1 Test Plan

1.1 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.

1.2 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.

1.3 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	
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	

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
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	
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	

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Coverage
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	
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	

1.4 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	