SRS-DLD

S/W Detailed Level Design

Project Name	Currency Exchange System — Release 2		
Block Name	Compile Commander		
Author	**Magnus Jaaska**	**Approver**	
Team	Compile Commander		

This document represents Detailed Level Design (DLD). It describes the detailed system design and implementation plan in alignment with Agile principles. The DLD is updated incrementally with each release to reflect system evolution.

Contents

■ Revision History

Version	Date	Revised contents	Author	Approver
1.0	28.10.2025	**Initial Release 2 DLD; three-layer refactor; interfaces; method responsibilities; traceability.**	Magnus Jaaska	

■ Terms and Abbreviations

Term	Description
UI	User Interface (ConsoleUI)
DLD	Detailed Level Design document
SRS	Software Requirements Specification
DIP	Dependency Inversion Principle
SRP	Single Responsibility Principle

■ References

1. SW Requirements Specification (docs/release-1/SRS.pdf)
2. Lecture 5–7 (Design for Quality, Modularity, Abstraction, Polymorphism)
3. Example DLD Currency Exchange (reference structure)

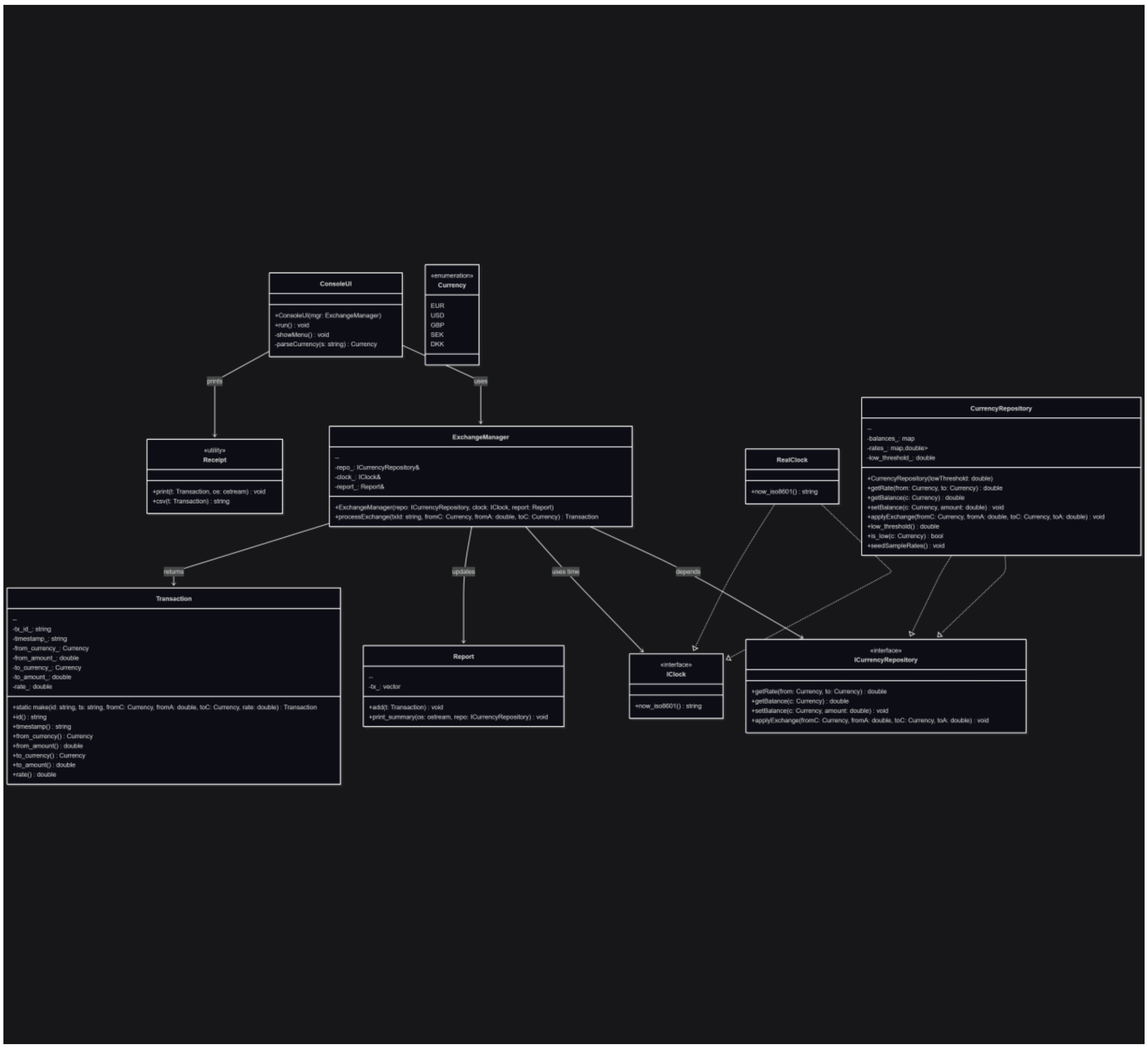
1. Overview

The Currency Exchange System automates currency conversion, verifies reserves before payout, records transactions in memory, and provides a daily summary. This DLD defines the Release 2 design: modular three-layer architecture (UI → Logic → Data), class responsibilities, interfaces, function contracts, constants, operation flow, and SRS traceability. Validation, exception handling, and file-based persistence are planned for Release 3.

2. System Overview / Architectural Context

- **UI:** ConsoleUI — console interaction & formatting only.
- **Logic:** ExchangeManager — coordinates conversion, reserve checks, and reporting.
- **Data:** ICurrencyRepository / CurrencyRepository — in-memory balances & exchange rates.
  Principles: Separation of Concerns, SRP, DIP, encapsulation via header/source split.

3. UML Class Diagram (Technical Design)


ConsoleUI
+ConsoleUI(mgr: ExchangeManager)
+run() : void
-showMenu() : void
-parseCurrency(s: string) : Currency
«enumeration»
Currency
EUR
USD
GBP
SEK
DKK
prints
uses
«utility»
Receipt
+print(t: Transaction, os: ostream) : void
+csv(t: Transaction) : string
ExchangeManager
--
-repo_: ICurrencyRepository&
-clock_: IClock&
-report_: Report&
+ExchangeManager(repo: ICurrencyRepository, clock: IClock, report: Report)
+processExchange(txId: string, fromC: Currency, fromA: double, toC: Currency) : Transaction
RealClock
+now_iso8601() : string
CurrencyRepository
--
-balances_: map
-rates_: map,double>
-low_threshold_: double
+CurrencyRepository(lowThreshold: double)
+getRate(from: Currency, to: Currency) : double
+getBalance(c: Currency) : double
+setBalance(c: Currency, amount: double) : void
+applyExchange(fromC: Currency, fromA: double, toC: Currency, toA: double) : void
+low_threshold() : double
+is_low(c: Currency) : bool
+seedSampleRates() : void
returns
updates
uses time
depends
Transaction
--
-tx_id_: string
-timestamp_: string
-from_currency_: Currency
-from_amount_: double
-to_currency_: Currency
-to_amount_: double
-rate_: double
+static make(id: string, ts: string, fromC: Currency, fromA: double, toC: Currency, rate: double) : Transaction
+id() : string
+timestamp() : string
+from_currency() : Currency
+from_amount() : double
+to_currency() : Currency
+to_amount() : double
+rate() : double
Report
--
-tx_: vector
+add(t: Transaction) : void
+print_summary(os: ostream, repo: ICurrencyRepository) : void
«interface»
IClock
+now_iso8601() : string
«interface»
ICurrencyRepository
+getRate(from: Currency, to: Currency) : double
+getBalance(c: Currency) : double
+setBalance(c: Currency, amount: double) : void
+applyExchange(fromC: Currency, fromA: double, toC: Currency, toA: double) : void

4. Class Specifications

Class	Type	Description	Attributes	Methods
ConsoleUI	UI	Console I/O and formatting; no business logic.		run(), parseCurrency(s), showMenu()
ExchangeManager	Logic	Coordinates exchange; checks reserve; updates report.	repo_: ICurrencyRepository&, clock_: IClock&, report_: Report&	processExchange(txId, fromC, fromA, toC): Transaction
ICurrencyRepository	Interface	Abstracts data access for rates/balances.		getRate(from,to), getBalance(c), setBalance(c,amt), applyExchange(fromC,fromA,toC,toA)
CurrencyRepository	Data	In-memory balances & rates; implements interface.	balances_, rates_, low_threshold_	getRate(), getBalance(), setBalance(), applyExchange(), is_low(c), seedSampleRates()
Transaction	Domain	Immutable exchange record w/ computed to-amount.	tx_id_, timestamp_, from_currency_, from_amount_, to_currency_, to_amount_, rate_	static make(id,ts,fromC,fromA,toC,rate): Transaction
Report	Domain	Aggregates transactions; prints daily summary.	tx_: vector<Transaction>	add(t), print_summary(os, repo)
Receipt	Utility	Formats receipt and CSV line.		print(t, os), csv(t): string
IClock	Interface	Time abstraction.		now_iso8601(): string

RealClock	Service	System clock implementation.	now_iso8601(): string

5. Interfaces and Abstractions

Interface	Purpose	Key Methods	Planned For (Release)
ICurrencyRepository	Decouple logic from storage; enable File/DB later	getRate(), getBalance(), setBalance(), applyExchange()	R2 (implemented)
IClock	Decouple time for tests	now_iso8601()	R2 (implemented)

6. Function Responsibilities

Class	Method	Purpose	Input	Output	Notes
ExchangeManager	processExchange()	Computes payout, checks reserve, applies exchange, records transaction	txId, fromC, fromA, toC	Transaction	Throws on insufficient reserve/bad rate
CurrencyRepository	applyExchange()	Update balances for from/to	fromC, fromA, toC, toA		Encapsulated state change

Report	add()	Append to daily collection	Transaction		Called after success
Report	print_summary()	Emit daily totals/closing balances	ostream, ICurrencyRepository		Presentation kept minimal
Receipt	print()	Human-readable receipt	Transaction, ostream		UI use
Receipt	csv()	CSV transaction line	Transaction	string	For logging later
RealClock	now_iso8601()	Timestamp		string	Local ISO-8601
ConsoleUI	run()	Interactive loop; parse/execute	stdin	stdout	Uses manager; handles errors

7. Operation Flow

1. ConsoleUI reads input (“EUR 100 USD”).
2. Parses → ExchangeManager::processExchange().
3. Manager gets rate from ICurrencyRepository, computes payout.

4. Validates reserves → applies exchange in repo.
5. Creates Transaction, adds to Report.
6. UI prints Receipt; session end → Report::print_summary().

8. Enumerations & Constants

Name	Value / Type	Description
enum class Currency	{EUR, USD, GBP, SEK, DKK}	Supported currencies
LOW_THRESHOLD	double = 100.0	Alert threshold for low balances
INITIAL_BALANCES	per-currency doubles	Startup config for interactive run

9. Validation Rules & Future Work

Rule / Planned Feature	**Description**	**Target**
Input validation & errors	amount > 0, valid currency codes; clear error messages	R3
Strict reserve enforcement	Block payout > reserve; error codes	R3
Exception policy	try/catch at UI; noexcept in Data	R3
File I/O persistence	Save/load rates, tx, reports	R3
Unit tests	Mock IClock & repo; per-layer tests	R3

10. Traceability Matrix

Requirement (SRS)	Class / Method (DLD)	Notes
RQ-01 Convert currency	ExchangeManager::processExchange()	Core conversion via repo rate
RQ-02 Show receipt	Receipt::print(), ConsoleUI::run()	UI formatting only
RQ-03 Maintain reserves	CurrencyRepository::applyExchange()	Encapsulated data update
RQ-04 Daily report	Report::add(), Report::print_summary()	End-of-day summary
RQ-05 Time stamps	RealClock::now_iso8601()	Abstracted time

11. Code Structure and File Mapping

Class / Module	Files
ConsoleUI	include/console_ui.h, src/console_ui.cpp
ExchangeManager	include/exchange_manager.h, src/exchange_manager.cpp
ICurrencyRepository	include/currency_repository.h (interface)
CurrencyRepository	include/currency_repository.h, src/currency_repository.cpp
Transaction	include/transaction.h, src/transaction.cpp
Report	include/report.h, src/report.cpp
Receipt	include/receipt.h, src/receipt.cpp
IClock / RealClock	include/iclock.h, src/iclock.cpp
Entry point	src/main.cpp
Build	Makefile

12. Revision History

Date	Version	Change Summary	Author
28.10.2025	1.0	Initial DLD created from Release 2 refactor	Magnus Jaaska