Currency Exchange System
Release 1 – Fundamentals of C++

Team: Magnus Jaaska (@magna413)

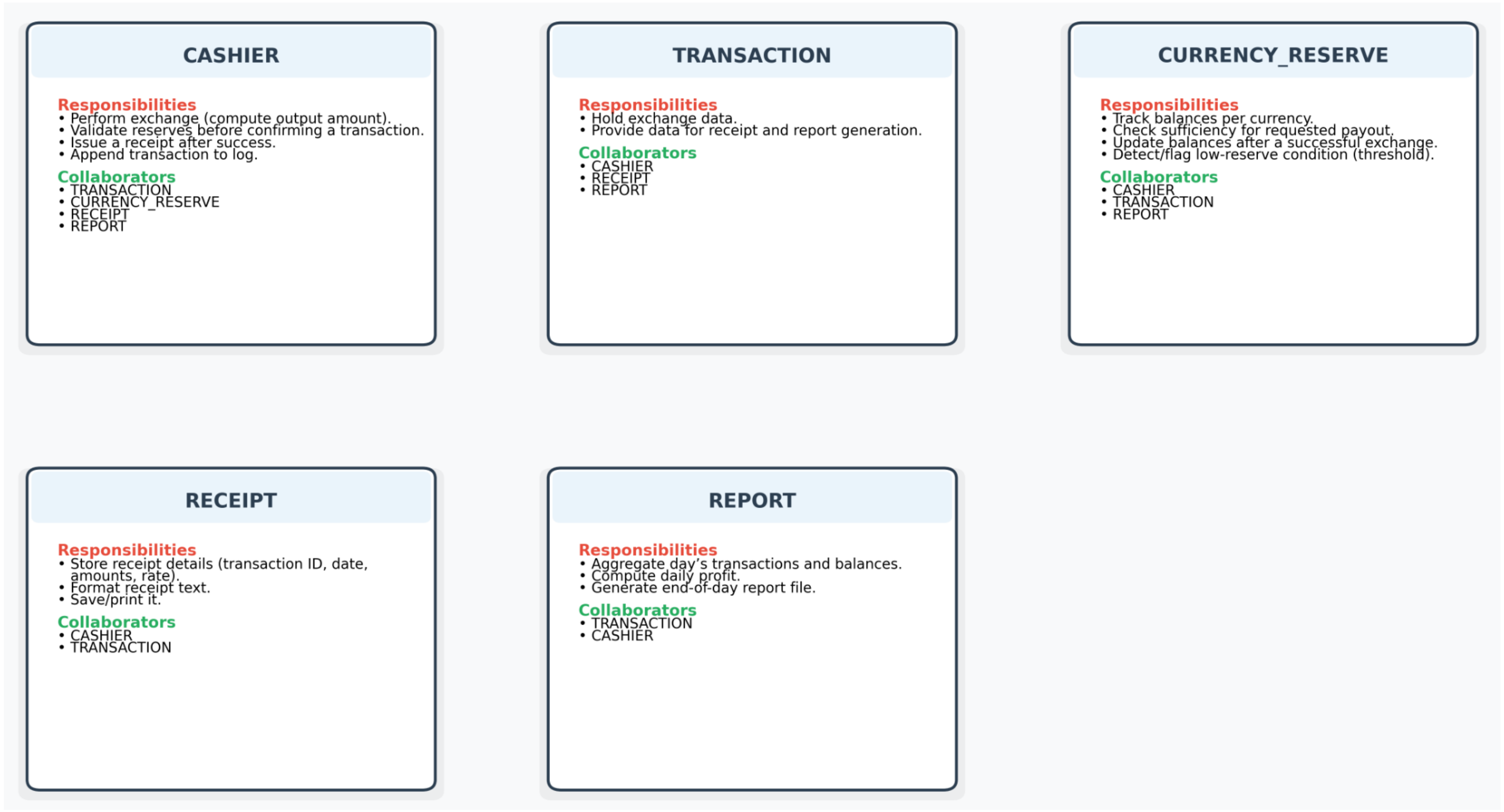
Problem Statement & Scope

- Problem: Small exchange offices rely on manual conversions → error-prone, slow, not auditable.
- Scope (In): Manual rates, Exchange validation, Receipts, Reports, Low-reserve warnings.
- Scope (Out): Live rate feeds, Multi-branch sync, External tax/banking integration.
- Constraints: C++17, CLI, CSV logs, Win10+/Ubuntu22.04, ≤2s per tx.

User Stories

- • US-1: As a Cashier, I want to enter the exchange details so that the system calculates the converted amount and validates reserves.
- • US-2: As a Client, I want to receive a printed receipt so that I have proof of the transaction.
- • US-3: As a Manager, I want a daily report so that I can review totals and profit.
- • US-4: As a Manager, I want to set/update rates so that profitability stays controlled.
- • US-5: As a Cashier, I want low-reserve warnings so that I don’t confirm impossible operations.

CRC Cards Overview



CASHIER

Responsibilities
- Perform exchange (compute output amount).
- Validate reserves before confirming a transaction.
- Issue a receipt after success.
- Append transaction to log.

Collaborators
- TRANSACTION
- CURRENCY_RESERVE
- RECEIPT
- REPORT

TRANSACTION

Responsibilities
- Hold exchange data.
- Provide data for receipt and report generation.

Collaborators
- CASHIER
- RECEIPT
- REPORT

CURRENCY_RESERVE

Responsibilities
- Track balances per currency.
- Check sufficiency for requested payout.
- Update balances after a successful exchange.
- Detect/flag low-reserve condition (threshold).

Collaborators
- CASHIER
- TRANSACTION
- REPORT

RECEIPT

Responsibilities
- Store receipt details (transaction ID, date, amounts, rate).
- Format receipt text.
- Save/print it.

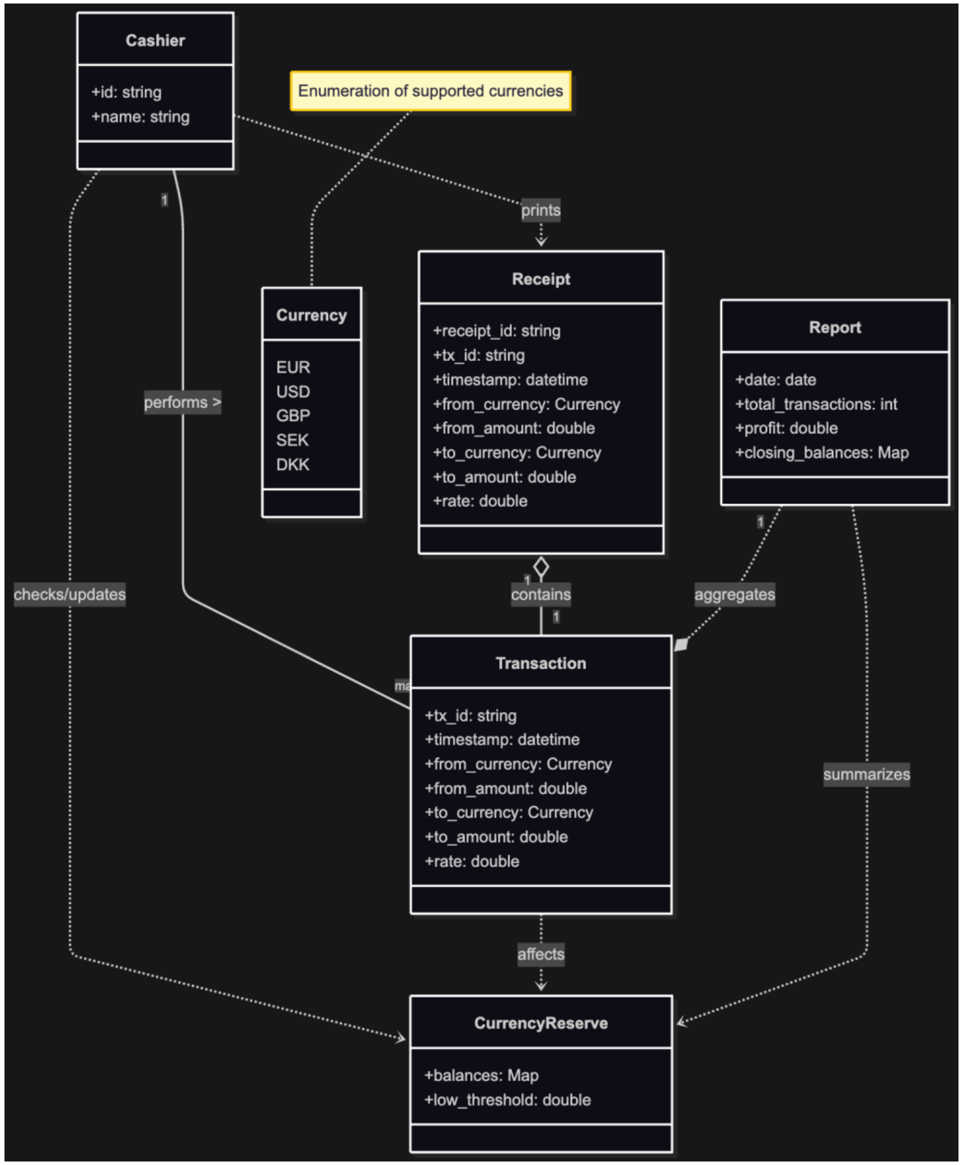
Collaborators
- CASHIER
- TRANSACTION

REPORT

Responsibilities
- Aggregate day’s transactions and balances.
- Compute daily profit.
- Generate end-of-day report file.

Collaborators
- TRANSACTION
- CASHIER

Conceptual Simple Class Diagram



Prototype (C++17, CLI)

- Implemented:
- - enum Currency
- - struct Transaction::make()
- - Cashier::exchange(), print_receipt()
- - CurrencyReserve::can_fulfill(), apply(), is_low()
- - Receipt::print(), csv()
- - Report::add(), print_summary()

Team Roles

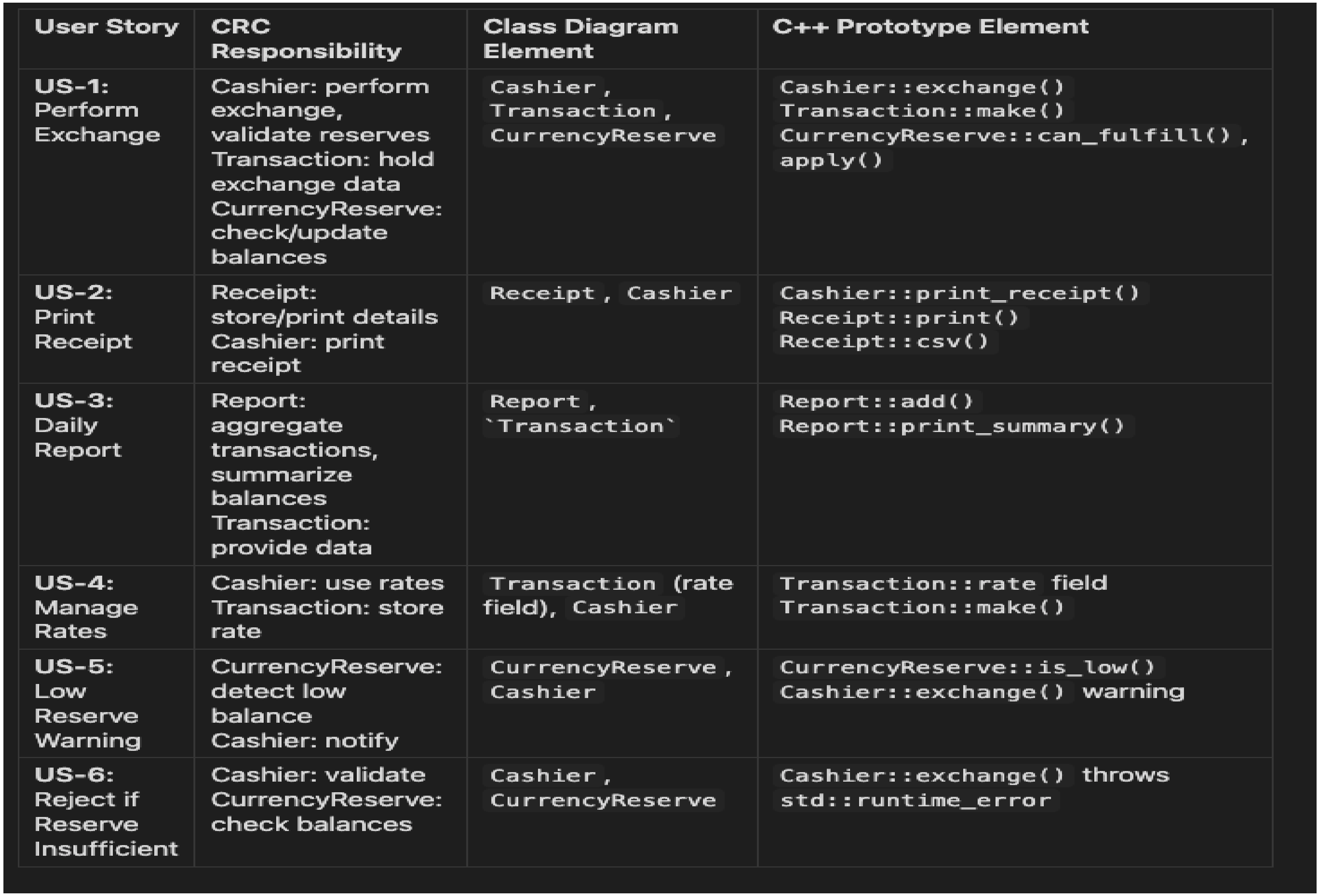
- Requirements specification (SRS)
- CRC cards & UML diagram
- C++ prototype implementation
- Repository setup (README, Makefile, structure)
- Traceability table
- Release 1 presentation & documentation

Lessons Learned

- Easy
  - Writing user stories and mapping them to CRC cards and UML
  - Designing a small but clear system in C++17

- Challenging
  - Keeping traceability consistent across requirements, design, and code
  - Modeling reserve validation and error handling in a simple prototype

Traceability Table (User Story → Design → Code)



User Story	CRC Responsibility	Class Diagram Element	C++ Prototype Element
US-1: Perform Exchange	Cashier: perform exchange, validate reserves Transaction: hold exchange data CurrencyReserve: check/update balances	`Cashier`, `Transaction`, `CurrencyReserve`	`Cashier::exchange()` `Transaction::make()` `CurrencyReserve::can_fulfill()`, `apply()`
US-2: Print Receipt	Receipt: store/print details Cashier: print receipt	`Receipt`, `Cashier`	`Cashier::print_receipt()` `Receipt::print()` `Receipt::csv()`
US-3: Daily Report	Report: aggregate transactions, summarize balances Transaction: provide data	`Report`, `` `Transaction` ``	`Report::add()` `Report::print_summary()`
US-4: Manage Rates	Cashier: use rates Transaction: store rate	`Transaction` (rate field), `Cashier`	`Transaction::rate` field `Transaction::make()`
US-5: Low Reserve Warning	CurrencyReserve: detect low balance Cashier: notify	`CurrencyReserve`, `Cashier`	`CurrencyReserve::is_low()` `Cashier::exchange()` warning
US-6: Reject if Reserve Insufficient	Cashier: validate CurrencyReserve: check balances	`Cashier`, `CurrencyReserve`	`Cashier::exchange()` throws `std::runtime_error`