

Software Engineering Theory and Practice

Referral Coursework Report (2025)

Budget Management App

Problem Specification

1.1 Requirements Elicitation Process

To gather user requirements for the budget management app, I created and distributed a short online questionnaire. This was shared with a small group of people aged 18 to 35 who regularly manage their personal finances. The questionnaire was designed to understand what features users expect in a budgeting tool and what problems they experience with existing apps.

The main reason for choosing a questionnaire was because it's quick for me to set up, easy to share, and effective for getting structured feedback from multiple people in a short amount of time.

Example Questions Included:

- What features would help you manage your spending more effectively?
- How often do you track your expenses?
- Would you rather add expenses manually or have them auto-categorised?
- What do you dislike about current budgeting apps?
- Would you find it useful to see your spending visualised in charts?

Summary of Responses:

Respondent	Key Feedback
------------	--------------

R1	Wants to set a monthly budget and track how close they are to overspending
R2	Prefers a clean and simple layout with as few distractions as possible
R3	Would like to see a breakdown of spending by category, preferably in graphs
R4	Wants a way to add recurring expenses quickly without retyping everything

Common Themes:

The responses highlighted a few consistent patterns. Most users wanted a simple way to record expenses and set a monthly limit. Many mentioned they prefer seeing their spending visually, especially through graphs. Several people said they value control over how expenses are organised and want to assign their own categories. Others mentioned that current apps feel too complicated or bloated with features they don't use.

This feedback helped shape the core features of the app by focusing on clarity, customisation, and visual tracking.

1.2 User Requirements

Based on the data gathered from the questionnaire, the following user requirements were identified. Each one is clearly linked back to the original responses to show how it came from real feedback.

ID	User Requirement	Source
1	Users should be able to add a new expense with a title, amount, date, and category	R1, Q1
2	Users should be able to view a summary of all expenses over time periods like daily, weekly, or monthly	R3, Q5
3	Users should be able to set a monthly spending limit and see how close they are to reaching it	R1, Q1
4	Users should be able to view spending by category such as food or transport	R3, Q3
5	Users should be able to add recurring expenses without re-entering the same details	R4, Q2
6	The app interface should be clean and easy to use	R2, Q4

7	Users should be able to change or delete any expense they entered earlier	General expectation
8	Spending should be displayed visually in pie charts or bar graphs	R3, Q5

1.3 System Requirements

The user requirements were then translated into specific system requirements that guide how the app will work. These are divided into functional and non-functional requirements.

Functional Requirements

ID	Functional Requirement
1	The app should allow users to input expenses with details such as title, date, amount, and category
2	The app should allow users to filter and view expenses by day, week, or month
3	The app should allow users to set a budget and track their remaining balance
4	The app should calculate and display total spending by category
5	The app should support adding recurring expenses with predefined values
6	The app should allow users to edit or delete previously added expenses
7	The app should display charts that show where the user's money is going

Non-Functional Requirements

ID	Non-Functional Requirement
1	The app should work smoothly on both desktop and mobile screens
2	The interface should be responsive and load quickly without lag
3	The app should save data reliably so that nothing is lost between sessions
4	The layout should be clean and uncluttered to help users focus on their tasks
5	The system should be lightweight and not require an account to use basic features

2.1 System Architecture

The budget management app uses a simple Model View Controller structure. This architecture helps separate the user interface, business logic, and data management, making the system easier for me to develop and maintain.

Overview of Components:

- **Model:** Handles all data related logic. This includes storing expenses, categories, and budget limits.
- **View:** The interface users interact with. This includes pages or screens for adding expenses, viewing summaries, and checking graphs.
- **Controller:** Manages the flow of the app. It handles user input, updates the model, and selects the correct view to display.

This approach was chosen because it supports clear separation of concerns and makes the system easier to test and scale later.

Chapter 2.2 – Use Case Modelling

Field	Description
Use Case ID	1
Use Case Name	Add Expense
Actor	User

Description	Describes how a user adds a new expense entry to the system by providing details such as title, amount, date, and category
Preconditions	The user has launched the app and is on the "Add Expense" page
Basic Flow	<ol style="list-style-type: none"> 1. User clicks the add button to add a new expense. 2. System displays a form with fields: Title, Amount, Date, Category. 3. User fills in the form and submits. 4. System saves the entry to the Google Sheet and refreshes the view.

Field	Description
Use Case ID	2
Use Case Name	View Expenses
Actor	User
Description	Allows the user to view a complete list of all recorded expenses, optionally filtered by date or category
Preconditions	At least one expense has already been added
Basic Flow	<ol style="list-style-type: none"> 1. User navigates to the Expenses screen. 2. System loads all stored expenses. 3. User optionally uses search or filter. 4. System displays the matching results.

Field	Description
Use Case ID	3

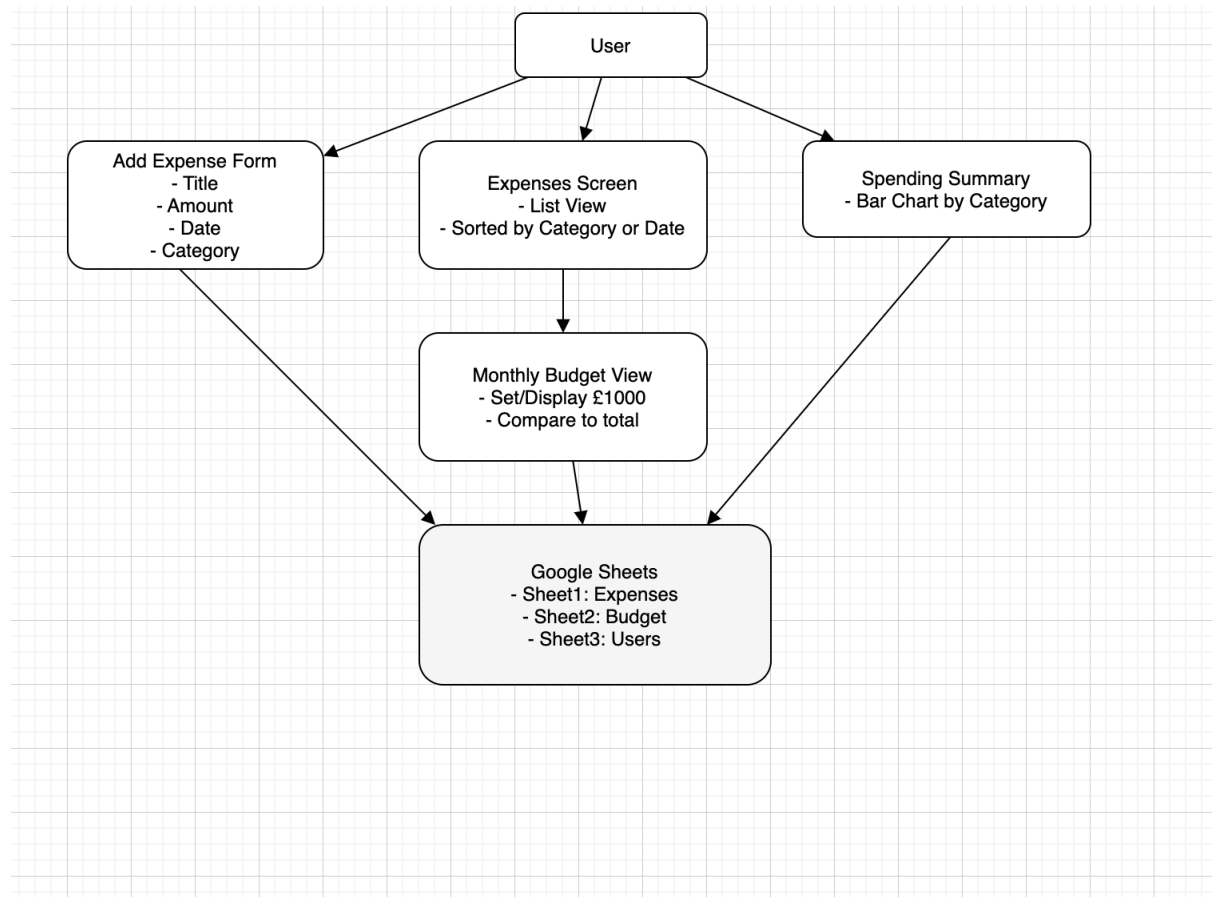
Use Case Name	Set Budget
Actor	User
Description	Enables the user to define a monthly budget limit that the system uses to track their spending progress
Preconditions	User has accessed the "Set Budget" page
Basic Flow	<ol style="list-style-type: none"> 1. User navigates to the budget screen. 2. System displays the current monthly budget value. 3. User enters a new amount. 4. System updates the budget entry in google sheet.

Field	Description
Use Case ID	4
Use Case Name	View Spending Summary
Actor	User
Description	Allows the user to view their spending data presented as a visual breakdown using charts (e.g., pie or bar)
Preconditions	At least one expense must exist in the system
Basic Flow	<ol style="list-style-type: none"> 1. User navigates to the spending summary screen. 2. System retrieves all stored expenses. 3. System groups expenses by category. 4. System calculates total amount spent per category.

	<ol style="list-style-type: none"> 5. System displays this as a visual chart . 6. User views the breakdown and interprets spending patterns.
--	--

Field	Description
Use Case ID	5
Use Case Name	Edit or Delete Expense
Actor	User
Description	Allows the user to modify the details of a past expense or remove it entirely from the system
Preconditions	At least one expense exists in the system
Basic Flow	<ol style="list-style-type: none"> 1. User navigates to the Expenses screen. 2. System displays a list of recorded expenses. 3. User selects a specific expense to edit or delete. 4. System displays editable fields for that expense. 5. User updates the information or confirms deletion. 6. System saves the changes or removes the entry from the database.

2.3 Data Flow Diagram



2.4 Design Decisions

The app is designed with simplicity in mind. A bottom tab bar makes it easy for users to switch between key screens like expenses, budget, and summary. Forms were used for adding and editing expenses to keep the process focused and user-friendly. The use of charts helps users visually understand where their money is going. Overall, the layout is beneficial for non-technical users.

Design and Implementation

This section explains how the budgeting app was planned and built. I used Glide to design and implement the prototype, which connects directly to a Google Sheet for all the data handling.

3.1 Tools Used

I created the app using Glide, which is a platform that lets you create apps from Google Sheets. I chose it because it allowed me to quickly put together a working app with a proper user interface without writing loads of code. It also made it easier to manage data visually and link different parts of the app like forms, charts, and lists.

The backend of the app is handled through a google sheet, which updates automatically whenever users submit data through the app.

3.2 App Data and Structure

The app uses one main table to store all expenses. Each row includes:

- A title for the expense (Rent or Coffee)
- The amount spent
- The date of the expense
- The category (Food, Housing, Travel)

Example of the data used:

Title	Amount	Date	Category
Rent	600	2025-07-01	Housing
Coffee	3.5	2025-07-30	Food
Gym Membership	25	2025-07-15	Health
Transport	20	2025-07-10	Travel
Netflix	10	2025-07-08	Entertainment

There's also a second table that holds the monthly budget value (£1000), which users can update directly in the app.

3.3 Navigation and Screen Layout

The app has a bottom navigation bar with four main tabs:

- Expenses – shows a list of all added expenses with title, date, and category.

- Add Expense – a form screen where users can enter new expenses with fields for title, amount, date, and category.
- Spending Summary – displays a bar chart that visualises how much has been spent in each category.
- Budget – shows the current monthly budget and lets the user change it if needed.

3.4 Components Used

Here's how the app is put together on Glide:

Screen	Components	What it does
Expenses	List view (Collection)	Shows all expenses linked to the sheet
Add Expense	Form + Text/Number/Choice	Inputs data into the Google Sheet
Spending Summary	Bar Chart	Graph showing total spending per category
Budget	Number entry	Lets user set or update the monthly budget

Everything is connected to the same google sheet, and any new input automatically updates the app and the spreadsheet in real time.

3.5 Design Choices and Improvements

To make the app more user-friendly and meet the design criteria, I made a few enhancements:

- Replaced the default side menu with a bottom navigation bar for faster access.
- Used dropdowns for categories to avoid input errors.
- Added a chart so spending data is easier to understand visually.
- Made sure all screens were clearly labelled and easy to navigate.

The layout was kept simple and clean on purpose so users can focus on entering data or checking their spending.

3.6 Summary

Overall, this prototype meets the basic requirements of a working budgeting app. It lets users input expenses, see a breakdown by category, and update their budget. Using Glide helped speed up the process, and all app features are backed by a live google sheet.

The GitHub repository was updated closer to submission as the prototype was primarily created in Glide.

Feature	Test Description	Input Example	Expected Outcome	Result
Add Expense	Add a new expense with all fields completed	Title: Groceries, Amount: 20, Date: Today, Category: Food	Expense is saved and appears in the expenses list	Planned
View Expenses	View all expenses from a specific month	Filter by "This Month"	Only expenses from the selected month are shown	Planned
Set Monthly Budget	Enter a new monthly budget amount	Budget = 1000	Budget value is stored and displayed in budget screen	Planned
View Category Totals	Add expenses across categories and view totals	Food: 20, Transport: 40	System displays correct totals grouped by category	Planned
Add Recurring Expense	Add a repeating monthly rent expense	Title: Rent, Amount: 600, Category: Housing, Repeats: Monthly	Expense is saved with recurring flag	Planned

Edit/Delete Expense	Update an old entry and remove a test one	Edit: Coffee £3.50 to £4.00		
Delete: "Test Item"	Edited item is updated, deleted item is removed	Planned		
View Chart Summary	Open the chart to view breakdown by category	Multiple expenses added	A bar chart displays total spending by category	Planned

This test plan was written to ensure each major feature of the app works as expected. The plan outlines the actions taken, sample inputs, and what the system should do. Testing will be finalised once the working prototype is complete. Each result will be updated accordingly.

Appendix



5:00



Expenses



 Search

Coffee

Food



Rent

Housing



Groceries

Food



Gym Membership

Health



Transport

Travel



Netflix

Entertainment



Expenses



Users



Spending Summ...



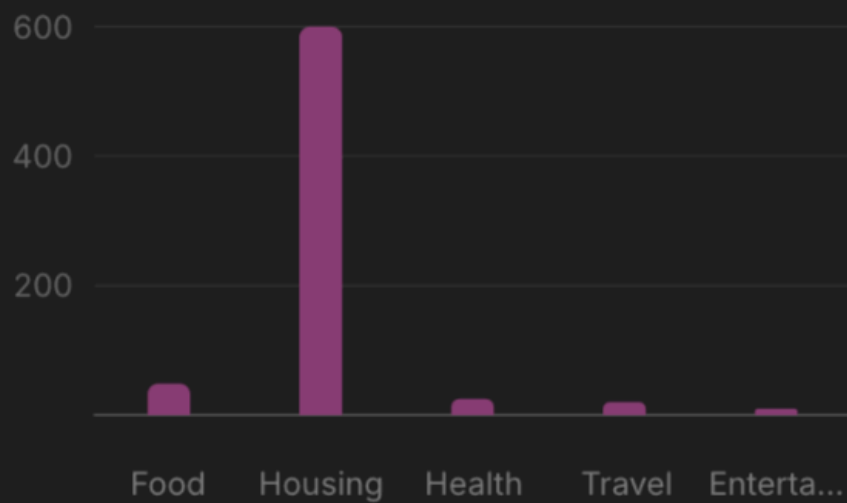
Budget

5:01







Spending by Category

■ Category



	T Title	123 Amount	📅 Date	T Category
1	Coffee	3.5	2025-07-01	Food
2	Rent	600	2025-07-12	Housing
3	Groceries	45	2025-07-23	Food
4	Gym Membership	25	2025-07-15	Health
5	Transport	20	2025-07-18	Travel
6	Netflix	10	2025-07-08	Entertainment
	+ New row			
↑ Import Export Show API Feedback 4 columns × 6 rows				

	T Name	@ Email	🖼️ Photo	T Role	+
1	John Smith	john@gmail.com		Admin	
2	Sarah Lee	sarah@gmail.com		Standard	
3	Alex Thompson	alex@gmail.com		Guest	
4	Ben West	Ben@gmail.com		Viewer	
	+ New row				⌵ ⌵ ⌵ ⌵
↑ Import Export Show API Feedback 4 columns × 4 rows					