Software Requirements Specification

Wallee - Personal Finance Mobile App

Team Members

Emma Bahr ebahr2022@my.fit.edu
Matteo Caruso mcaruso2023@my.fit.edu
Joshua Cajuste jcajuste2022@my.fit.edu
Kyle Gibson kgibson2021@my.fit.edu

Client & Faculty Advisor

Dr. Siddhartha Bhattacharyya sbhattacharyya@fit.edu
Doug Gibson

Meeting Dates with Clients

Meeting 1 with Mr. Gibson: September 15, 2025
Meeting 1 Dr. Bhattacharyya: September 26, 2025

1.0 Introduction

1.1 Purpose

The purpose of this document is to specify the requirements for Wallee, a personal finance and budgeting application. Wallee is designed to help users track their income, expenses, and savings goals by securely connecting to their financial accounts and providing actionable insights. The application integrates automated bi-weekly budgeting, expense categorization, and an AI-power assistant to provide personalized guidance and education.

This requirements document defines what the system should do, specifying the expected features, behaviors, and performance criteria. It is intended for developers, testers, advisors, and clients to establish a shared understanding of the system's functionality before design and implementation.

1.2 Scope

Wallee will be developed as a cross-platform mobile application using Flutter with backend services implemented in Node.js and Python. The system will integrate securely with financial institutions via Plaid and similar providers, ensuring accurate real-time data synchronization. Key features:

- Secure bank account linking and real-time transactions updates
- Automated transaction categorization with manual adjustments
- Paycheck based bi-weekly budgeting and safe-to-spend summaries
- Short-term and long-term savings goal creation and tracking
- AI-powered financial assistant for proactive tips and education
- Visualization of spending and budget comparisons in table and graphs

The application will serve students, freelancers, young professionals, retirees, and households. Wallee's focus is on automation, personalization, and proactive financial support, differentiating it from static tools such as Mint or PocketGuard.

1.3 Definitions, Acronyms, and Abbreviations

- Safe-to-Spend: Amount left over after accounting for recurring bills and saving goals.
- AI Assistant: Chatbot powered by OpenAI/spaCy that provides financial guidance
- NLU: Natural Language Understanding for chatbot functionality
- OAuth 2.0: Secure authentication standard for bank integration
- Node.js(TS): Backend runtime for API/webhooks
- PostgreSQL: Core ledger and budgeting database
- FastAPI: Lightweight Python framework for analytics and forecasting services.
- AES Encryption: Industry-standard encryption for sensitive data storage.

1.4 Overview

The remainder of this document provides an overall description of Wallee, a breakdown of functional, interface, and performance requirements, and assumptions and dependencies. Together, these sections define the expected behaviors and constraints of the system.

2.0 Overall Description

2.1 Product Perspective

Wallee builds on existing concepts of budgeting tools but ass unique features that distinguish it from competitors:

- Paycheck-based recalculations that adapt dynamically to overspending or income changes
- AI-driven insights that explain financial data in simple language and recommend actionable steps
- Tailored experiences for different users (students, freelancers, retirees, couples).

2.2 Product Functions

- Bank Account Linking: Secure connection to financial accounts
- Expense Categorization: Automatic and customizable classification of transactions
- Budget Generation: Bi-weekly budgets aligned with pay periods
- Safe-to-Spend tracking: Real time calculations of discretionary funds
- Financial Goals: Creation, tracking, and adjustments of savings goals
- Reports & Summaries: Bi-weekly, weekly, and monthly breakdowns
- AI Assistance: Chatbot providing tips, reminders, and financial literacy support

2.3 User Characteristics

- Students - Manage tuition, rent, and discretionary spending
- Freelancers - Adjust budgets around irregular income and tax payments
- Young Professionals - Track recurring bills, savings, and lifestyle expenses
- Couples/Retirees - Share budgets and manage household or retirement goals

All users are expected to have smartphone literacy and access to the internet.

2.4 Assumptions and Dependencies

- Banking APIs remain available and reliable
- Users consent to data sharing for personalized insights
- Mobile devices run on current iOS/Android versions compatible with Flutter
- Internet access is available for synchronization
- Third-party AI services maintain uptime and reasonable latency.
- Dependence on app store policies (Google Play / Apple App Store approval)
- Assumptions about user financial literacy level
- Dependencies on mobile device features (push notifications, biometrics)
- Dependencies on third-party services (Plaid availability, OpenAI API rate limits, hosting uptime like AWS/GCP)

3.0 Specific Requirements

3.1 Functional Requirements

3.1.1 Bank Integration

- 3.1.1.1 The system shall allow users to securely link bank accounts via Plaid
- 3.1.1.2 The system shall update transactions and balances in real time
- 3.1.1.3 The system shall encrypt all sensitive financial data using AES
- 3.1.1.4 The system shall handle errors if API connectivity fails

3.1.2 Expense Categorization

- 3.1.2.1 The system shall automatically categorize transactions
- 3.1.2.2 The system shall allow users to reassign or rename categories
- 3.1.2.3 The system shall learn from user overrides to improve future categorizations
- 3.1.2.4 The system shall maintain a history of categorized transactions for reporting

3.1.3 Budgeting Management

- 3.1.3.1 The system shall generate bi-weekly budgets based on income and recurring expenses
- 3.1.3.2 The system shall display a safe-to-spend amount after each paycheck
- 3.1.3.3 The system shall adjust budgets when overspending occurs
- 3.1.3.4 The system shall notify users when spending nears or exceeds category limits

3.1.4 Savings Goals

- 3.1.4.1 The system shall allow creation of savings goals with amounts and deadlines
- 3.1.4.2 The system shall track progress toward goals in real time
- 3.1.4.3 The system shall suggest adjustments if progress falls behind
- 3.1.4.4 The system shall support multiple simultaneous goals

3.1.5 Reports and Visualization

- 3.1.5.1 The system shall provide weekly, bi-weekly, and monthly financial summaries
- 3.1.5.2 The reports shall include budgeted, spent, and remaining balances in tables

- 3.1.5.3 The system shall display visual graphs comparing budget vs. actual spending
- 3.1.5.4 The system shall highlight overspending trends with color-coded alerts

3.1.6 AI Financial Assistant

- 3.1.6.1 The chatbot shall answer queries in natural language
- 3.1.6.2 The chatbot shall provide personalized financial tips
- 3.1.6.3 The chatbot shall send reminders about bills and goals
- 3.1.6.4 The chatbot shall provide financial literacy education modules

3.2 Interface Requirements

3.2.1 Touchscreen User Interface

- The system shall accept input via touchscreen
- The system shall support text entry through mobile keyboards
- The system shall allow biometric authentication (FaceID/TouchID) where available

3.2.2 Graphical User Interface

- 3.2.2.1 The system shall provide a personalized dashboard with budgets, goals, and safe-to-spend
- 3.2.2.2 Reports shall be displayed in both tables and graphs
- 3.2.2.3 The system shall allow switching between daily, weekly, and monthly views
- 3.2.2.4 The system shall provide customization of dashboard widgets

3.2.3 Ergonomics

- 3.2.3.1 The system shall notify users of overspending, upcoming bills, and progress updates
- 3.2.3.2 Notifications shall be configurable
- 3.2.3.3 The system shall provide quick access widgets for safe-to-spend and AI assistant

3.3 Performance Requirements

- The system shall refresh financial data within 5 seconds of API requests
- The system shall support up to 10,000 transactions per user without slowdown
- The system shall handle up to 1,000 concurrent users without service degradation
- The AI chatbot shall provide responses within 2 seconds for standard queries
- The categorization algorithm shall achieve at least 90% accuracy
- The system shall maintain 99.5% uptime excluding planned maintenance

4. Use Cases

4.1 Use Case 1: Bank Account Linking

Actor: User, Plaid API
Preconditions: User has valid bank credentials

Main Flow:
1. User selects “Link Bank Account”
2. User chooses bank provider
3. Credentials entered securely
4. System imports transaction

Success Scenario: Transactions appear in the dashboard
Alternative Scenario: API error = system prompts retry

4.2 Use Case 2: Expense Categorization

Actor: User, System
Preconditions: Linked account with transactions

Main Flow:
1. User views transaction list
2. System auto-assigns categories
3. User edits if needed

Success Scenario: Transactions are categorized correctly
Alternative Scenario: Unknown category assigned = “Uncategorized” with user prompt.

4.3 Use Case 3: Budget Management

Actor: User, System
Preconditions: Linked income source

Main Flow:
1. User creates a budget
2. System assigns spending limits per category
3. Safe-to-Spend calculated after each paycheck

Success Scenario: Budget updates correctly and reflects spending
Alternative Scenario: Overspending = red warning alert displayed

4.4 Use Case 4: Savings Goals

Actor: User
Preconditions: Budget Created

Main Flow:
1. User sets goal amount and deadline
2. System tracks progress from income vs. expenses
3. Progress visualized in dashboard

Success Scenario: Goal progress shows with timeline
Alternative Scenario: Falling behind = suggestion to adjust budget

4.5 Use Case 5: AI Assistant Query

Actor: User, AI Assistant
Preconditions: Active session

Main Flow:
1. User types “What’s my safe-tospend?”
2. AI fetches current budget data
3. System responds in natural language

Success Scenario: AI replies with accurate safe-to-spend amount
Alternative Scenario: API timeout = fallback message “Unable to fetch now”

5.0 Success Scenarios Summary

- Bank Linking: Successful import of transactions
- Categorization: 90% accuracy, editable by users
- Budgeting: Real-time safe-to-spend visible
- Goals: Progress tracked with alerts
- AI Assistant: Responses within 2 seconds