% Personal Finance App SRS

Software Requirements Specification (SRS)

Project Name: Personal Finance Manager – SMS-Based Tracker Prepared for: Personal Use (Android Device, React Native CLI) Prepared by: Atharv Tambekar Date: 01-10-2025 Version: 1.0

---

1. Introduction

1.1 Purpose The purpose of this application is to provide a personal finance management system that automatically tracks transactions from Google Pay, UPI, and bank notifications using SMS parsing, allows manual categorization, and provides insights through dashboards and charts. The app is designed for personal use on Android devices.

1.2 Scope The app will: - Automatically read past and incoming transaction SMS. - Categorize transactions via floating overlay popup for immediate input. - Provide filters and search by date, category, and transaction type. - Display analytics including total income/expenses and category-wise breakdown. - Allow backup and restore of transaction data in JSON/CSV format.

The app will not: - Include bill reminders or alert notifications. - Include dark mode or other visual theming (except system default).

---

1. Overall Description

2.1 Product Perspective Standalone personal finance app with SMS parsing implemented using Java native modules on Android.

2.2 User Characteristics - Primary user is the app owner. - Familiarity with Android apps.

2.3 Operating Environment - Android 8.0 or above. - React Native CLI development environment.

2.4 Design & Implementation Constraints - Must request READ_SMS and RECEIVE_SMS permissions. - Floating overlay requires SYSTEM_ALERT_WINDOW permission. - Data stored locally (SQLite / AsyncStorage) and optionally exported.

---

1. Functional Requirements

3.1 Transaction Capture Layer 1. SMS History Scan 2. Real-Time SMS Listener 3. Floating Overlay Pop-up 4. Manual Transaction Entry

3.2 Transaction Data Processing - Identify transaction type: Debit / Credit. - Extract amount, merchant / VPA / bank name, reference number, date, raw SMS. - Assign category. - Optional notes/tags.

3.3 Filtering & Views - By Date: Daily, Weekly, Monthly, Custom Range. - By Category: Food, Rent, Travel, Shopping, Bills. - By Type: Debit or Credit. - Search by merchant, keyword, or reference ID.

3.4 Analytics & Insights - Total income, total expense, net balance. - Category-wise spend: Pie/Bar charts, Trend line.

3.5 Backup & Restore - Export transactions in JSON or CSV. - Import JSON/CSV and merge with existing data.

---

1. Non-Functional Requirements
2. Performance: Real-time SMS parsing within 2-3 seconds.
3. Security: Local storage not accessible to other apps.
4. Usability: Floating overlay intuitive, one-tap category assignment.
5. Reliability: Handle multiple SMS formats, avoid duplicates.
6. Scalability: Support thousands of transactions.
7. Maintainability: Modular React Native + Java architecture.

---

1. System Architecture
2. Presentation Layer (React Native): UI screens, floating overlay.
3. Application Layer: JS functions for filtering, analytics, storage.
4. SMS Module (Native Java): BroadcastReceiver, history scan, parser.
5. Data Layer: Local storage + Backup/Restore.

---

1. Use Cases 6.1 Scan Past SMS 6.2 Real-Time Transaction Categorization 6.3 Filtering Transactions 6.4 Viewing Analytics 6.5 Backup & Restore

---

1. UI Screens
2. Dashboard
3. Transaction List
4. Floating Overlay Popup
5. Analytics
6. Backup & Restore

---

1. Permissions Required (Android)
2. READ_SMS
3. RECEIVE_SMS
4. SYSTEM_ALERT_WINDOW

---

1. Assumptions
2. User grants SMS & overlay permissions.
3. Android 8.0 or higher.

4. Only Google Pay / UPI / bank SMS are auto-parsed.
5. Personal use only.

---

1. Glossary
2. UPI: Unified Payments Interface
3. VPA: Virtual Payment Address
4. Debit: Outgoing money
5. Credit: Incoming money
6. Floating Overlay: Draggable window appearing on top of other apps

---

Next Steps - Design wireframes / screen flow - Define database schema - Implement SMS parsing module - Implement floating overlay popup and categorization UI - Build filtering & analytics screens - Add backup / restore functionality