|  |
| --- |
|  |

|  |
| --- |
| REVISION HISTORY |

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **DESCRIPTION** | **AUTHOR** |
| 1.0 | 22.09.2025 | Initial Prototype | **Zaletajev K.** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| CONTENTS |

[1 SW System Overview 4](#_Toc208787546)

[1.1 Purpose 4](#_Toc208787547)

[1.2 Scope 4](#_Toc208787548)

[1.3 Use-Case Diagram 4](#_Toc208787549)

[1.4 General Constraints 4](#_Toc208787550)

[1.5 Assumptions and Dependencies 4](#_Toc208787551)

[1.6 Acronyms and Abbreviations 4](#_Toc208787552)

[2 SW Functional Requirements 5](#_Toc208787553)

[2.1 Features / Functions to be Implemented 5](#_Toc208787554)

[2.1 Acceptance Criteria 5](#_Toc208787555)

[2.2 Implementation Requirements 5](#_Toc208787556)

[3 SW Non-Functional Requirements 6](#_Toc208787557)

[3.1 Resource Consumption 6](#_Toc208787558)

[3.2 License Issues 6](#_Toc208787559)

[3.3 Coding Standard 6](#_Toc208787560)

[3.4 Modular Design 6](#_Toc208787561)

[3.5 Reliability 6](#_Toc208787562)

[3.6 Portability 6](#_Toc208787563)

[3.7 General Operational Guidelines 6](#_Toc208787564)

[4 SW Design Artifacts 7](#_Toc208787565)

[4.1 CRC Cards (Class–Responsibility–Collaboration) 7](#_Toc208787566)

[4.2 Conceptual UML Diagram (entities & relationships) 7](#_Toc208787567)

# SW System Overview

This SRS outlines the specifications need for a software application designed to facilitate currency conversion services. This application will streamline cashier operations by automating exchange calculations, generating transaction receipts and maintaining digital records for end-of-day reporting. The main objective of this software is to minimize human errors, accelerate customer transaction processing and do make managerial oversight easier

## Purpose

The main purpose of this system is to provide currency conversion transactions within a 2-second timeframe, generate receipts and maintain operation records for further analysis

## Scope

1. Included functionality: transaction data input, automated exchange calculations, transaction creation
2. Excluded functionality: internet banking, tax services,
3. Benefits: reduced human error, faster service, easier management of funds
4. Key features: automized calculations, retrievable receipts, file storage

## Use-Case Diagram

|  |
| --- |
|  |

## General Constraints

*List technical and business constraints such as programming language, operating system, performance limitations, and standards.*

## Assumptions and Dependencies

*State assumptions (e.g., availability of internet, supported devices) and dependencies (e.g., external APIs, hardware).*

## Acronyms and Abbreviations

List all acronyms and abbreviations used in the document along with their explanations.

|  |  |
| --- | --- |
| **Terms Used** | **Description of terms** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# SW Functional Requirements

## 2.1 Features / Functions to be Implemented

Example User Stories:

* As a client, I want to exchange a specific currency at a given rate, so I receive the target amount.
* As a client, I want to receive a receipt of the transaction, so I have proof of the transaction.
* As a cashier, I want to generate the end-of-day report, so managers can see transactions that happened.
* As a cashier, I want to check the currency amount in the reserve before a transaction, so I can ensure that the transaction can happen
* As a manager, I want to change the exchange rate of a specific currency, so I control revenue.

From stated user stories we can identify functions:

* User interactions: entering transaction data, printing receipts.
* Business process: transaction handling, end-of-day reporting.
* System logic: automated calculations of exchanged amounts, transaction logging

## Acceptance Criteria

* Transaction receipts must be created and stored digitally following each completed exchange.
* Exchange amount calculations must maintain precision to two decimal places.
* End-of-day summary reports must contain total transaction count and cumulative exchange values.

## Implementation Requirements

* Transaction data must be saved in CSV format including timestamp, source currency, target currency, and value.
* The application must operate exclusively in command-line mode (CLI).

# SW Non-Functional Requirements

## Resource Consumption

* Exchange operation processing time: ≤ 2 seconds
* Peak memory utilization: ≤ 100 MB
* Daily log file size limit: ≤ 5 MB

## License Issues

* Exclusively standard C++ STL libraries are permitted.
* Proprietary third-party components are prohibited.
* External dependencies are acceptable only with permissive open-source licensing (MIT, Apache-2.0).

## Coding Standard

* All functions and classes require comprehensive documentation comments.
* Unit testing must validate all essential components (e.g., exchange amount calculations).

## Modular Design

The application shall be structured with distinct modules for:

* Exchange computation
* Digital logging
* Report generation
* User interface management

Components must maintain loose coupling and strong cohesion principles.

## Reliability

* **The application must handle incorrect input gracefully without system failure.**
* **Data writing operations must be atomic to prevent file corruption.**
* **System errors must be documented in log files for diagnostic purposes.**

## Portability

* The application must build and execute on Windows 10+ and Ubuntu Linux platforms.
* Consistent inputs must yield consistent results across both operating systems.

## General Operational Guidelines

* The application must demonstrate stability, maintainability, and user-friendliness.
* Daily initialization functionality must be available to begin each business day with a fresh state.
* All activities must be documented for transparency and audit requirements.

# SW Design Artifacts

## CRC Cards (Class–Responsibility–Collaboration)

List the main classes with their responsibilities (action verbs) and collaborators (related classes); keep items concise and implementation-agnostic.

## Conceptual UML Diagram (entities & relationships)

Draw a conceptual class diagram with key entities and their relationships; focus on nouns from User Stories/Use Cases, omit methods and low-level details.