# Currency Exchange Codebase Overview

## Project Layout

Repository root contains Makefile, README.md, Requirements.txt, and the primary directories data/, docs/, include/, src/, and tests/.

The include/ folder hosts public headers such as include/exchange\_manager.h and include/persistence.h, while src/ contains the corresponding implementations like src/exchange\_manager.cpp and src/persistence.cpp.

## Build Configuration and Entry Point

Makefile (lines 1-35) compiles every source file in src/ using g++ with C++17, strict warnings, and the include/ search path. Targets named all, run, and clean streamline builds and execution.

src/main.cpp (lines 13-92) is the application entry point. It validates command-line options, constructs DataStore with the data/ directory, loads or seeds rates and reserves via defaultReserveBalances and ensureDefaultRates, builds ExchangeOffice, and starts ConsoleUI for terminal interaction.

## Domain Model

include/exchange\_manager.h (lines 1-145) declares the core domain types: Reserve, RateTable, Receipt, DailyReport, ExchangeOffice, and the BonusPolicy hierarchy. src/exchange\_manager.cpp (lines 7-401) defines their behavior, including conversion math, commission handling, and daily reporting.

include/utils.h together with src/utils.cpp (lines 1-112) supply shared enums, error types, ExchangeRequest, and ExchangePortion validation. They guarantee only one remainder payout exists and that allocations never exceed the provided source amount.

## Operational Roles

include/employee.h (lines 1-74) models Employee, Cashier, and Manager roles. src/employee.cpp (lines 1-120) wires those roles to ExchangeOffice, including receipt printing, reserve alerts, and manager reporting or bonus calculations.

include/client.h and src/client.cpp (lines 1-24) wrap customer identity and generate ExchangeRequest objects consumed by the cashier flow.

## Persistence Layer

include/persistence.h (lines 12-54) defines DataStore, which owns the filesystem paths under data/. src/persistence.cpp (lines 1-247) implements CSV-based loading and saving for reserves, rates, critical minima, person registry, transaction logs, and timestamped reports.

DataStore::ensurePersonId normalizes names by role so repeated logins reuse identifiers, while appendTransaction and persistReport stream plain-text history into data/transactions.log and data/reports/report-YYYYMMDD-HHMMSS.txt.

## Console Interface

src/console\_ui.cpp (lines 1-356) provides the terminal workflow. Shared helpers validate numeric input, currencies, and denomination lists. employeeSession lets cashiers perform exchanges and inspect reserves, whereas managerSession offers reporting, rate updates, critical threshold tuning, and daily cycle resets.

Each cashier exchange persists receipts with DataStore::appendTransaction and refreshes reserves through saveReserve, keeping disk state synchronized after every operation.

## Data Flow and Correlations

ConsoleUI creates Cashier and Manager objects on demand, which delegate to ExchangeOffice. ExchangeOffice logs every transaction into dailyTransactions and exposes compileDailyReport for managerial summaries saved by DataStore.

RateTable conversions rely on base currency fallbacks when direct pairs are missing, and Reserve tracks per-currency balances so critical minimum checks in ExchangeOffice feed both cashier alerts and manager dashboards.

## First-Time Run and Testing

When data/ is empty, DataStore::initialize rebuilds directories and loadReserve plus loadRates fall back to defaults supplied by main(). The seeded values match defaultReserveBalances and ensureDefaultRates, guaranteeing a smooth first launch.

tests/test\_basic.sh (lines 1-24) automates an end-to-end cashier session by piping scripted input into ./main --console, verifying that receipt output, handler attribution, and graceful shutdown all occur as expected.