CSE305 Software Engineering - Sprint 3 Report (MVP)

Student: Başar Orhanbulucu
Student ID: 221805031
Project: My Budget Flow (Budget & Cash Flow Manager)
Sprint: #3 (Construction Phase & MVP Delivery)

1. Plan for Sprint 3

This section summarizes the final Sprint goals within the "Construction Phase" of the Unified Process.

- Sprint Goal: To complete the "Goals & Analytics" module, finalize the User Profile management, handle custom categories, and polish the UI/UX for the final MVP presentation.
- Date Range: Dec 29 - Jan 14
- Status: ☑ Completed

Sprint Backlog Scope & Changes

User Story	Task	Description	Status
US 5.1	Goal Management	Implementation of GoalRepository, GoalProvider, and UI for creating Savings/Expense goals.	Done
US 5.2	Profile Editing	Users can now update their Name/Surname via ProfileScreen.	Done
US 5.3	Custom Categories	Logic for users to add their own transaction categories with custom icons and colors.	Done
Tech	Connection Guard	Added ConnectionWrapper to block UI when internet connection is lost.	Done
Tech	Unit Testing	Wrote Unit Tests for Business Logic (Recurring dates, Goal progress).	Done
US 6.1	Notifications	Local notifications for upcoming bills.	Removed*

Scope Change Note: During the Construction Phase, it was decided to remove the "Local Notifications" feature. The focus was shifted to the "Recurring Transaction Engine," which automatically generates transactions when due, providing better automation than passive notifications. Additionally, a "No Internet" guard was added to improve app stability.

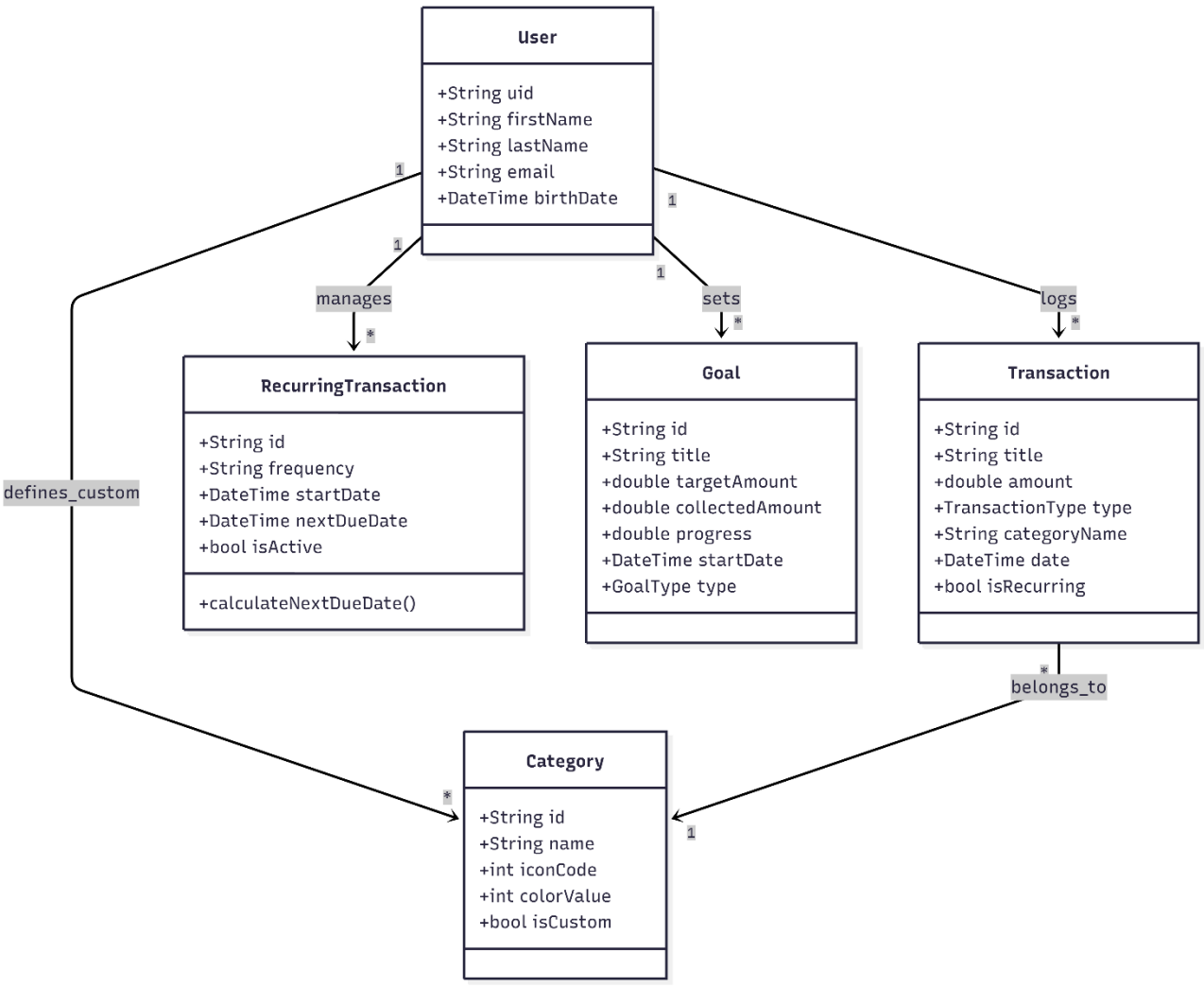
2. MVP Architecture Enhancements

Compared to the initial Sprint 1 prototype, the MVP features a significant UI/UX overhaul:

- Glassmorphism: Used GlassContainer widgets for a modern, premium look on the Dashboard and Calendar.
- Visual Feedback: Added ScaleButton for haptic feedback and bouncy animations on interactions.
- Reliability: The app now listens to connectivity changes and prevents data corruption by pausing operations when offline.

3. Domain Class Diagram (Final MVP)

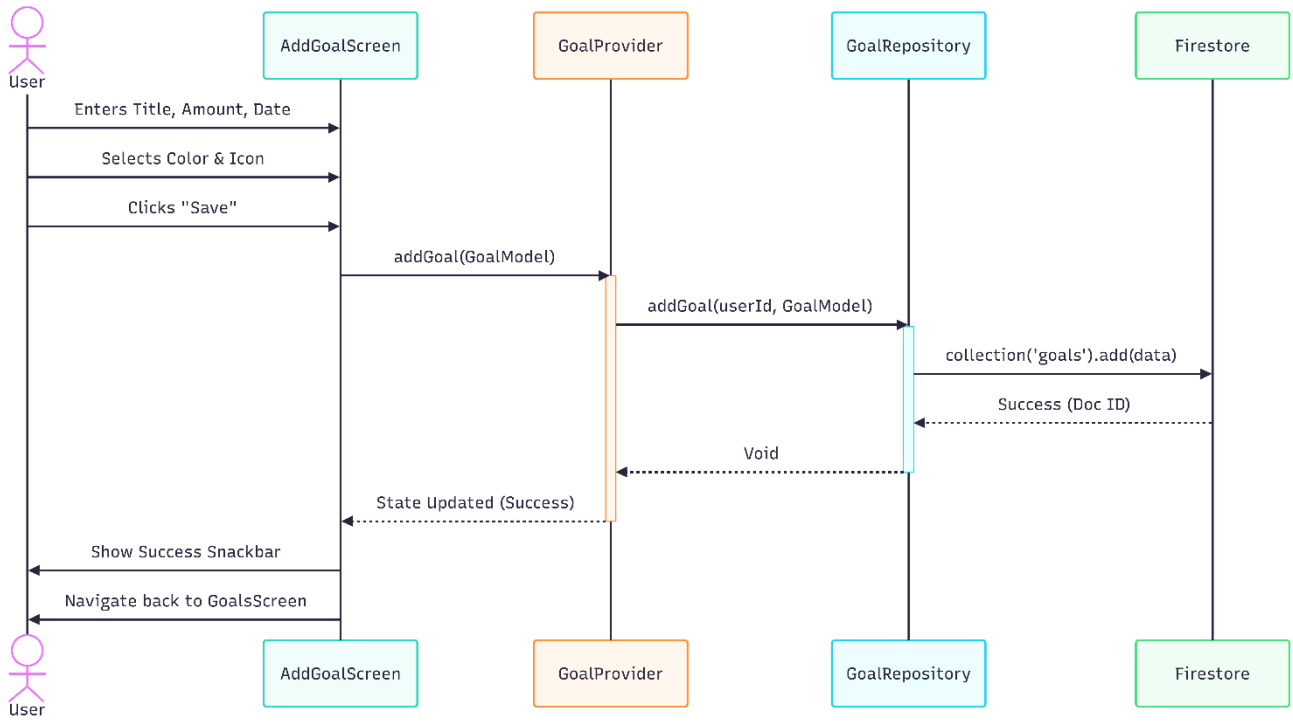
The Domain Model has evolved to support Goals and Recurring Transactions.



4. Sequence Diagrams

Scenario 1: Creating a Financial Goal

This scenario describes how a user sets up a new savings goal in the system.



Actors and Components:

- User: The person setting the goal.
- AddGoalScreen: The UI form for entering goal details (amount, date, type).
- GoalProvider: The state manager handling the logic.
- GoalRepository: The data layer communicating with Firestore.
- Firestore: The cloud database.

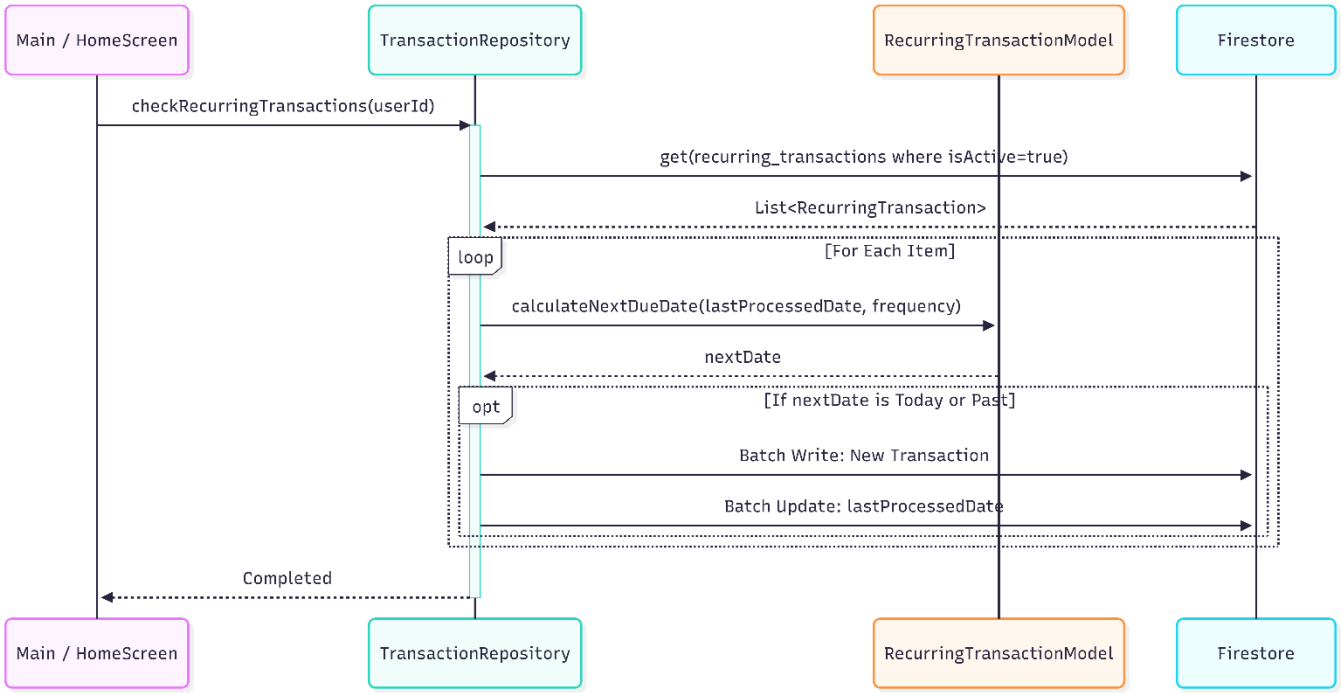
Step-by-Step Process:

1. Input: The User enters the goal title, target amount, selects a deadline, and chooses a color.
2. Trigger: The User clicks the "Save Goal" button.
3. State Call: AddGoalScreen calls the addGoal method in GoalProvider.
4. Persistence: The Provider delegates the data payload to GoalRepository.

5. Database Write: The Repository creates a new document in the users/{uid}/goals collection in Firestore.
6. Refresh: Upon success, the goalsWithProgressProvider automatically refreshes the list.
7. Feedback: The UI closes the form and shows a success SnackBar.

Scenario 2: Auto-Generating Recurring Transactions

This is the core automation logic that runs silently when the app starts, replacing the need for manual notifications.



Actors and Components:

- AppStart (Main): The initialization point of the application.
- TransactionRepository: The logic layer that checks for due dates.
- RecurringTransactionModel: The domain entity that calculates next due dates.
- Firestore: Stores the transaction history and recurring templates.

Step-by-Step Process:

1. Initialization: The app launches, and HomeScreen initializes.
2. Check Trigger: The app requests checkRecurringTransactions from the repository.
3. Fetch: The Repository fetches all *active* recurring templates from Firestore.

4. Logic Calculation: For each template, the RecurringTransactionModel calculates if the nextDueDate is today or in the past.
5. Batch Write: If due, the Repository creates a new Transaction record and updates the template's lastProcessedDate.
6. Completion: The process finishes, and the Dashboard updates automatically with the new transactions.

5. Unit Testing Results

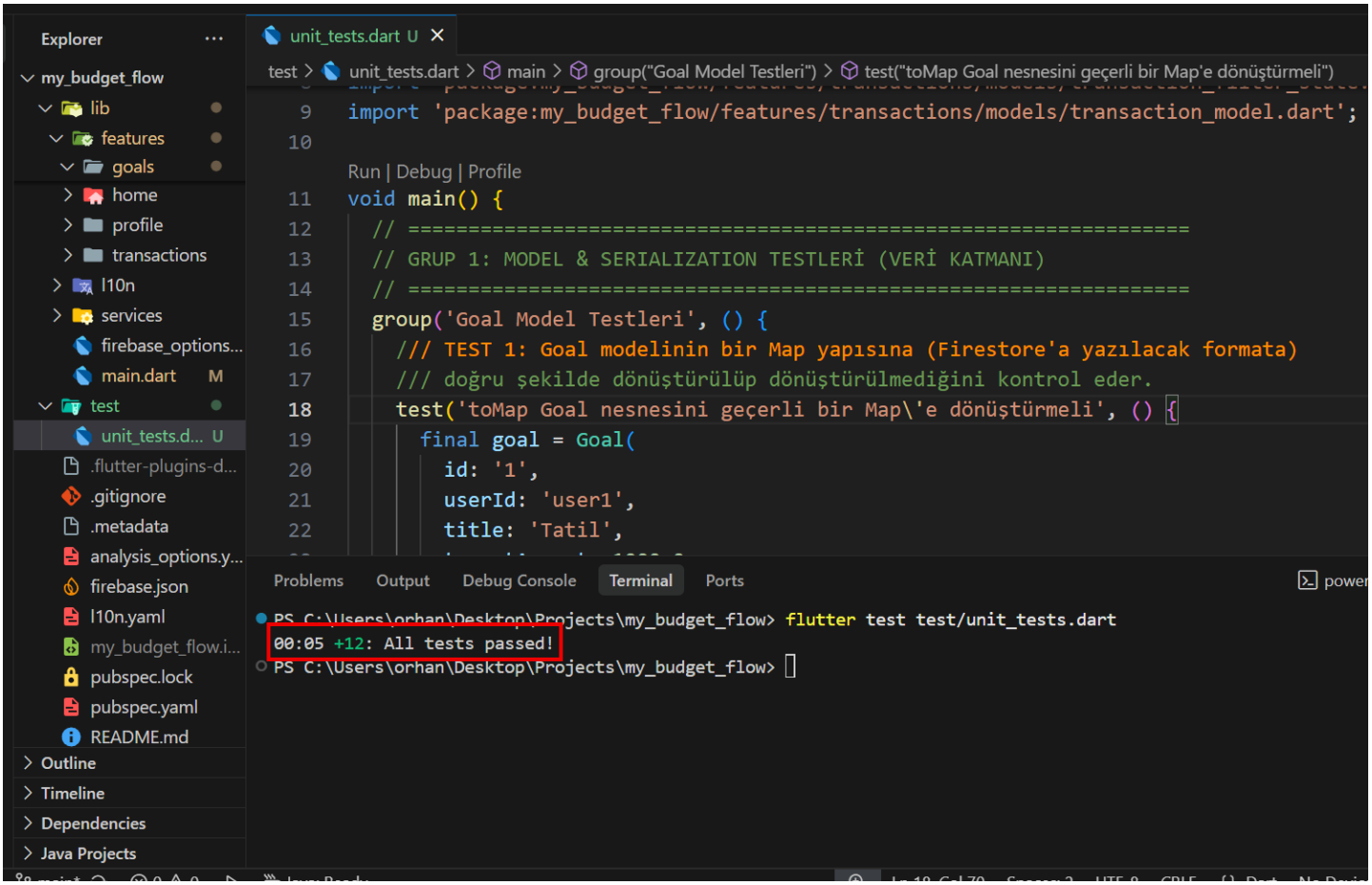
As required for the Construction Phase, Unit Tests were developed to verify critical business logic without relying on the UI.

Test Coverage:

1. Recurring Logic: Verifies that calculateNextDueDate correctly handles month rollovers (e.g., Jan 31 -> Feb 28).
2. Goal Logic: Verifies that Goal.progress correctly calculates the percentage (0.0 to 1.0) based on target vs. collected amount.
3. Filter State: Verifies that TransactionFilterState remains immutable and updates correctly via copyWith.

Test Execution:

All tests passed successfully using the flutter test command.



6. Trello Board Management

https://trello.com/invite/b/6921eb3ffb6fe09b90661536/ATTIa90b3944b9023750cc0ed893be35c9f093256CBD/my-budget-flow

7. GitHub Repository

https://github.com/basarob/my_budget_flow

8. PSI (Potentially Shippable Increment) - The MVP

At the end of Sprint 3, the Minimum Viable Product (MVP) is ready. The application is fully functional, stable, and meets the core requirements defined in the Agile Process Model.

Delivered Modules:

1. Auth System: Login, Register, Forgot Password (fully integrated with Firebase).
2. Dashboard: Real-time net balance calculation, charts, and date filtering.
3. Transaction Wallet: Adding income/expense, custom categories, and pagination.
4. Automation: Recurring transactions are automatically generated.
5. Goals: Users can track savings progress visually.
6. Profile: User details are editable.
7. Stability: Internet connection handling and comprehensive error messages.

Conclusion:

The project has successfully met the objectives of the Construction Phase. The architecture is scalable, the code is tested, and the user experience is polished.

Screenshots of the Increment: