Marcus Rappenborg Kjærsgaard
20-11-2025

Software Requirements Specification (SRS) | version 1.0

Multi-Currency Wallet Simulator

1. Introduction and Purpose

This document specifies the functional and non-functional requirements for the Multi-Currency Wallet Simulator. The system is a small web-based application that allows a user to create and manage virtual wallets in different currencies, perform deposits, withdrawals, and currency exchanges, and inspect transaction history.

The application is intended for educational and demonstration purposes only. It does not handle real-world money or connect to any payment providers. The goal of the SRS is to be concise, readable, and precise enough to support formal review, test design, and implementation.

2. System Overview

The system consists of a browser-based frontend, a backend implemented as a web API, a relational database, and an integration with an external public currency exchange rate API.

The user interacts with the system through the frontend to:

- Create wallets in supported currencies
- View existing wallets and their balances
- Perform deposits, withdrawals, and exchanges between wallets
- View a list of past transactions

The backend exposes endpoints that implement all wallet and transaction operations and store data in a database. The system uses an external public API to retrieve exchange rates when converting amounts between different currencies.

3. Functional Behaviour

Supported currencies. The system shall support exactly three currencies: DKK, EUR, and USD. Any attempt to create or use a wallet in another currency shall be rejected as invalid input.

Wallet creation and status. The system shall allow the user to create a new wallet by selecting one of the supported currencies. Each wallet shall maintain a single balance in its currency and a status indicating whether it is active, frozen, or closed. Newly created wallets shall start with status active and an initial balance of zero, unless a non-zero initial deposit is explicitly provided and validated.

Transaction types. The system shall allow the user to perform the following transaction types on an active wallet:

- Deposit: increase the wallet balance by a positive amount
- Withdrawal: decrease the wallet balance by a positive amount not exceeding the current balance
- Exchange: transfer a positive amount from one wallet to another, optionally in a different currency

Validation rules for amounts. For withdrawals and exchanges, the system shall reject any request where the amount is not strictly greater than zero or where the amount exceeds the available balance in the source wallet. In such cases, the balance of the involved wallets shall not be modified.

Currency exchange behaviour. For exchanges between wallets with different currencies, the system shall obtain or use a currency exchange rate and calculate the credited amount in the target wallet's currency. The system shall apply a clear and deterministic rounding strategy to the calculated amounts (for example, rounding to two decimal places), and this behaviour shall be consistent across all operations.

Wallet status and allowed operations. The system shall prevent transactions on wallets whose status is not active. Deposits, withdrawals, and exchanges shall not be allowed on frozen or closed wallets. The system shall allow the wallet status to change between a defined set of states (for example, active to frozen, frozen to active, and active or frozen to closed) and shall prevent any further transactions once a wallet is closed.

Transaction recording and history. Each transaction shall be stored with at least: type, source wallet, target wallet where applicable, amount, resulting balances, and a status indicating whether the transaction was completed or failed. A transaction shall be marked as failed if validation checks or external API calls fail, and in that case no wallet balance shall change.

The system shall provide the user with a view that lists existing wallets and their current balances, and a view of transaction history for a selected wallet. The user shall be able to see, in a clear way, whether a transaction was completed or failed.

4. Error Handling and Constraints

Invalid input. If the user submits invalid input (such as unsupported currency, non-numeric amount, non-positive amount, or reference to a non-existent wallet), the system shall reject the request, shall not modify any balances, and shall return a clear error response.

External API failures. If the external currency exchange API is unavailable, responds with an error, or provides data that cannot be processed, the system shall treat any affected exchange transaction as failed and shall not modify any balances. The system may reuse recently obtained rates from a local cache when available but shall not silently invent or guess exchange rates.

Simulation scope. The system shall not perform any real financial transfers, shall not store sensitive payment information, and shall operate only on simulated wallet data stored in its own database.

5. Non-Functional Aspects

Usability and interface. The system shall provide a simple web-based user interface that can be used on a modern desktop browser without requiring browser extensions. Basic operations such as viewing wallets and performing a single transaction shall be understandable without prior training.

Performance and responsiveness. Under normal development, typical operations (viewing wallets, performing a deposit, withdrawal, or exchange) should complete within a reasonable response time for a local or demonstration environment. The system is not intended for production-scale load but shall be testable with automated functional and performance testing tools.

Architecture and testability. The system shall be implemented using a web frontend, a backend web API, a relational database, and integration with an external public currency exchange rate API, in a way that allows the functional behaviour described in this document to be tested using unit tests, integration tests, API tests, end-to-end UI tests, and stress performance tests.