Unified Smart Wallet with Budgeting and Insights

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Functionality 4

2.1 <Functional Requirement One> 5

3. Usability 5

3.1 <Usability Requirement One> 5

4. Reliability 5

4.1 <Reliability Requirement One> 5

5. Performance 5

5.1 <Performance Requirement One> 6

6. Supportability 6

6.1 <Supportability Requirement One> 6

7. Design Constraints 6

7.1 <Design Constraint One> 6

8. Online User Documentation and Help System Requirements 6

9. Purchased Components 6

10. Interfaces 6

10.1 User Interfaces 6

10.2 Hardware Interfaces 6

10.3 Software Interfaces 6

10.4 Communications Interfaces 6

11. Licensing Requirements 7

12. Legal, Copyright, and Other Notices 7

13. Applicable Standards 7

# Introduction

The Unified Smart Wallet with Budgeting and Insights concept targets to make the process of handling one’s finances easier and more effective by incorporating features such as budgeting, tracking of expenses, linking with the wallet, and scheduling of the spending reports. This document states the other needs that were not illustrated in the system’s use cases which include security, performance, and system constraints. The following specifications are put in place to ensure that the system achieves its intended purpose and objectives while at the same time being easily usable, secure, and flexible.

## Purpose

[Specify the purpose of this **Supplementary Specification.**]

## Scope

[A brief description of the scope of this **Supplementary Specification**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]

## Definitions, Acronyms, and Abbreviations

* PCI DSS: Payment Card Industry Data Security Standard
* API: Application Programming Interface
* UI/UX: User Interface/User Experience
* MTBF: Mean Time Between Failures

## References

[This subsection provides a complete list of all documents referenced elsewhere in the **Supplementary Specification**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

## Overview

This document contains additional details for the Unified Smart Wallet with Budgeting and Insights project. This outlines the system's functionality, security, performance, and design constraints for additional context to the use-case descriptions. These specifications are designed to guarantee that the wallet is easy to use, scalable, and secure, and offers useful features such as budgeting, money management, and transaction tracking capabilities.

# Functionality

## Functional Requirement One: Budget Management

* Add customizable spending categories for better tracking.
* Allow spending alerts to be set for specific categories (e.g., Food, Travel).
* Notifications: Alert users when they exceed set category budgets.

## Functional Requirement Two: Transaction Tracking

* **Manual Transaction Entry:** Make it possible for the users to input information about the transactions they made offline in the given month (limit: 50 transactions).
* **Digital Wallet Sync:** Multiple wallet connection options.

## Functional Requirement Three: Spending Insights Report

* **Report Generation:** Create monthly and on-demand **spending reports**.
* **Export Options:** Support PDF and **CSV export** (future expansion).
* **Data Analysis:** Show top spending categories and monthly trends.

## Functional Requirement Four: Wallet Integration

* **Multi-Wallet Support:** The proposed enhancements comprise the capability to include several wallets whereas there is only one active wallet at a given time.
* **Manual Wallet Switching:** Allow the user to switch between wallets after they have already authenticated themselves.

## Functional Requirement Five: Notifications Management

* **Custom Notifications Settings:** Allows the users to set notifications frequency, spending alerts, and monthly summaries.

## Functional Requirement Six: Money Management

* **Send Money:** Allow users to securely transfer funds to other wallets using email, phone numbers, or wallet IDs.
* **Receive Money:** Enable users to receive funds from other wallets or external accounts.
* **Transaction History:** Record all money transfers for tracking and reporting.
* **Transfer Limits:** Include daily and monthly limits for sending and receiving money based on subscription plans.
* **Security:** Require PIN verification or biometric authentication for all transactions.

# Usability

## Usability Requirement One

The system should have a user-friendly interface that follows the current UI/UX guidelines and should be accessible to users with visual disabilities as well as should be supportive of multiple languages.

## Simplified User Flow for Money Management

Ensure sending and receiving money involves a minimal number of steps (e.g., recipient details, amount, PIN verification). and providing clear feedback during each step (e.g., confirmation for successful transfers or errors).

## Accessibility

Add support for visual and auditory impairments, such as larger text options, high-contrast modes, and voice commands for money management.

## Error Handling

Clearly communicate errors during money transfers, such as insufficient balance, invalid recipient details, or failed transactions, with actionable next steps.

## Transaction History Filters

Allow users to filter transactions (e.g., by date, amount, type: sent/received, or categories like Food/Transport).

# Reliability

## Consistent Terminology:

* Ensure terms like “Send Money,” “Receive Money,” and “Wallet Switching” are consistently used across the document.
* Avoid overly technical jargon; replace with user-friendly terms where possible.

## Grouping and Organization:

* Group related features under clear headings (e.g., **Budgeting features**, **Money Management Features, Security Features**) for easier navigation.

## Step-by-Step Examples:

* Provide examples for workflow for critical features like sending money, receiving money, or generating spending reports.

## Tooltips and Help Guides:

* Mention the addition of in-app tooltips for complex features (e.g., what is a “Transaction Pin”?)

## Step-by-Step Examples:

* Provide examples for workflow for critical features like sending money, receiving money, or generating spending reports.

# Performance

This section specifies the performance requirements for the Unified Smart Wallet system to ensure scalability, responsiveness, and user satisfaction.

## Response Time

* **Transaction Processing:** Money transfers (send/receive) must be completed within 5 seconds for domestic transactions.
* **Budget Updates:** Adding or updating budgets must reflect changes in under 2 seconds.
* **Notification Delivery:** Notifications for transactions and budget alerts must be sent within 5 seconds of a triggering event.

## Scalability

* The system must support up to 50,000 concurrent users without performance degradation.
* **Scalable Infrastructure:** The backend must be designed to handle up to 1 million monthly active users, with room for future growth.

## System Uptime

* The system must maintain 99.9% uptime, excluding scheduled maintenance.
* Any downtime must not exceed 4 hours per month, including updates and unexpected outages.

## Data Handling

* **Transaction History:** The system must process up to 10,000 transactions per minute without delays.
* **Reports:** Monthly spending reports for up to 100,000 users must generate in under 10 seconds.

## Resource Efficiency

* The mobile app must run efficiently on devices with at least 2GB of RAM.
* Server CPU utilization must remain below 70% under peak load conditions.

# Supportability

This section defines the requirements for maintaining and supporting the Unified Smart Wallet system.

## Administration

* **Role-Based Access Control:** Administrators must be able to manage user roles (e.g., Admin, Standard User) and permissions.
* **Dispute Resolution:** Admins must have tools to handle money transfer disputes, such as reversals and fraud detection.

## Monitoring and Alerts

* **Real-Time Monitoring:** The system must provide real-time monitoring dashboards for server health, transaction logs, and system activity.
* **Alerts:** Automatic alerts must notify administrators of system errors, high server load, or suspicious activity.

## System Maintenance

* **Scheduled Maintenance:** Maintenance tasks must be scheduled during off-peak hours and notify users 24 hours in advance.
* **Backup and Recovery:** Automatic daily backups of user data and system configurations must be performed, with recovery times under 15 minutes.

## Documentation and User Support

* **Technical Documentation:** Provide complete documentation for developers, including API references, architecture diagrams, and deployment guides.
* **User Help Guides:** Include in-app tutorials, FAQs, and troubleshooting steps for all core features (e.g., sending money, managing budgets).

## Compatibility

* **Cross-Platform Support:** Ensure compatibility with Android (9.0+), iOS (12+), and major web browsers.
* **API Upgrades:** Allow seamless integration with updated versions of wallet APIs (e.g., Google Wallet, Apple Wallet).

## Customer Support

* **Support Channels:** Provide 24/7 email and chat support for users.
* **Response Time: S**upport requests must receive responses within 1 hour during business hours and 24 hours for non-urgent issues.

# Design Constraints

This section outlines the design constraints that must be adhered to during the development of the Unified Smart Wallet with Budgeting and Insights. These constraints ensure the system is scalable, secure, and compliant with industry standards.

## Software Language

* The application must be developed using Java, following Object-Oriented Programming (OOP) principles to ensure modularity, reusability, and scalability.

## Software Process Requirements

* The project must follow the Agile Development Process, ensuring incremental delivery and incorporating user feedback at every stage.
* Use the Model-View-Controller (MVC) architecture to separate concerns between the user interface, business logic, and data layers.

## Developmental Tools

* integrated Development Environment (IDE): Use IntelliJ IDEA or Eclipse as the primary IDE.
* Version Control: Employ Git for version control and GitHub or GitLab for repository management.
* Build Tools: Use Maven or Gradle for dependency management and project builds.

## Architectural and Design Constraints

* The system must support multi-device compatibility, including Android (9.0+) and iOS (12.0+) for mobile users.
* Ensure the design supports high scalability to handle up to 50,000 concurrent users without performance degradation.
* All sensitive data (e.g., transactions, user details) must be processed and stored securely, complying with PCI DSS and GDPR standards.

## Purchased Components

* APIs: Integrate with third-party APIs, such as Google Wallet API and Apple Wallet API, for seamless wallet linking.
* Cloud Storage: Use AWS S3 or Azure Storage for secure and reliable data backups.

## Constraints on Money Management Feature

* Money transfers must complete within 5 seconds for domestic transactions.
* Include daily and monthly transfer limits per user, customizable based on subscription tiers.
* Require PIN or biometric authentication for every money transfer.

## Performance and Reliability

* The system must ensure 99.9% uptime, excluding scheduled maintenance.
* Notifications for money transfers and budget alerts must be delivered within 5 seconds of a triggering event.

# Purchased Components

* Google Wallet and Apple Wallet APIs (licensed as required).
* Cloud services for storage and backup (e.g., AWS or Azure).

# Interfaces

## User Interfaces

Responsive web and mobile interfaces designed using modern UI frameworks like React or Flutter.

## Hardware Interfaces

The application must run on mobile devices supporting Android (version 9+) and iOS (version 12+).

## Software Interfaces

* Integration with third-party financial APIs.
* Payment processing gateways for in-app purchases.

## Communications Interfaces

* Use HTTPS and JSON APIs for secure data exchange.

# Licensing Requirements

* Users must accept an End-User License Agreement (EULA) before accessing the application.
* Licensing terms must comply with payment processor agreements.
* Add terms specific to money management operations, ensuring compliance with financial transaction regulations and third-party wallet APIs.

# Legal, Copyright, and Other Notices

Trademark notices for third-party services used must be displayed.

# Applicable Standards

* Java SE 17 (or the latest stable version) for development.
* Object-Oriented Programming (OOP) Standards, focusing on:

1. **Modularity**: Breaking the system into components.
2. **Code Reusability**: Encouraging code reuse through classes and interfaces.
3. **Readability**: Ensuring clean and readable code structure.