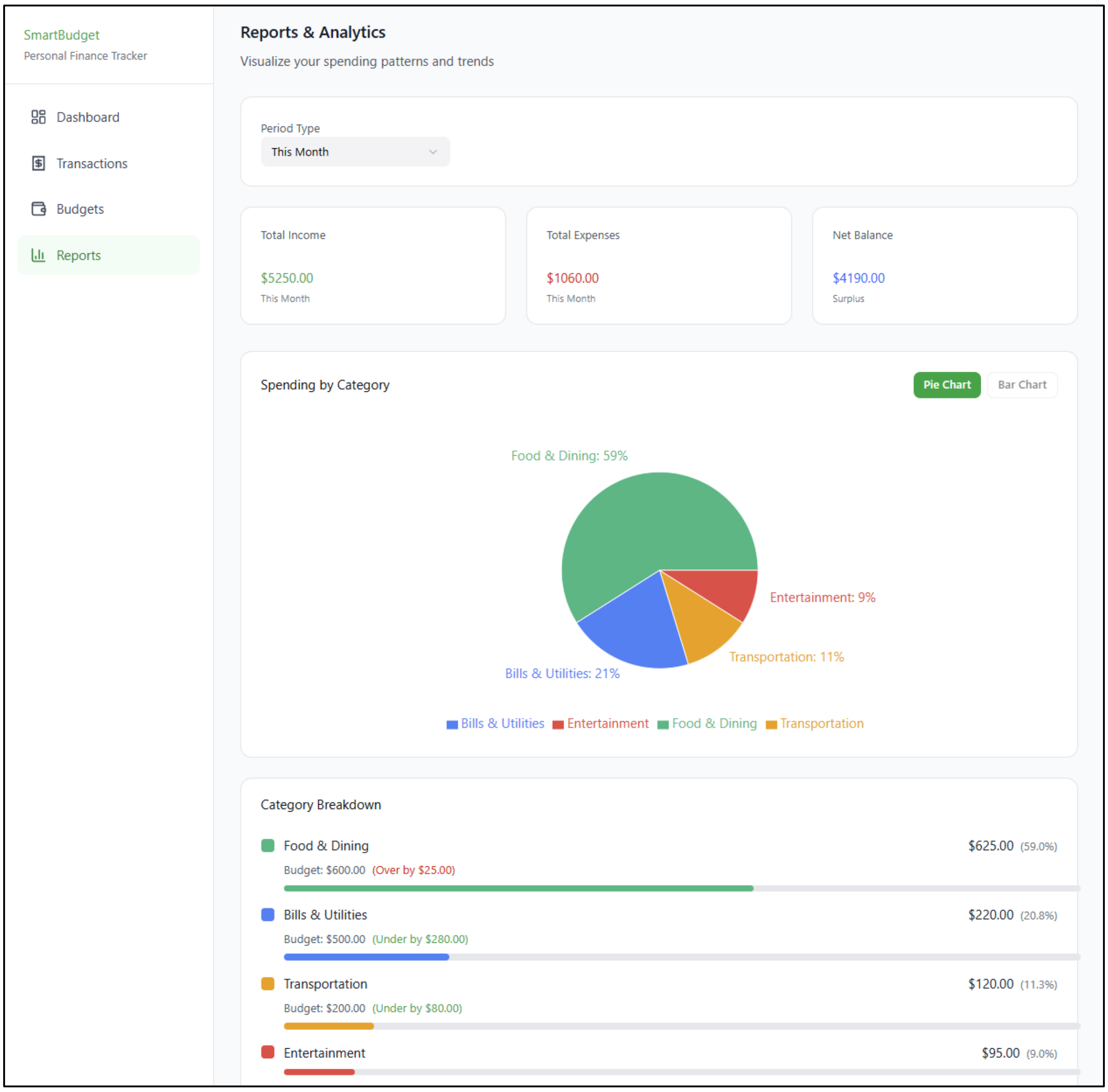
Edit Budget
Update the monthly budget limit for this category.

Category
Food & Dining
Category cannot be changed after creation

Monthly Limit
600
Set how much you want to spend in this category per month

Cancel | Update Budget

7.4.9 Monthly summary with chart



SmartBudget
Personal Finance Tracker

- Dashboard
- Transactions
- Budgets
- Reports

Reports & Analytics
Visualize your spending patterns and trends

Period Type: This Month

Total Income	Total Expenses	Net Balance
$5250.00	$1060.00	$4190.00
This Month	This Month	Surplus

Spending by Category

Category Breakdown

Food & Dining Budget: $600.00 (Over by $25.00)	$625.00 (59.0%)
Bills & Utilities Budget: $500.00 (Under by $280.00)	$220.00 (20.8%)
Transportation Budget: $200.00 (Under by $80.00)	$120.00 (11.3%)
Entertainment	$95.00 (9.0%)

7.5 Alignment With Requirements and Testing

The prototype reflects all functional and non-functional requirements documented in Deliverable 2.1. It also supports the testing plan by clearly showing:

- Each interface where a test case action will occur
- Expected UI behaviour (inputs, buttons, messages)
- Visual confirmation of flows covered in the traceability matrix

Although the prototype is not executable, it provides the complete UI foundation needed for implementation in next iterations.

7.6 Prototype Link

The full interactive prototype can be viewed here:
https://dandy-wages-80515360.figma.site/

8 Code Repository and Branching Strategy

The SmartBudget project is hosted on GitHub, and all project deliverables, documents, and source files are stored and updated there throughout each iteration.

8.1 Repository Link:

https://github.com/Hercules-Hawk/smart-budget-personal-finance-tracker

8.2 Branching Strategy:

For branching, our team follows a simple and organized strategy to keep the code stable while allowing each deliverable to be developed separately. The main branch serves as the stable branch and contains the most reliable version of the project. We do not use a dedicated development branch.

For each deliverable, we create a separate feature branch where we work on the required files for that iteration (for example: docs/deliverable-1.1, docs/deliverable-1.2, docs/deliverable-2.1). After completing the work for a deliverable, the branch is reviewed and then merged into the main branch. This approach keeps the repository organized and ensures that every deliverable is tracked clearly through its own branch, while the main branch always stays clean and stable.

9 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.

9.1 Iteration Plan

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

- Deliverable 1.1 – Vision Document: Submitted.

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

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

9.1.3 Iteration 3 (Weeks 6–9 –Testing Document)

- Deliverable 2.1 – Testing Document: Submitted

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

- Deliverable 2.2 – Mid Project Report Submission: Current deliverable

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

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

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

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

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

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

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

- Deliverable 3.3 – Final Documentation Preparation

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

- Deliverable 3.3 – Final Project Submission

9.2 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.)

10 Conclusion

This deliverable brings together all the work completed so far for the SmartBudget project. In this iteration, we refined the functional and non-functional requirements, updated the use cases and user stories, and completed a full testing document with clear traceability to every requirement. We also added a complete user manual and prepared an updated prototype to guide the future development of the system. The prototype is a UX/UI design created in Figma, and it is not an implementation of the final Java desktop application. Instead, it serves as a visual model that shows how the final system should look and how users will move between features such as transactions, budgets, and reports.

These updates build on the foundation established in earlier deliverables and show steady progress toward developing the actual SmartBudget application. The documentation, test cases, and prototype created here will guide the design and implementation work in the next phases and ensure that development stays aligned with the system requirements.

In the upcoming iterations, we will start moving from planning and design into actual development. The next steps include creating the detailed software architecture, building the JavaFX interface, implementing the core features, and beginning unit testing. As we continue through the Winter 2026 term, we will focus on integration, GUI refinement, performance improvements, and completing the final testing and documentation required for the final submission. Following the Agile workflow and building on the strong base established in this deliverable, we are prepared for the next stages of the project.