1. Business Requirements Document (BRD) – V1

1.1 Background & Problem Statement

Managing personal finances is currently done manually or using generic tools (spreadsheets, bank apps) that don’t reflect the user’s actual categories, habits, or goals. This often leads to:

- Low visibility into **monthly spending vs budget**
- No clear view of **savings progress**
- No simple, focused tool that aligns with the user’s mental model

The goal is to build a **simple, focused budgeting web app** that supports manual entry of income and expenses and provides clear **monthly insights**.

1.2 Objectives

Primary objectives (V1):

1. Allow the user to **manually record** income and expenses in a simple UI.
2. Track spending and savings across **a small, fixed set of categories**:
    - Groceries
    - Rent
    - Family maintenance
    - Fuel
    - Miscellaneous
    - Savings
3. Provide a **monthly summary** showing:
    - Total income
    - Total expenses
    - Net savings (income – expenses)
    - Spend per category
4. Make the app simple enough that it can realistically be used **weekly** without friction.

Secondary objectives (later versions):

- Allow users to define and edit budgets per category.

- Provide basic charts (e.g., category spending pie chart, monthly trend line).
- Support CSV import from bank statements.

1.3 Scope

In scope for V1:

- Single user (no multi-user authentication).
- Manual creation, update, deletion of transactions.
- Fixed category list (6 categories mentioned above).
- Storage using **SQLite**.
- Simple web UI using **Streamlit**.
- Basic monthly insights (current month, possibly selectable month).

Out of scope for V1 (future):

- Multi-user login, roles, permissions.
- Bank API integrations.
- Complex budgeting rules (envelope rolling, carryover).
- Multi-currency support.
- Mobile app packaging (e.g., iOS/Android store).

My view: We're being very strict about scope. This makes it shippable.

1.4 Stakeholders

- **Primary user / Customer**: Individual tracking personal expenses.
- **Product Owner**: Me, in the role of a BA/MBA, defining requirements.
- **Developer**: Also me, wearing the dev hat.
- **Future stakeholders** (later): Potential users (friends, classmates), who may provide feedback.

1.5 Functional Requirements (V1)

FR-1: Manual transaction entry
The system shall allow the user to create a transaction with:

- Date

- Description (text)
- Amount
- Type (Income / Expense)
- Category (Groceries, Rent, Family maintenance, Fuel, Miscellaneous, Savings)

FR-2: Transaction storage
The system shall store all transactions in a **SQLite** database so that data persists across sessions.

FR-3: View transaction list
The system shall display a list/table of transactions with options to:

- Sort or at least show them in descending date order
- Filter by month (or default to current month)

FR-4: Edit / delete transaction
The system shall allow the user to:

- Edit an existing transaction's fields
- Delete a transaction

FR-5: Monthly summaries view
The system shall provide a **Monthly Insights** view that shows, for a selected month:

- Total income
- Total expenses
- Net amount (income – expenses)
- Total spent per category
- Total "Savings" (sum of all transactions categorized as Savings)

FR-6: Category-based aggregation
The system shall calculate spending per category by summing transaction amounts grouped by category for the selected month.

FR-7: Basic visualization (optional in V1.1)
The system may display:

- A bar chart or pie chart of spending per category for the selected month.

1.6 Non-functional Requirements (NFRs)

NFR-1: Usability

- The app shall be usable by a non-technical person with minimal instructions.
- Form labels and buttons should be self-explanatory.

NFR-2: Performance

- For up to 5,000 transactions, page loads and summaries should feel instantaneous (under 1 second perceived time on a typical laptop).

NFR-3: Portability / Setup

- The app shall run locally with:
  - Python installed
  - A simple requirements.txt
  - A single command like streamlit run app.py

NFR-4: Reliability

- Data shall be persisted in an SQLite database file (e.g., budget.db) in the project directory.
- The app should not lose data between restarts unless the file is manually deleted.

NFR-5: Maintainability

- Code shall be organized into:
  - app.py (Streamlit UI)
  - db.py or similar (database access functions)
  - models.py (if needed, to define domain logic)
- Clear comments and a README.md describing how to run the project.

1.7 Assumptions

- Single user; no need for login/authentication.
- All amounts are in one currency.
- The user is comfortable manually entering transactions weekly.
- The time zone and locale are not critical (we'll use local machine settings and simple date handling).

1.8 Risks & Constraints

- **Risk**: User may stop entering data after a few weeks if the UI is too clunky.
    - *Mitigation*: Keep form simple, pre-fill today's date, limited categories.
- **Risk**: Developer (me) is new to Python/Streamlit/SQLite.
    - *Mitigation*: Start with a very small vertical slice (add & list transactions) before adding charts.
- **Constraint**: No backend server beyond what Streamlit provides; app runs locally for now.