Software Requirements Specification (SRS)
Expense Tracker Mobile Application

Prepared by: [Your Name]

August 11, 2025

Contents

1 Introduction

1.1 Purpose

The purpose of this SRS is to document the requirements for a mobile-first expense tracker application designed for Indian users, focusing on UPI-based transactions. The system will help users track expenses, receive alerts, and enforce budget limits, ensuring essential fixed costs are always covered.

1.2 Scope

The solution will:

- Track and categorize expenses from multiple UPI apps using the Account Aggregator (AA) framework.
- Alert users when budgets are exceeded.
- Enforce limits to prevent overspending into fixed cost reserves.
- Support fund separation for different categories (e.g., rent, subscriptions).

Secondary goals:

- Optional transaction blocking if fixed cost funds are at risk.
- Expense analytics and trends.

1.3 Definitions, Acronyms, and Abbreviations

- **UPI** - Unified Payments Interface
- **AA** - Account Aggregator
- **SRS** - Software Requirements Specification
- **FR** - Functional Requirement
- **NFR** - Non-Functional Requirement

1.4 References

- NPCI UPI Framework Documentation
- RBI Account Aggregator Guidelines

2 Overall Description

2.1 Product Perspective

The product will be a standalone mobile application that integrates with the AA framework to fetch transaction data, process it, and present it to the user with alerts and budget enforcement.

2.2 User Characteristics

- Age group: 18-30
- College students and early career professionals
- Familiar with smartphones and UPI apps

2.3 Constraints

- Bank SMS and email are unreliable; only AA framework is considered dependable.
- SBI bank account as primary integration example.
- Mobile-first solution.

3 Functional Requirements (FRs)

1. Register new users.
2. Authenticate users via secure login.
3. Allow users to set budget categories and limits.
4. Fetch transactions from AA framework.
5. Categorize transactions automatically.
6. Store transactions in a secure database.
7. Notify users when budgets are exceeded.
8. Allow fund separation for fixed costs.
9. View expense analytics.
10. Ensure notifications are sent at any time.
11. Maintain service continuity even if user auth expires temporarily.

4 Non-Functional Requirements (NFRs)

- **Performance:** Transactions fetched within 5 seconds of availability.
- **Reliability:** Background service uptime $> 99\%$.
- **Security:** End-to-end encryption for all transaction data.
- **Usability:** Simple, intuitive UI designed for mobile screens.
- **Compliance:** Follow RBI and NPCI guidelines.

5 System Architecture (High Level)

5.1 Components

- Mobile App (Frontend)
- Backend API
- Database
- Background Worker Service
- Notification Service
- AA API Integration Layer

5.2 Flow

1. User registers and sets budgets.
2. Background service fetches transactions via AA.
3. Transactions are categorized and stored.
4. Budget rules are checked, alerts sent if exceeded.
5. User views analytics on demand.

5.3 [Diagram Placeholder]

6 Future Scope

- AI-based expense categorization.
- Integration with investment tracking.
- Cross-platform web dashboard.