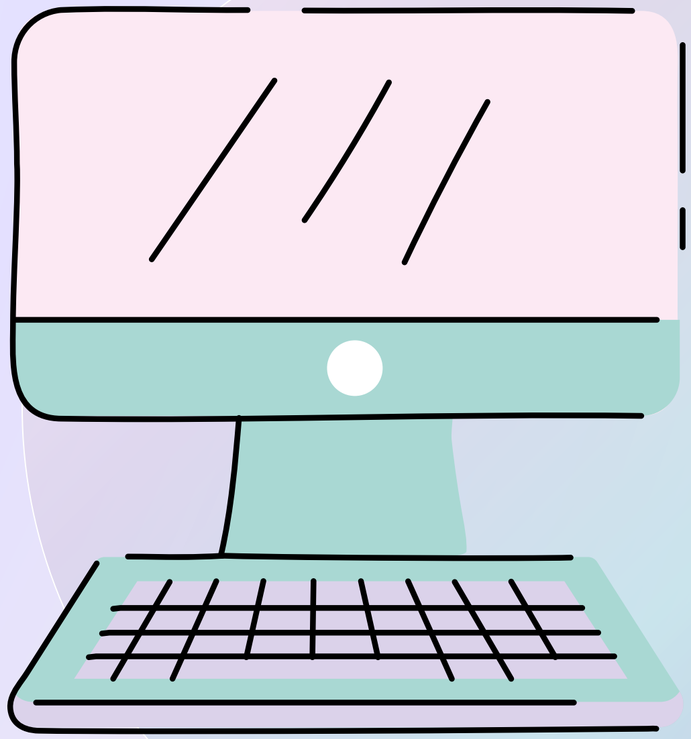
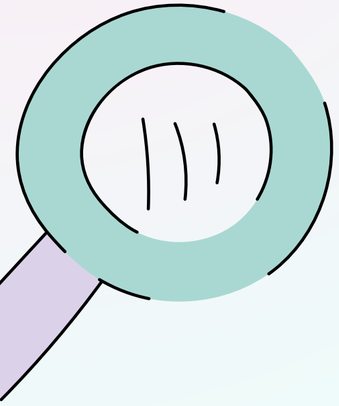
PROJECT:
PERSONAL FINANCE MANAGEMENT APPLICATION

GROUP 9



GROUP MEMBERS



1.	Nguyễn Hữu Duy Nhất	24162085
2.	Nguyễn Đức Học	24162039
3.	Nguyễn Đức Huy	24162045
4.	Bùi Tiến Khải	24162059
5.	Nguyễn Đình Lĩnh	24162067

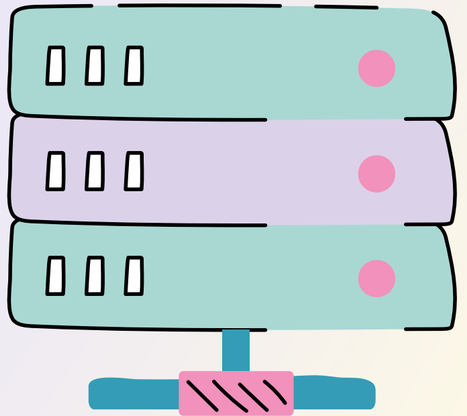
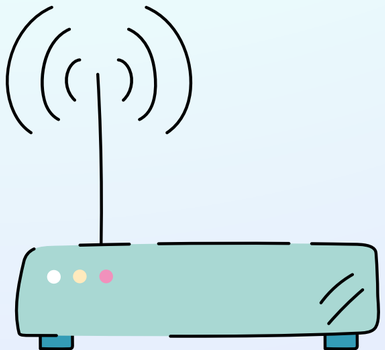
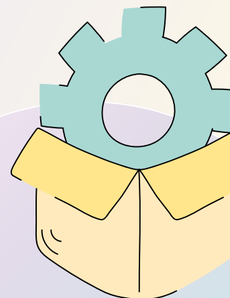


TABLE OF CONTENTS





1. INTRODUCTION

1.1 THE REASON WHY WE CHOSE THIS TOPIC

Managing personal income and expenses is a very common need, but data is easily lost or misplaced, making it difficult to compile reports.

Aimed to practice Object-Oriented Programming (OOP) in a real-world project: modeling entities (accounts, transactions, loans), handling business logic, and generating financial summaries.

The project is broad enough to apply most of the knowledge learned in OOP: inheritance, polymorphism, encapsulation, collections + streams, exception handling, and File I/O.

1.2. Product Objectives

- Build a complete personal finance management system:

• Manage accounts (bank, e-wallet, cash, etc.).

• Manage transactions (income/expense) with categorization.

• Support internal transfers between accounts.

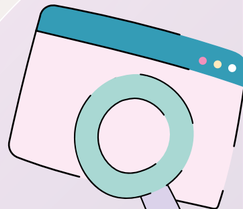
• Manage loans and lendings, track partial repayments.

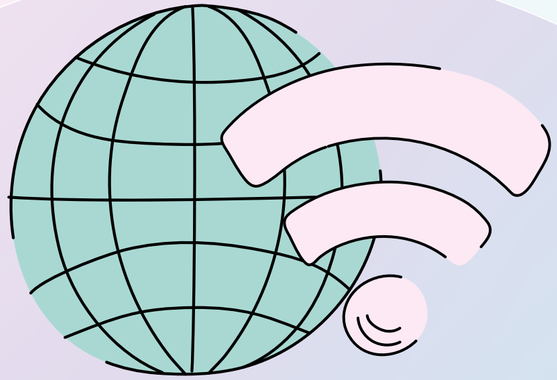
• Generate financial reports over selected time periods or accounts.

- Designed using a multi-layered architecture to allow future expansion (e.g., GUI interface, CSV export, database connection).

1.3 Core Features Implemented

- Account management: add/edit/delete, view balance, total balance.
- Transaction management: add/edit/delete, INCOME/EXPENSE type, assign Category.
- Internal transfer: record dual transactions between accounts.
- Loan/Lending management: add/edit/delete, track payments and remaining amount.
- Report generation: overview, income-expense by period, by account, by month (grouped).
- Input validation and custom business exceptions.





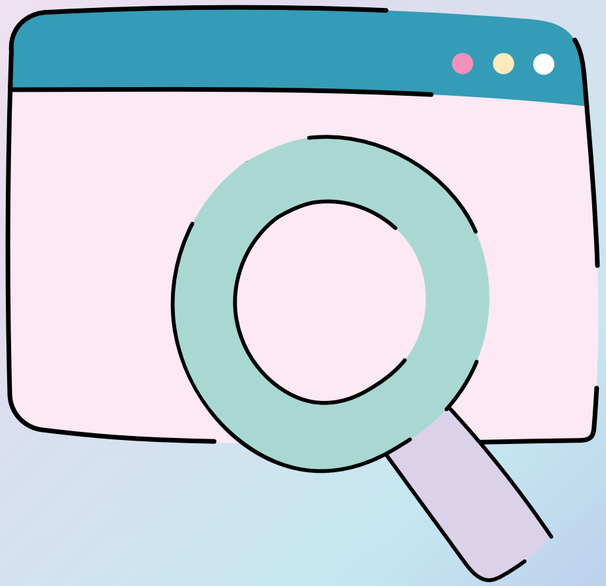
1.4 Target Users and Scope

-Target users: individual users, students, office workers.

-Demo version scope: console interface (CLI), temporary in-memory storage, CSV export currently prints to console (ready for implementation).

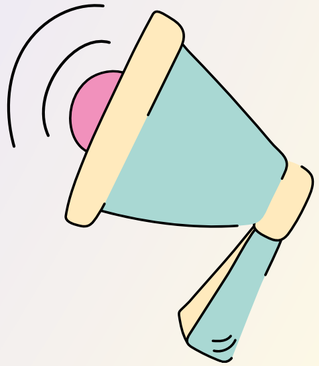
1.5 Knowledge and Skills Applied

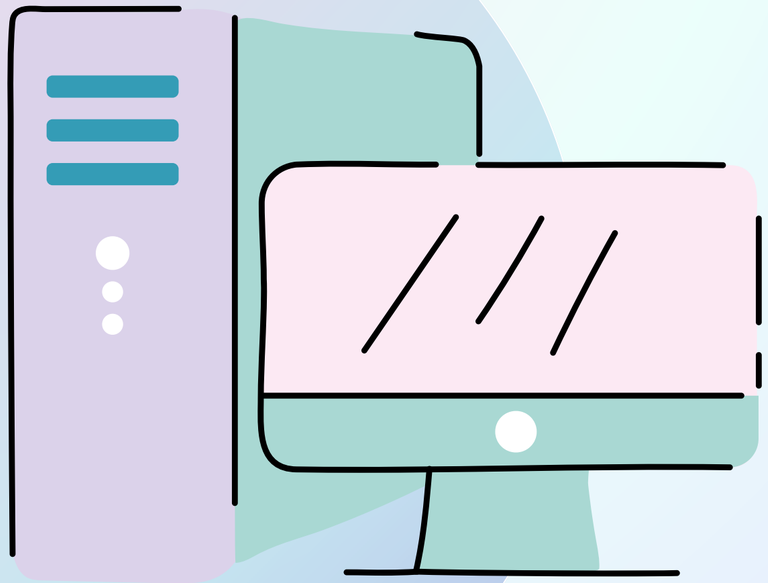
- OOP principles: class decomposition, use of interfaces/abstracts, data encapsulation.
- Java Collections & Streams: filter, group, aggregate; use of YearMonth for monthly grouping.
- Exception handling: domain-specific exceptions (e.g., insufficient balance, account not found).
- Validation: positive amounts, valid date ranges, category validation.
- Layered architecture: View (console) – Service – Model – Utility.
- Clean code practices: meaningful naming, modularization, DTO design, minimizing I/O in core logic.



1.6. Demo Evaluation Criteria

- Correct functionality (CRUD, transfers, loan tracking, etc.).
- Accurate and consistent report data, proper [from–to] date filtering.
- Readable, testable, and extensible code (ready for GUI/CSV upgrades).





2. SYSTEM DESIGN

2.1. Overall Architecture (Layered)

- UI (Console) ↔ Service (Business Logic) ↔ Model (Domain Layer) ↔ Utils/Validation.
- The Main class serves as the entry point that initializes the system and displays the feature menu.

```java
public class Main {
    private static FinanceManager financeManager = new FinanceManager();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        displayWelcomeMessage();
        int choice;
        do {
            displayMainMenu();
            choice = getIntInput("Chon chuc nang: ");
            // ... handle user choice
        } while (choice != 5);
    }
}
```

Main.java

```java
public class ReportService {
    private AccountService accountService;
    private TransactionService transactionService;
    private LoanService loanService;

    public ReportService(AccountService accountService,
                         TransactionService transactionService,
                         LoanService loanService) {
        this.accountService = accountService;
        this.transactionService = transactionService;
        this.loanService = loanService;
    }
}
```

ReportService.java

2.2. Class Diagram (Main Entities and Relationships)

- Core entities: Account, Transaction, Category, Loan, Lending, Payment, TransactionType.
- Each Transaction links to one Account (via accountId) and optionally to one Category.

```java
public class Transaction {
    private String transactionId;
    private String accountId;
    private TransactionType type;
    private double amount;
    private LocalDateTime date;
    private String description;
    private Category category; // 0..1

    public String getFormattedDate() {
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        return date.format(fmt);
    }
}
```

Transaction.java

```java
public enum TransactionType {
    INCOME("Thu nhập"),
    EXPENSE("Chi tiêu"),
    TRANSFER("Chuyển khoản");

    private final String vietnameseName;
    public String getVietnameseName() { return vietnameseName; }
}
```

Transaction.java

2.3. Typical Use Case Flows

- Add Transaction → create a Transaction → send to TransactionService → update and display report.
- Internal Transfer → verify accounts and sufficient balance → record paired transactions.
- Loan/Lending & Repayment → create Loan/Lending → add Payment → compute remaining and status.

2.3. Typical Use Case Flows

```java
// Internal account transfer
public boolean transferBetweenAccounts(String fromId, String toId, double amount)
        throws AccountNotFoundException, InsufficientBalanceException {

    Optional<Account> fromOpt = findAccountById(fromId);
    Optional<Account> toOpt   = findAccountById(toId);

    if (fromOpt.isEmpty()) throw new AccountNotFoundException("Khong tim thay tai khoan nguon: " + fromId);
    if (toOpt.isEmpty())   throw new AccountNotFoundException("Khong tim thay tai khoan dich: " + toId);

    // ... check balance, debit source, credit destination
}
```

AccountService.java

```java
public class LoanService {
    private List<Loan> loans = new ArrayList<>();
    private List<Lending> lendings = new ArrayList<>();

    // ADD: create a loan with term months (template ready)
    public void addLoan(String lender, double amount, double interest, /* ... */) {
        // ... initialize Loan and push to loans list
    }
}
```

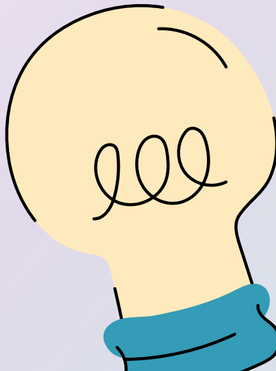
LoanService.java

2.4. Data Rules and Constraints

- Amount values are always positive; the transaction type determines whether to add or subtract from totals.
- Time filtering follows the inclusive range [from..to]; date and time formats are consistent.
- Category may be null → must always be checked before grouping or displaying data.
- Transfers: from != to, accounts must exist, and source balance must be sufficient.

```java
private static final DateTimeFormatter DATE_FORMATTER =
        DateTimeFormatter.ofPattern("dd/MM/yyyy");
private static final DateTimeFormatter DATE_TIME_FORMATTER =
        DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

public static boolean isWithinRange(LocalDate d, LocalDate start, LocalDate end) {
    return d != null && start != null && end != null
            && !d.isBefore(start) && !d.isAfter(end); // inclusive
}
```

DateUtils.java

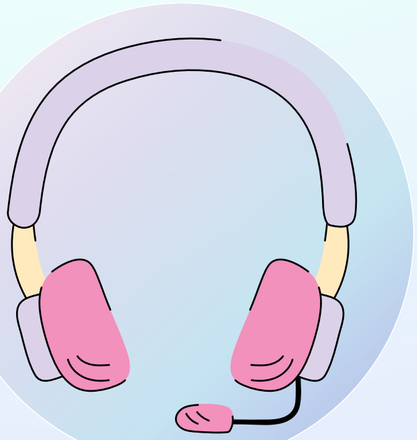


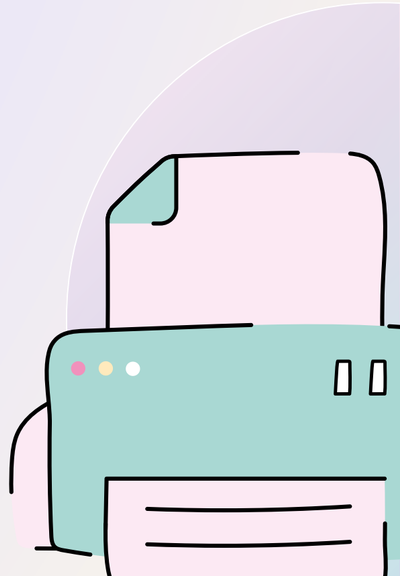
2.5. Testing and Quality Assurance

- Top-N largest transactions: sort by descending amount, limit(5) to verify boundary values.
- Avoid division by zero when total income = 0 → display “N/A”.
- Prevent NPE when Category == null.

```java
System.out.println("\nTOP 5 GIAO DICH LON NHAT:");
accountTransactions.stream()
    .sorted((t1, t2) -> Double.compare(t2.getAmount(), t1.getAmount()))
    .limit(5)
    .forEach(tx -> {
        String symbol = tx.getType() == TransactionType.INCOME ? "↑" : "↓";
        System.out.printf("  %s %-12s: %,.2f VND - %s\n",
                symbol, tx.getCategory().getName(), tx.getAmount(), tx.getDescription());
    });
```

ReportService.java



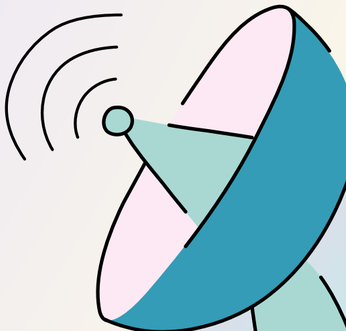


2.6. Future Enhancements (Short Roadmap)

- **CSV/Excel Export**: replace console output with proper file writing (RFC 4180 escaping).
- **GUI (Swing/JavaFX)**: charts for income/expense, date filters, and category selection.
- **Persistent storage**: replace in-memory with JSON or SQLite database.
- Loan reminders and simple/compound interest support.

```
System.out.println("\nXUAT DU LIEU (CSV Format):");
System.out.println("TAI_KHOAN_ID,TEN,LOAI,SO_DU");
accountService.getAllAccounts().forEach(acc → {
    System.out.printf("%s,%s,%s,%.2f\n",
            acc.getAccountId(), acc.getAccountName(), acc.getAccountType(), acc.getBalance());
});
// → Should be replaced with Files.newBufferedWriter(...) to write to an actual file.
```

ReportService.java

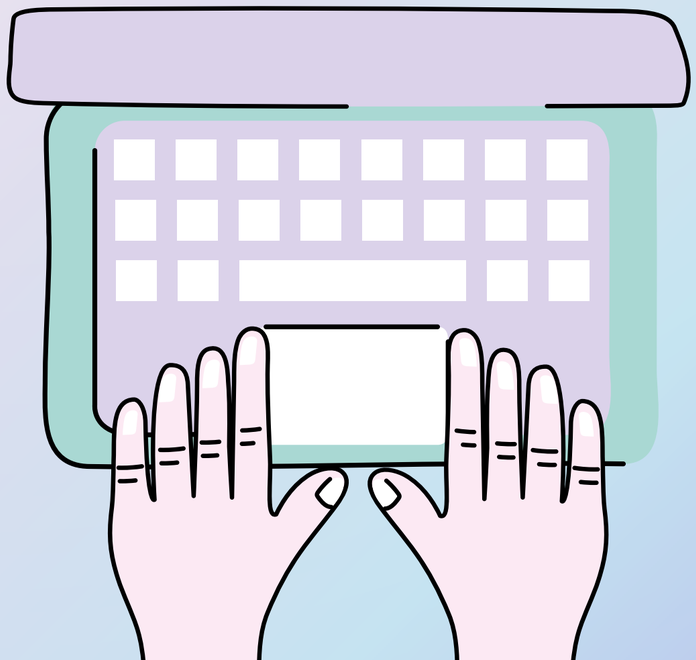


3. PRODUCT

3.1. OverView

Finance Manager is a comprehensive personal finance management application developed in Java, designed to help users track and organize their financial activities efficiently. The application follows a clear multi-layered architecture, features a user-friendly console interface, and provides full Vietnamese language support.

Key features



- Multi-account management (bank accounts, e-wallets, cash, etc.)
- Income and expense tracking with categorized transactions
- Loan and lending management with interest calculation
- Rich reporting and analytics system
- Financial validation and warning mechanisms

3.2. Detailed Feature Description

Account Management

Transaction Management

Loan & Lending Management

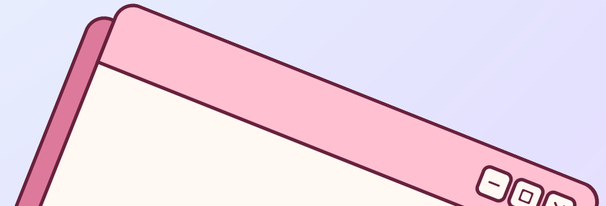
Reporting System

Account Management

Description: Allows users to create and manage multiple financial accounts.

Detail:

- Create new accounts
- View accounts
- Delete accounts
- Internal transfer
- Search accounts
- Top up



Create new accounts

Function: Create a new account with the following information: name, type, account number, initial balance.

Validation:

- Use Validator.validateAccount to check the validity of the name, account type, account number, and balance.
- Account name must be 2-50 characters long, containing only letters, numbers, and spaces.
- Account type must be one of the following values: BANK, WALLET, CASH, SAVINGS, CREDIT.
- Account number must contain only numbers, 5-30 characters long.
- Balance must not be negative

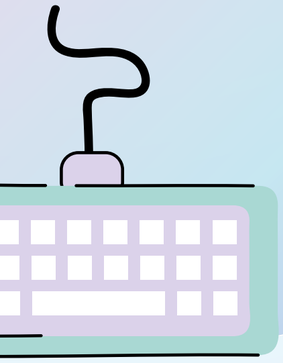
```java
//Account.java
public Account(String accountName, String accountType, String accountNumber, double initialBalance) {
    this.accountId = "ACC_" + UUID.randomUUID().toString().substring(0, 8);
    this.accountName = accountName;
    this.accountType = accountType;
    this.accountNumber = accountNumber;
    this.balance = initialBalance;
    this.currency = "VND";
}

//AccountService.java
public void addAccount(Account account) {
    if (account.isValid()) {
        accounts.add(account);
        System.out.println("Them tai khoan thanh cong: " + account.getAccountName());
    } else {
        System.out.println("Tai khoan khong hop le!");
    }
}
```

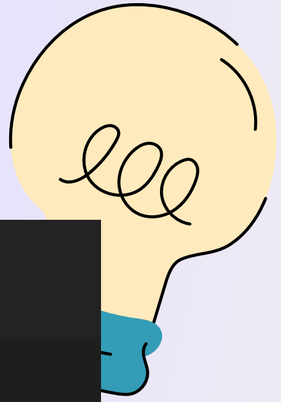
Processing:

- Automatically generate ID using UUID.
- Add the account to the list if valid

Create new accounts

Result





```
--- THEM TAI KHOAN MOI ---
Nhap ten tai khoan: Nhat
Nhap loai tai khoan (BANK/WALLET/CASH/SAVINGS/CREDIT): bank
Nhap so tai khoan: 123456
Nhap so du ban dau: 30000
Them tai khoan thanh cong: Nhat
```

View accounts

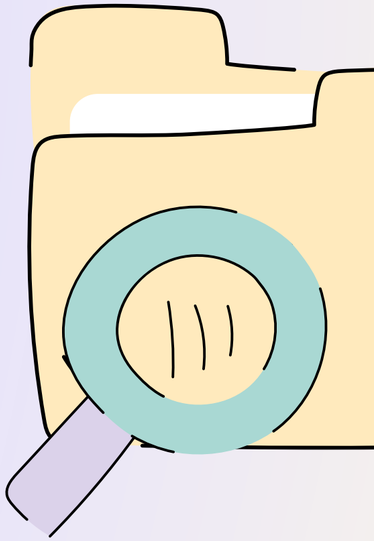
Function: Display all accounts with detailed information and total balance.

- **Display in a beautiful table format:** Show ID, name, type, account number, balance, using Unicode characters to create the frame.
- **Dynamic width calculation:** Automatically adjust table width based on content.
- **Balance aggregation:** Display the total balance of all accounts

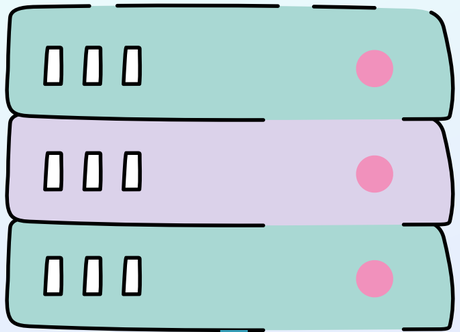
```java
// Account.java
public void displayInfo() {
    int maxWidth = calculateMaxWidth();
    String horizontalLine = "┌" + "─".repeat(maxWidth + 2) + "┐";
    String middleLine = "├" + "─".repeat(maxWidth + 2) + "┤";
    String bottomLine = "└" + "─".repeat(maxWidth + 2) + "┘";

    System.out.println(horizontalLine);
    System.out.printf("│ %-" + (maxWidth) + "s │\n", "THÔNG TIN TÀI KHOẢN");
    System.out.println(middleLine);
    System.out.printf("│ %-" + (maxWidth) + "s │\n", "ID: " + accountId);
    System.out.printf("│ %-" + (maxWidth) + "s │\n", "Tên: " + accountName);
    System.out.printf("│ %-" + (maxWidth) + "s │\n", "Loại: " + accountType);
    System.out.printf("│ %-" + (maxWidth) + "s │\n", "Số TK: " + accountNumber);
    System.out.printf("│ %-" + (maxWidth) + "s │\n",
            String.format("Số dư: %,.2f %s", balance, currency));
    System.out.println(bottomLine);
}
```

View accounts

Result



```
DANH SACH TAI KHOAN (1 tai khoan)
==========================================

  THÔNG TIN TÀI KHOẢN
  ID: ACC_1f4bbfa4
  Tên: Nhat
  Loại: BANK
  Số TK: 123456
  Số dư: 30.000,00 VND

TONG SO DU TAT CA TAI KHOAN: 30000,00 VND
```



Delete accounts

Function: Delete account based on ID

- **Existence verification:** Check if the account exists before deleting
- **Clear notification:** Notify success or reason for failure
- **Data safety:** No side effects when deleting a non-existent account

```java
public boolean deleteAccount(String accountId) {
    Optional<Account> accountOpt = findAccountById(accountId);
    if (accountOpt.isPresent()) {
        accounts.remove(accountOpt.get());
        System.out.println("Xoa tai khoan thanh cong!");
        return true;
    } else {
        System.out.println("Khong tim thay tai khoan voi ID: " + accountId);
        return false;
    }
}
```

Result

```
Nhap ID tai khoan can xoa: ACC_62057627
Xoa tai khoan thanh cong!
```

Internal transfer

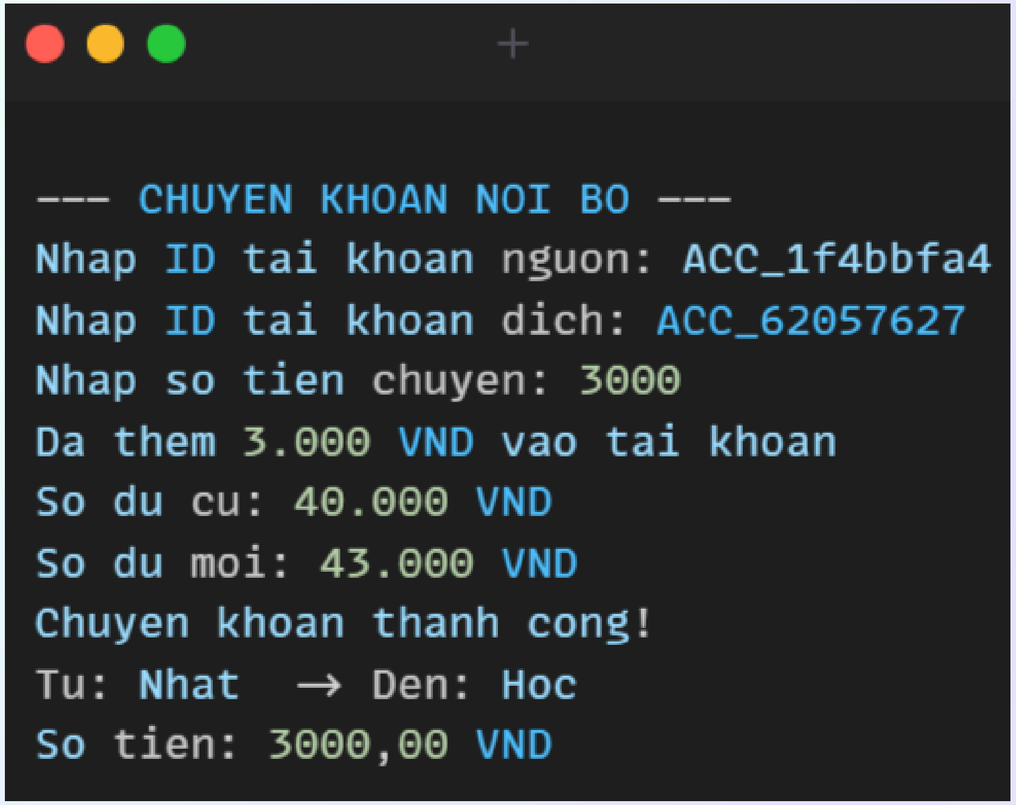
Function: Transfer money from a source account to a destination account.

- **Exception handling:** Use custom exceptions (AccountNotFoundException, InsufficientBalanceException)
- **Atomic operation:** Ensure either both accounts are updated, or nothing changes
- **Balance check:** Ensure the source account has sufficient funds
- **Detailed logging:** Record full transfer information

```java
public boolean transferBetweenAccounts(String fromAccountId, String toAccountId, double amount)
        throws AccountNotFoundException, InsufficientBalanceException {

    Optional<Account> fromAccountOpt = findAccountById(fromAccountId);
    Optional<Account> toAccountOpt = findAccountById(toAccountId);

    if (fromAccountOpt.isEmpty()) {
        throw new AccountNotFoundException("Khong tim thay tai khoan nguon: " + fromAccountId);
    }

    if (toAccountOpt.isEmpty()) {
        throw new AccountNotFoundException("Khong tim thay tai khoan dich: " + toAccountId);
    }

    Account fromAccount = fromAccountOpt.get();
    Account toAccount = toAccountOpt.get();

    if (fromAccount.getBalance() < amount) {
        throw new InsufficientBalanceException(
                "So du khong du! So du hien tai: " + fromAccount.getBalance());
    }

    // Perform transfer
    fromAccount.withdraw(amount);
    toAccount.deposit(amount);

    System.out.println("Chuyen khoan thanh cong!");
    System.out.printf("Tu: %s → Den: %s\n", fromAccount.getAccountName(), toAccount.getAccountName());
    System.out.printf("So tien: %.2f VND\n", amount);

    return true;
}
```

Internal transfer

Result



Search accounts

Function: Search for accounts whose names contain the input string (case-insensitive).

- **Case-insensitive search:** toLowerCase() for both query and data
- **Partial matching:** Search by substring
- **Display results:** Use the same format as the account list

```java
public void findAccountByName(String name) {
    accountService.getAllAccounts().stream()
            .filter(acc → acc.getAccountName().toLowerCase().contains(name.toLowerCase()))
            .forEach(Account::displayInfo);
}
```

Result

```
Nhap ten tai khoan can tim: Nhat
┌──────────────────────────────┐
│ THÔNG TIN TÀI KHOẢN          │
├──────────────────────────────┤
│ ID: ACC_1f4bbfa4             │
│ Tên: Nhat                    │
│ Loại: BANK                   │
│ Số TK: 123456                │
│ Số dư: 27.000,00 VND         │
└──────────────────────────────┘
```

Top up

Function: add funds to the account using an ID

- **Amount validation:**
  - Check for negative numbers
  - Check format (no leading zeros)
  - Maximum limit of 1 billion
- **Real-time update:** Balance is updated immediately
- **Currency format:** Display amount with separator (1,000,000)

```java
public void deposit(double amount) {
    if (amount > 0) {
        double oldBalance = this.balance;
        this.balance += amount;
        System.out.println("Da them " + String.format("%,.0f", amount) + " " + currency + " vao tai khoan");
        System.out.println("So du cu: " + String.format("%,.0f", oldBalance) + " " + currency);
        System.out.println("So du moi: " + String.format("%,.0f", this.balance) + " " + currency);
    } else {
        System.out.println("So tien them phai lon hon 0!");
    }
}
```

Result

```
Nhap ID tai khoan can them tien: ACC_1f4bbfa4
Nhap so tien can them: 300000
Da them 300.000 VND vao tai khoan
So du cu: 27.000 VND
So du moi: 327.000 VND
Them tien thanh cong!
Tai khoan: Nhat
So tien them: 300.000,00 VND
So du moi: 327.000,00 VND
```

Transaction Management

Description: Records and monitors all income and expense transactions.

Detail:

- Add transactions
- View transaction history
- View current balance
- View transactions by category

Add transactions

Function: Supports two types — INCOME and EXPENSE — with categorized entries

- **Flexible category system:**
  - Default categories: Salary, Bonus, Investment, Food, Transportation, etc.
  - Automatically create new categories if they don't exist
- **Real-time balance check:** Prevent transactions exceeding the balance
- **Automatic update:** Account balance is updated immediately after adding a transaction
- **Transaction integrity:** Ensure transaction integrity

```java
public void addTransaction(String accountId, TransactionType type, double amount,
                           String description, String categoryName) {

    Optional<Account> accountOpt = accountService.findAccountById(accountId);
    if (accountOpt.isEmpty()) {
        System.out.println("Khong tim thay tai khoan!");
        return;
    }

    Optional<Category> categoryOpt = categories.stream()
            .filter(cat → cat.getName().equalsIgnoreCase(categoryName))
            .findFirst();

    Category category;
    if (categoryOpt.isPresent()) {
        category = categoryOpt.get();
    } else {
        // Create new category
        category = new Category(categoryName, "Mo ta", type);
        categories.add(category);
    }

    Transaction transaction = new Transaction(accountId, type, amount, description, category);

    if (transaction.isValid()) {
        transactions.add(transaction);
        account.updateBalance(amount, type);
        System.out.println("Them giao dich thanh cong!");
        transaction.displayInfo();
    }
}
```

Add transactions

Result

income

```
--- THEM GIAO DICH MOI ---
Nhap ID tai khoan: ACC_1f4bbfa4
Nhap loai giao dich (income/expense): income
Nhap so tien: 3000
Nhap mo ta: tien luong
Nhap danh muc: luong
Them giao dich thanh cong!

ID: TRX_9746bfdc
Tai khoan: ACC_1f4bbfa4
Loai: Thu nhập
So tien: 3000,00
Danh muc: Luong
Mo ta: tien luong
Thoi gian: 22/10/2025 00:29:04
```

expense

```
--- THEM GIAO DICH MOI ---
Nhap ID tai khoan: ACC_1f4bbfa4
Nhap loai giao dich (income/expense): expense
Nhap so tien: 400
Nhap mo ta: tien an vat
Nhap danh muc: an vat
Them giao dich thanh cong!

ID: TRX_55f61cb6
Tai khoan: ACC_1f4bbfa4
Loai: Chi tiêu
So tien: 400,00
Danh muc: an vat
Mo ta: tien an vat
Thoi gian: 22/10/2025 00:32:12
```

View transaction history

Function: review all transactions that have been made

- **Structured display:** Each transaction in a separate frame
- **Full information:** ID, account, type, amount, category, description, time
- **Count:** Display total number of transactions

```java
public void displayAllTransactions() {
    if (transactions.isEmpty()) {
        System.out.println("Khong co giao dich nao!");
        return;
    }

    System.out.println("\nLICH SU GIAO DICH (" + transactions.size() + " giao dich)");
    System.out.println("========================================");

    for (Transaction transaction : transactions) {
        transaction.displayInfo();
        System.out.println();
    }
}
```

View transaction history

Result

View current balance

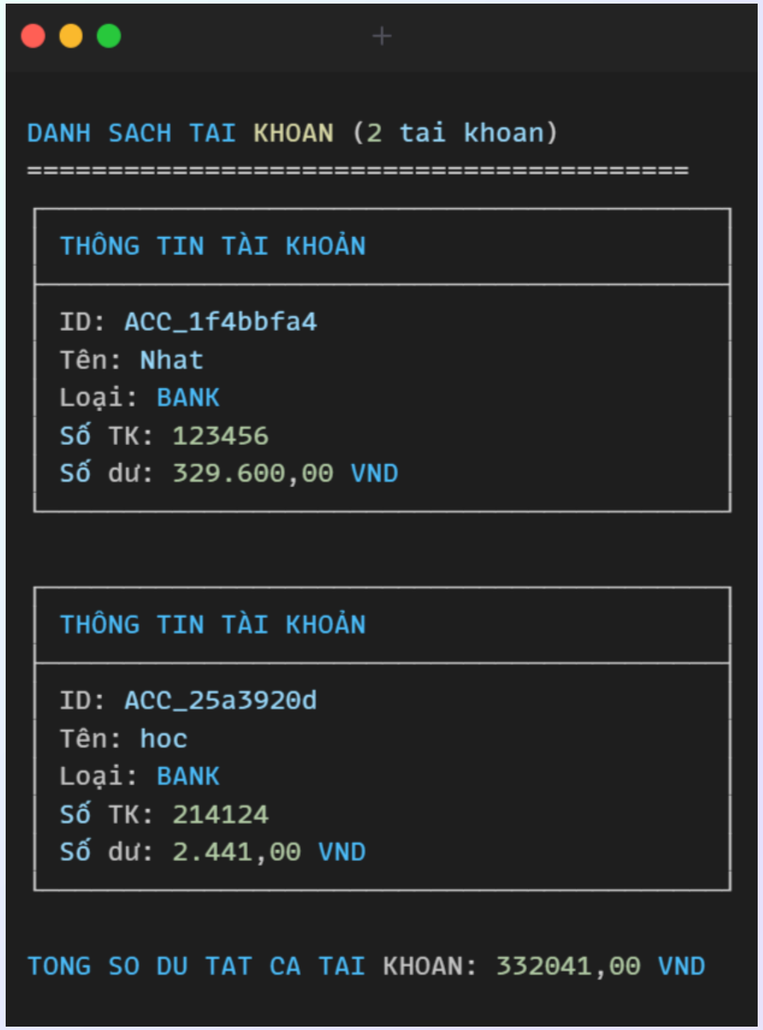
Function: Review account balances after making transactions

- **Multi-account aggregation:** Display individual account balances and total balance
- **Real-time data:** Balances are always updated to the latest
- **Consistency:** Ensure balances match transaction history

```java
public void displayAllAccounts() {
    if (accounts.isEmpty()) {
        System.out.println("Khong co tai khoan nao!");
        return;
    }

    System.out.println("\nDANH SACH TAI KHOAN (" + accounts.size() + " tai khoan)");
    System.out.println("=========================================");
    for (Account account : accounts) {
        account.displayInfo();
        System.out.println();
    }

    // Display total balance
    double totalBalance = getTotalBalance();
    System.out.printf("TONG SO DU TAT CA TAI KHOAN: %.2f VND\n", totalBalance);
}
```

View current balance

Result



View transactions by category

Function: Review transactions made by category

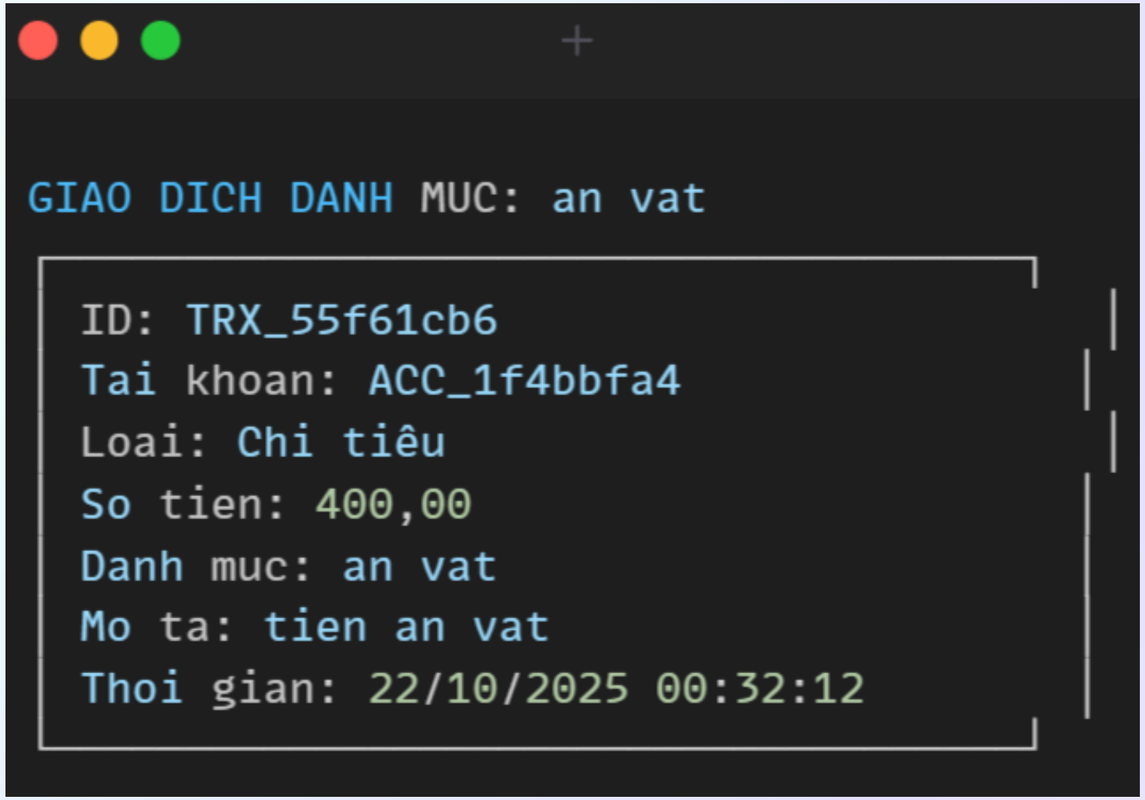
- **Powerful filtering:** Use Stream API to filter by category
- **Case-insensitive:** equalsIgnoreCase()
- **Handle no results:** Clearly notify when there are no transactions

```java
public List<Transaction> getTransactionsByCategory(String categoryName) {
    return transactions.stream()
            .filter(transaction → transaction.getCategory().getName().equalsIgnoreCase(categoryName))
            .collect(Collectors.toList());
}

public void displayTransactionsByAccount(String accountId) {
    List<Transaction> accountTransactions = transactions.stream()
            .filter(transaction → transaction.getAccountId().equals(accountId))
            .collect(Collectors.toList());

    if (accountTransactions.isEmpty()) {
        System.out.println("Khong co giao dich nao cho tai khoan nay!");
        return;
    }

    System.out.println("\nGIAO DICH TAI KHOAN " + accountId);
    System.out.println("=======================================");

    for (Transaction transaction : accountTransactions) {
        transaction.displayInfo();
        System.out.println();
    }
}
```

```java
// FinanceManager.java - Phương thức gọi từ menu chính
public void displayTransactionsByCategory(String category) {
    List<Transaction> transactions = transactionService.getTransactionsByCategory(category);
    if (transactions.isEmpty()) {
        System.out.println(" Khong co giao dich nao trong danh muc: " + category);
        return;
    }
    System.out.println("\n GIAO DICH DANH MUC: " + category);
    transactions.forEach(Transaction::displayInfo);
}
```

View transactions by category

Result

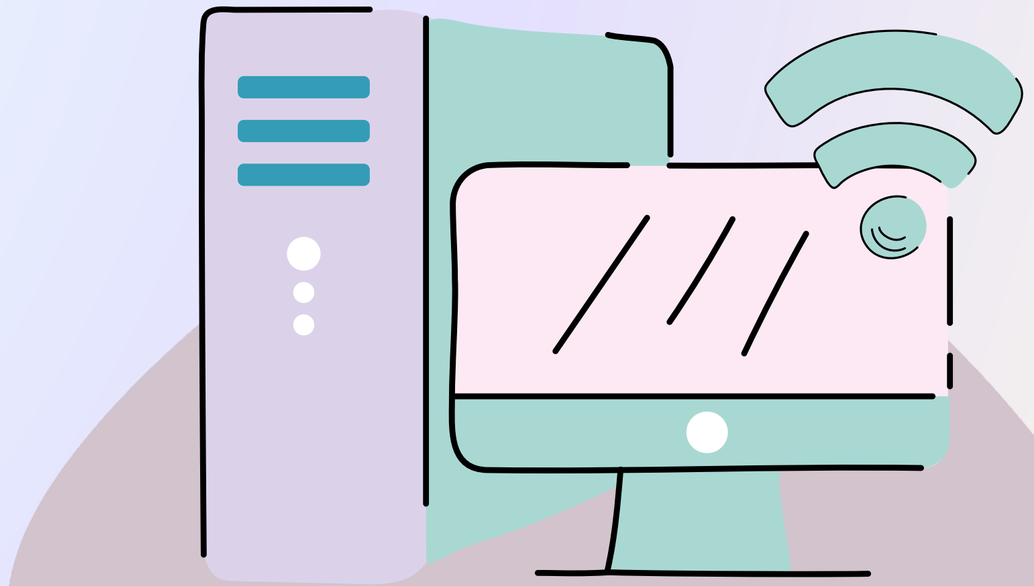


Loan and Lending Management

Description: Manages both borrowing (loans taken) and lending (loans given)

Detail:

- Create a new loan
- Create a new lending
- View loan list
- View lending list
- Repay loan
- Collect loan payment



Create a new loan

Function: Create a new loan with the following information: lender, amount, interest rate, number of months, description

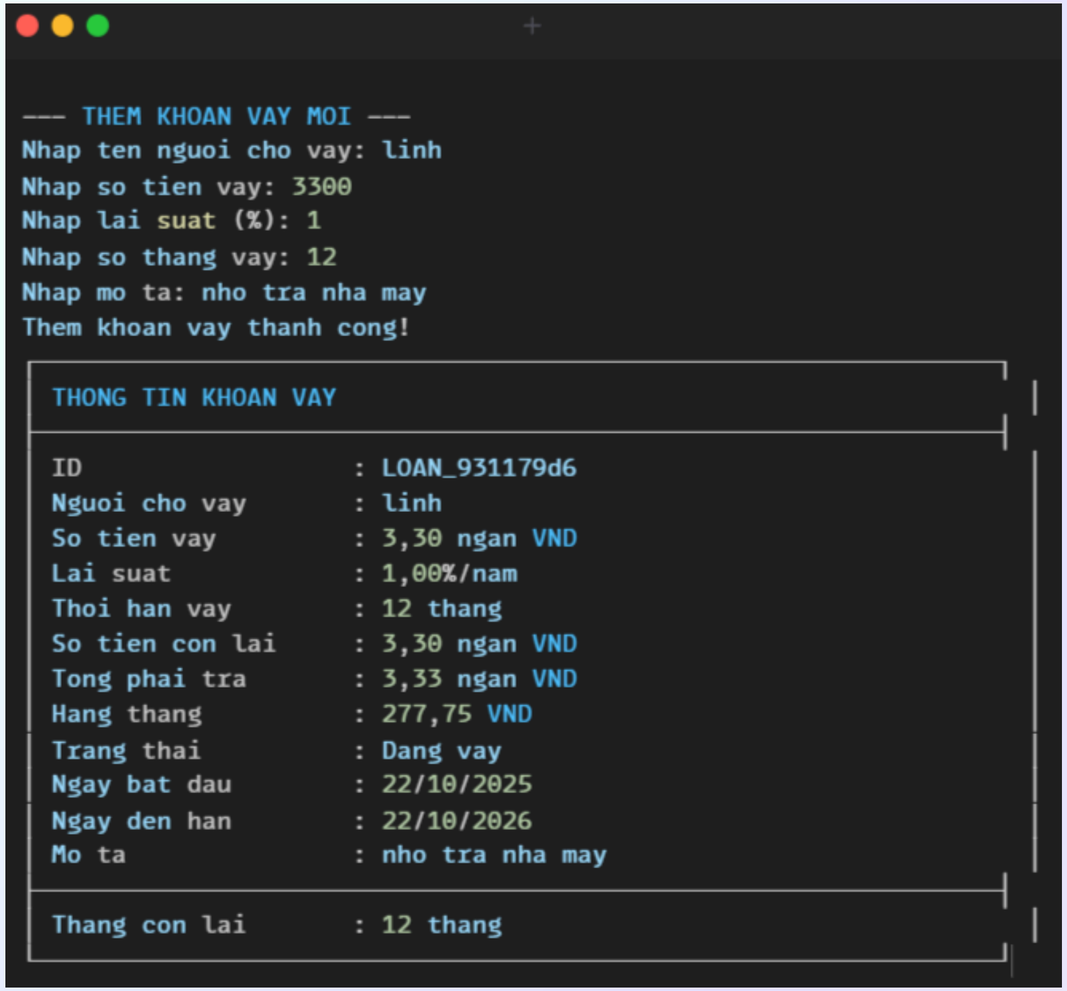
- **Validation:**
  - Amount, interest rate, number of months must be greater than 0.
  - Lender's name cannot be empty.
- **Processing:**
  - Automatically generate ID.
  - Calculate maturity date.
  - Initialize "ACTIVE" status.

```java
//LoanService.java
public void addLoan(String lender, double amount, double interest,
                    int months, String description) {
    try {
        Loan loan = new Loan(lender, amount, interest, months, description);
        if (loan.isValid()) {
            loans.add(loan);
            System.out.println("Them khoan vay thanh cong!");
            loan.displayInfo();
        }
    } catch (IllegalArgumentException e) {
        System.out.println("Loi: " + e.getMessage());
    }
}
```

```java
//Loan.java
public Loan(String lenderName, double principalAmount, double interestRate,
            int loanMonths, String description) {
    if (principalAmount <= 0) {
        throw new IllegalArgumentException("So tien vay phai lon hon 0");
    }
    if (interestRate < 0) {
        throw new IllegalArgumentException("Lai suat khong duoc am");
    }
    if (loanMonths <= 0) {
        throw new IllegalArgumentException("So thang vay phai lon hon 0");
    }
    this.loanId = "LOAN_" + java.util.UUID.randomUUID().toString().substring(0, 8);
    this.lenderName = lenderName;
    this.principalAmount = principalAmount;
    this.interestRate = interestRate;
    this.loanMonths = loanMonths;
    this.remainingAmount = principalAmount;
    this.description = description;
    this.status = "ACTIVE";
    this.startDate = LocalDateTime.now();
    this.dueDate = startDate.plusMonths(loanMonths);
    this.paymentHistory = new ArrayList<>();
}
```

Create a new loan

Result



Create a new lending

Function: Similar to a loan, but for lending to other

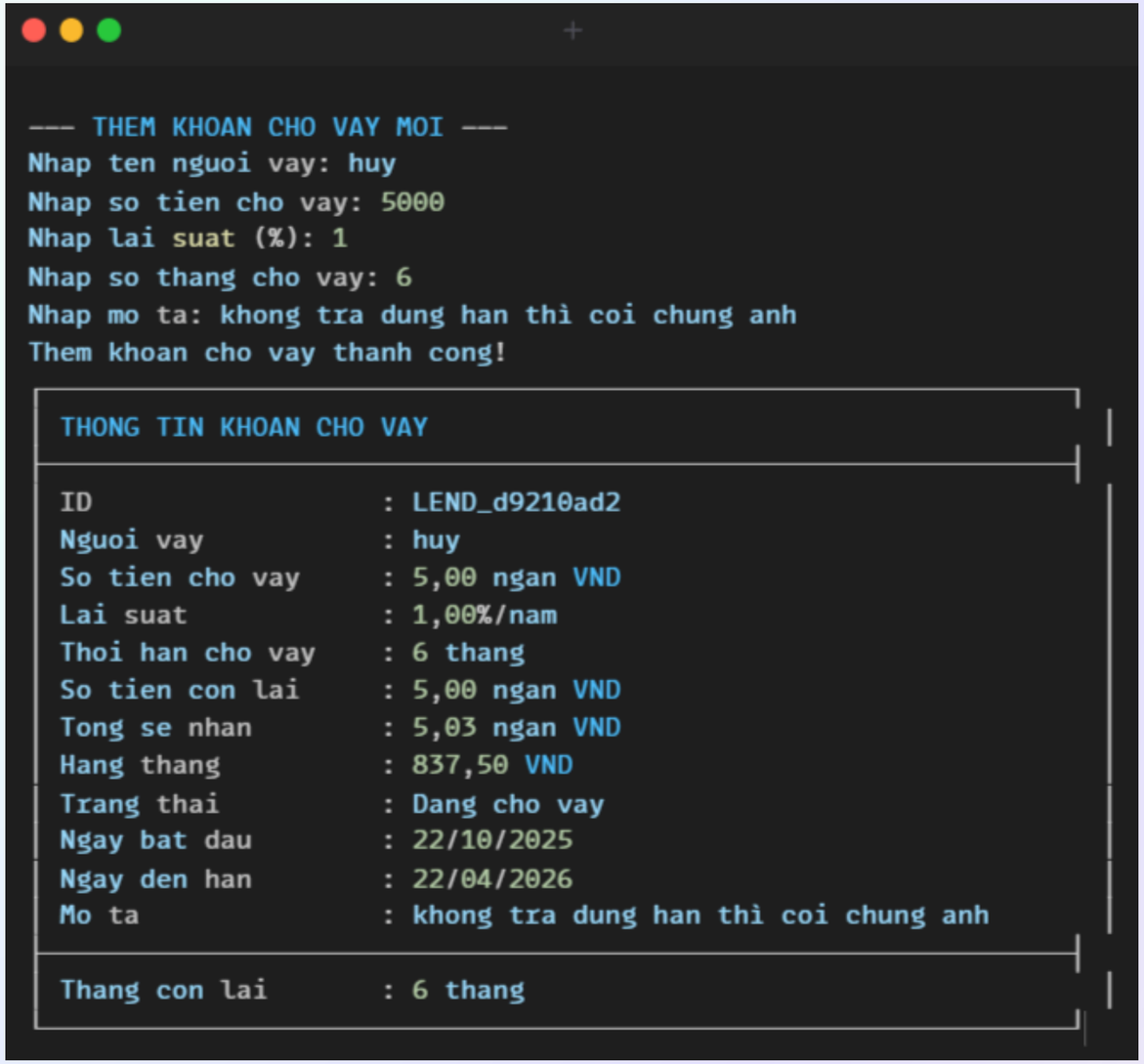
- **Automatic calculation:**
  - Total interest: principalAmount * (interestRate / 100) * (months / 12.0)
  - Total amount: Principal + Interest
  - Monthly payment: Total / months
- **Comprehensive validation:**
  - Amount > 0
  - Interest rate ≥ 0
  - Number of months > 0
  - Name cannot be empty
- **Automatic time setting:** Start date (current), due date (startDate + months)

```java
//LoanService.java
public void addLending(String borrower, double amount, double interest,
                       int months, String description) {
    try {
        Lending lending = new Lending(borrower, amount, interest, months, description);
        if (lending.isValid()) {
            lendings.add(lending);
            System.out.println("Them khoan cho vay thanh cong!");
            lending.displayInfo();
        }
    } catch (IllegalArgumentException e) {
        System.out.println("Loi: " + e.getMessage());
    }
}
```

```java
//Lending.java - Constructor
public Lending(String borrowerName, double principalAmount, double interestRate,
               int lendingMonths, String description) {
    if (principalAmount <= 0) throw new IllegalArgumentException("So tien cho vay phai lon hon 0");
    if (interestRate < 0) throw new IllegalArgumentException("Lai suat khong duoc am");
    if (lendingMonths <= 0) throw new IllegalArgumentException("So thang cho vay phai lon hon 0");
    if (borrowerName == null || borrowerName.trim().isEmpty()) {
        throw new IllegalArgumentException("Ten nguoi vay khong duoc de trong");
    }
    this.lendingId = "LEND_" + java.util.UUID.randomUUID().toString().substring(0, 8);
    this.borrowerName = borrowerName;
    this.principalAmount = principalAmount;
    this.interestRate = interestRate;
    this.lendingMonths = lendingMonths;
    this.remainingAmount = principalAmount;
    this.description = description;
    this.status = "ACTIVE";
    this.startDate = LocalDateTime.now();
    this.dueDate = startDate.plusMonths(lendingMonths);
    this.repaymentHistory = new ArrayList<>();
}
```

Create a new lending

Result



```
--- THEM KHOAN CHO VAY MOI ---
Nhap ten nguoi vay: huy
Nhap so tien cho vay: 5000
Nhap lai suat (%): 1
Nhap so thang cho vay: 6
Nhap mo ta: khong tra dung han thì coi chung anh
Them khoan cho vay thanh cong!

THONG TIN KHOAN CHO VAY

ID                  : LEND_d9210ad2
Nguoi vay           : huy
So tien cho vay     : 5,00 ngan VND
Lai suat            : 1,00%/nam
Thoi han cho vay    : 6 thang
So tien con lai     : 5,00 ngan VND
Tong se nhan        : 5,03 ngan VND
Hang thang          : 837,50 VND
Trang thai          : Dang cho vay
Ngay bat dau        : 22/10/2025
Ngay den han        : 22/04/2026
Mo ta               : khong tra dung han thì coi chung anh

Thang con lai       : 6 thang
```

View loan list

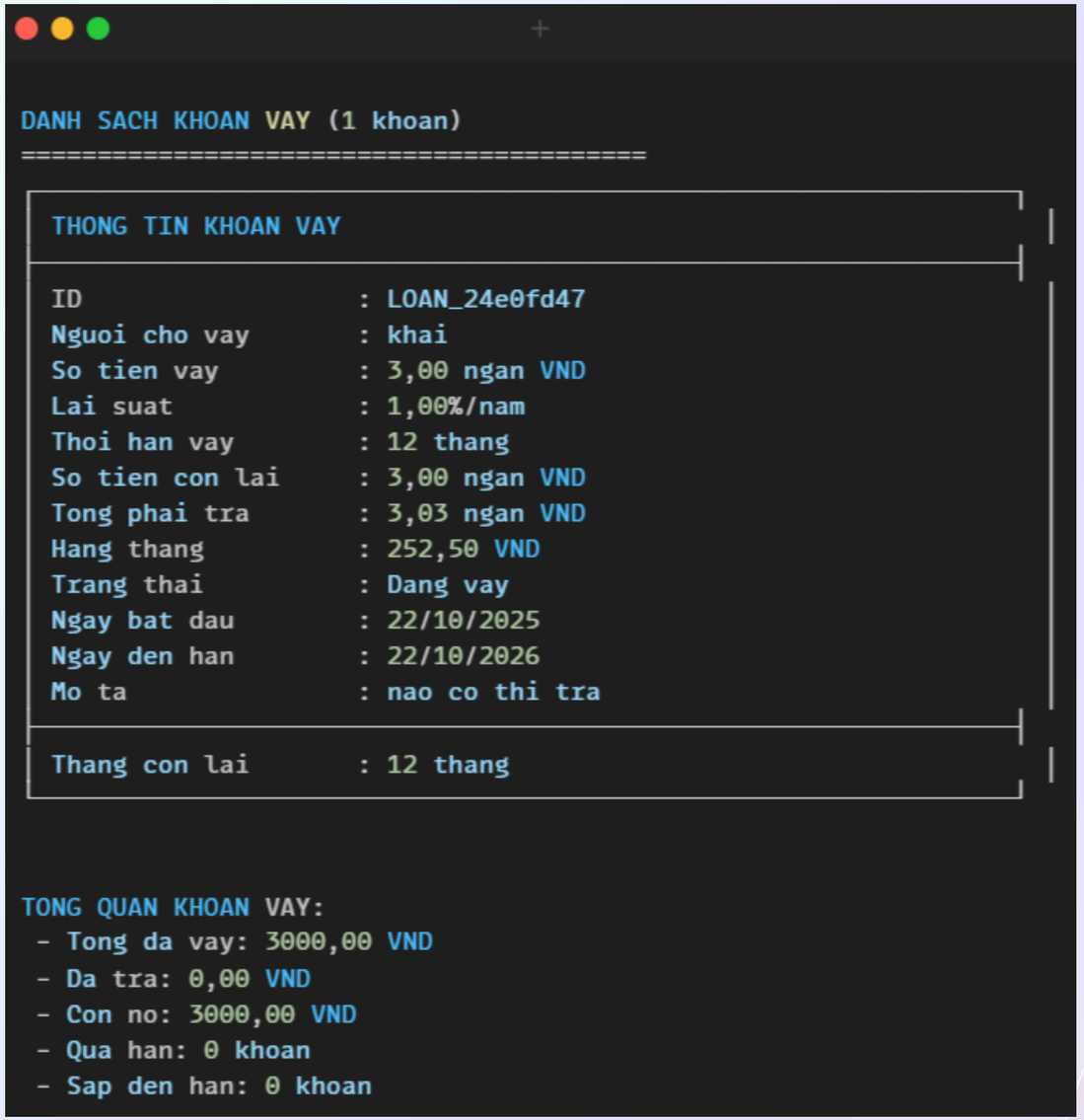
Function: Display all loans with detailed information and overview.

- **Detailed overview:** Display aggregated statistics as well
- **Visual warnings:** Colors and icons for statuses
- **Smart currency formatting:** Automatically convert to millions, billions when needed

```java
public void displayAllLoans() {
    if (loans.isEmpty()) {
        System.out.println("Khong co khoan vay nao!");
        return;
    }

    System.out.println("\nDANH SACH KHOAN VAY (" + loans.size() + " khoan)");
    System.out.println("========================================");

    for (Loan loan : loans) {
        loan.displayInfo();
        System.out.println();
    }

    // THÊM: Hiển thị tổng quan
    displayLoanSummary();
}

private void displayLoanSummary() {
    double totalBorrowed = loans.stream().mapToDouble(Loan::getPrincipalAmount).sum();
    double totalRemaining = loans.stream().mapToDouble(Loan::getRemainingAmount).sum();
    double totalPaid = totalBorrowed - totalRemaining;
    long overdueLoans = loans.stream().filter(Loan::isOverdue).count();
    long dueSoonLoans = loans.stream().filter(Loan::isDueSoon).count();

    System.out.println("\nTONG QUAN KHOAN VAY:");
    System.out.printf(" - Tong da vay: %.2f VND\n", totalBorrowed);
    System.out.printf(" - Da tra: %.2f VND\n", totalPaid);
    System.out.printf(" - Con no: %.2f VND\n", totalRemaining);
    System.out.printf(" - Qua han: %d khoan\n", overdueLoans);
    System.out.printf(" - Sap den han: %d khoan\n", dueSoonLoans);
}
```

View loan list

Result



```
DANH SACH KHOAN VAY (1 khoan)
========================================

┌──────────────────────────────────────────────────────────┐
│ THONG TIN KHOAN VAY                                      │
├──────────────────────────────────────────────────────────┤
│ ID                  : LOAN_24e0fd47                      │
│ Nguoi cho vay       : khai                               │
│ So tien vay         : 3,00 ngan VND                      │
│ Lai suat            : 1,00%/nam                          │
│ Thoi han vay        : 12 thang                           │
│ So tien con lai     : 3,00 ngan VND                      │
│ Tong phai tra       : 3,03 ngan VND                      │
│ Hang thang          : 252,50 VND                         │
│ Trang thai          : Dang vay                           │
│ Ngay bat dau        : 22/10/2025                         │
│ Ngay den han        : 22/10/2026                         │
│ Mo ta               : nao co thi tra                     │
├──────────────────────────────────────────────────────────┤
│ Thang con lai       : 12 thang                           │
└──────────────────────────────────────────────────────────┘


TONG QUAN KHOAN VAY:
 - Tong da vay: 3000,00 VND
 - Da tra: 0,00 VND
 - Con no: 3000,00 VND
 - Qua han: 0 khoan
 - Sap den han: 0 khoan
```

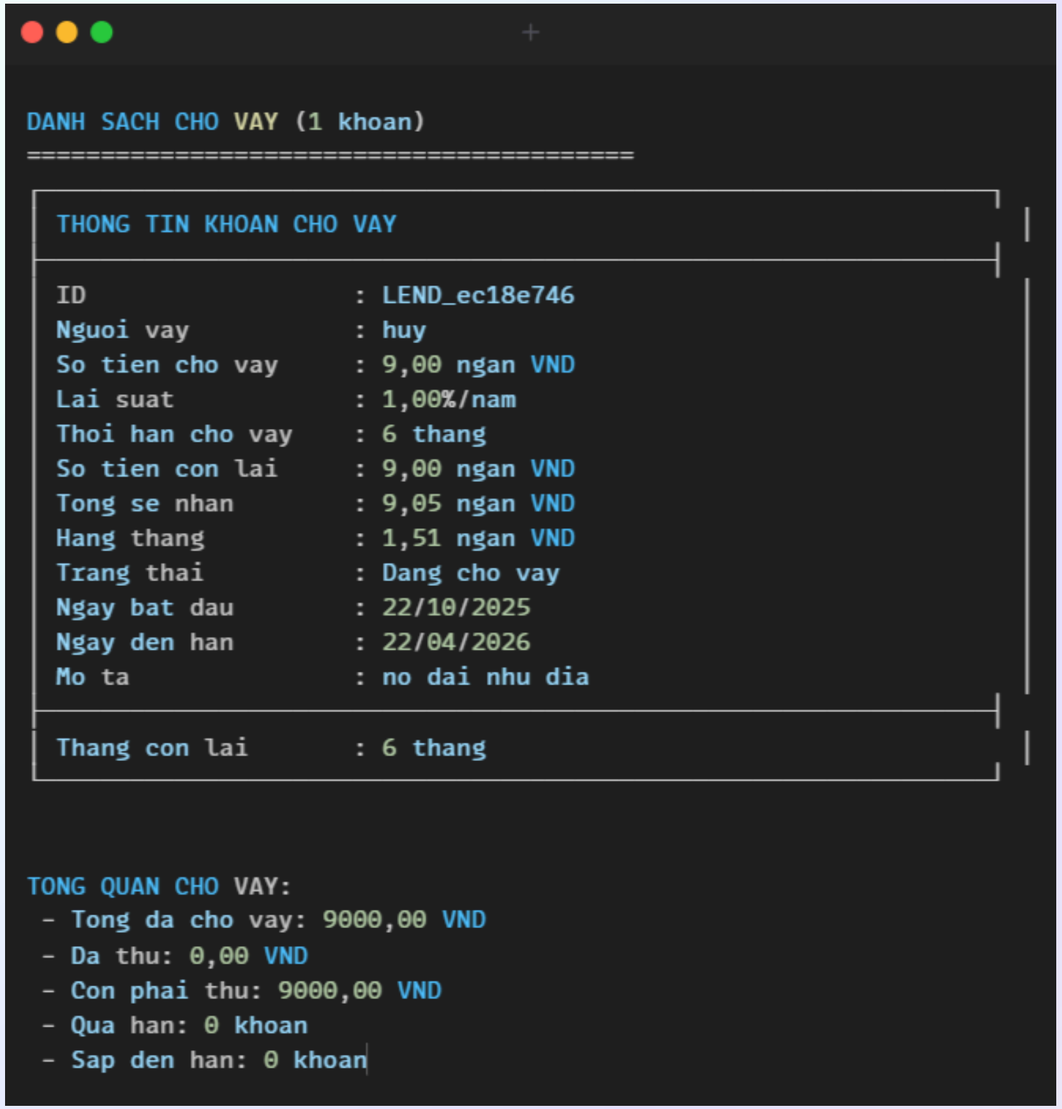
View lending list

Function: Similar to viewing the loan list.

```java
public void displayAllLendings() {
    if (lendings.isEmpty()) {
        System.out.println("Khong co khoan cho vay nao!");
        return;
    }

    System.out.println("\nDANH SACH CHO VAY (" + lendings.size() + " khoan)");
    System.out.println("========================================");

    for (Lending lending : lendings) {
        lending.displayInfo();
        System.out.println();
    }

    // THÊM: Hiển thị tổng quan
    displayLendingSummary();
}

private void displayLendingSummary() {
    double totalLent = lendings.stream().mapToDouble(Lending::getPrincipalAmount).sum();
    double totalRemaining = lendings.stream().mapToDouble(Lending::getRemainingAmount).sum();
    double totalCollected = totalLent - totalRemaining;
    long overdueLendings = lendings.stream().filter(Lending::isOverdue).count();
    long dueSoonLendings = lendings.stream().filter(Lending::isDueSoon).count();

    System.out.println("\nTONG QUAN CHO VAY:");
    System.out.printf(" - Tong da cho vay: %.2f VND\n", totalLent);
    System.out.printf(" - Da thu: %.2f VND\n", totalCollected);
    System.out.printf(" - Con phai thu: %.2f VND\n", totalRemaining);
    System.out.printf(" - Qua han: %d khoan\n", overdueLendings);
    System.out.printf(" - Sap den han: %d khoan\n", dueSoonLendings);
}
```

View lending list

Result



Repay loan

Function: Make partial or full loan payments.

- **Payment validation:**
  - Amount > 0
  - Does not exceed the remaining amount
  - Does not pay off an already paid-off amount
- **Real-time update:** Remaining amount and status are updated immediately
- **Payment history:** Records all payment transactions

```java
//LoanService.java
public boolean repayLoan(String loanId, double amount) {
    Optional<Loan> loanOpt = findLoanById(loanId);
    if (loanOpt.isPresent()) {
        Loan loan = loanOpt.get();

        if (amount <= 0) {
            System.out.println("So tien tra no phai lon hon 0!");
            return false;
        }

        if (amount > loan.getRemainingAmount()) {
            System.out.println("So tien tra no vuot qua so tien con lai!");
            return false;
        }

        if (loan.getStatus().equals("PAID")) {
            System.out.println("Khoan vay da duoc tra het!");
            return false;
        }

        loan.addPayment(amount);
        System.out.printf("Tra no thanh cong! So tien con lai: %.2f VND\n", loan.getRemainingAmount());
        return true;
    } else {
        System.out.println("Khong tim thay khoan vay!");
        return false;
    }
}
```

```java
//Loan.java - Xử lý thanh toán
public void addPayment(double amount) {
    if (amount <= 0) {
        throw new IllegalArgumentException("So tien thanh toan phai lon hon 0");
    }
    if (amount > remainingAmount) {
        throw new IllegalArgumentException(
                String.format("So tien thanh toan (%.2f) vuot qua so tien con no (%.2f)",
                        amount, remainingAmount));
    }

    remainingAmount -= amount;
    Payment payment = new Payment(amount);
    paymentHistory.add(payment);
    updateStatus();

    System.out.printf("Da thanh toan: %.2f VND. Con no: %.2f VND\n", amount, remainingAmount);
}
```

Repay loan

Result

Collect loan payment

Function: Similar to repayment, but it's collecting debt from the borrower.

- **Validation:**
  - The amount must be greater than 0 and not exceed the outstanding debt.
  - The loan must not have been fully repaid.

- **Processing:**
  - Deduct the payment amount from the outstanding debt.
  - Add to payment history.
  - Update status (if fully repaid, change to "PAID").

```java
//LoanService.java
public boolean collectLending(String lendingId, double amount) {
    Optional<Lending> lendingOpt = findLendingById(lendingId);
    if (lendingOpt.isPresent()) {
        Lending lending = lendingOpt.get();

        if (amount <= 0) {
            System.out.println("So tien thu no phai lon hon 0!");
            return false;
        }

        if (amount > lending.getRemainingAmount()) {
            System.out.println("So tien thu no vuot qua so tien con lai!");
            return false;
        }

        if (lending.getStatus().equals("PAID")) {
            System.out.println("Khoan cho vay da duoc thu het!");
            return false;
        }

        lending.addRepayment(amount);
        System.out.printf("Thu no thanh cong! So tien con lai: %.2f VND\n", lending.getRemainingAmount());
        return true;
    } else {
        System.out.println("Khong tim thay khoan cho vay!");
        return false;
    }
}
```

```java
//Lending.java - Xử lý thu nợ
public void addRepayment(double amount) {
    if (amount <= 0) throw new IllegalArgumentException("So tien tra phai lon hon 0");
    if (amount > remainingAmount) {
        throw new IllegalArgumentException(
                String.format("So tien tra (%.2f) vuot qua so tien con no (%.2f)", amount, remainingAmount));
    }

    remainingAmount -= amount;
    Payment payment = new Payment(amount);
    repaymentHistory.add(payment);
    updateStatus();

    System.out.printf("Da nhan tra no: %.2f VND. Con no: %.2f VND\n", amount, remainingAmount);
}
```

Collect loan payment

Result:

Reporting System

Description: Provides detailed and summary financial reports.

Detail:

- Account report
- Financial overview report
- Income and expenditure report
- Loan and lending report
- Monthly report
- Export CSV data

Account report

Function: Generate detailed reports for an account, including overview, categorization by category, recent transactions

- **Multi-dimensional analysis:**
  - Overview: Balance, income, expenses, net cash flow
  - Categorization by category
  - Top 5 largest transactions
- **Visualization:** Use icons and table format

```java
public void generateAccountReport(String accountId) {
    Optional<Account> accountOpt = accountService.findAccountById(accountId);
    if (accountOpt.isEmpty()) {
        System.out.println("Khong tim thay tai khoan!");
        return;
    }

    Account account = accountOpt.get();
    List<Transaction> accountTransactions = transactionService.getAllTransactions().
            .filter(tx -> tx.getAccountId().equals(accountId))
            .collect(Collectors.toList());

    System.out.println("\nBAO CAO TAI KHOAN: " + account.getAccountName());
    System.out.println("========================================");

    account.displayInfo();

    double accountIncome = accountTransactions.stream()
            .filter(tx -> tx.getType() == TransactionType.INCOME)
            .mapToDouble(Transaction::getAmount)
            .sum();

    double accountExpense = accountTransactions.stream()
            .filter(tx -> tx.getType() == TransactionType.EXPENSE)
            .mapToDouble(Transaction::getAmount)
            .sum();

    System.out.printf("\nTONG THU: %,.2f VND\n", accountIncome);
    System.out.printf("TONG CHI: %,.2f VND\n", accountExpense);
    System.out.printf("SO GIAO DICH: %d\n", accountTransactions.size());

    // Top 5 giao dịch lớn nhất
    System.out.println("\nTOP 5 GIAO DICH LON NHAT:");
    accountTransactions.stream()
            .sorted((t1, t2) -> Double.compare(t2.getAmount(), t1.getAmount()))
            .limit(5)
            .forEach(tx -> {
                String symbol = tx.getType() == TransactionType.INCOME ? "↑" : "↓";
                System.out.printf("  %s %-12s: %,.2f VND - %s\n",
                        symbol, tx.getCategory().getName(),
                        tx.getAmount(), tx.getDescription());
            });
}
```

View lending list

Result:

```
================================================================================
TAI KHOAN ID: ACC_60f7071b - nhat
================================================================================
============================================================
| BAO CAO TAI KHOAN: nhat                                  |
------------------------------------------------------------
| TONG QUAN TAI KHOAN                                      |
| +----------------------------------------------------+   |
| | So du hien tai: 48.300,00 VND                      |   |
| | Tong thu nhap:  300,00 VND                         |   |
| | Tong chi tieu:  4.000,00 VND                       |   |
| | Luong tien rong: -3.700,00 VND                     |   |
| +----------------------------------------------------+   |
------------------------------------------------------------
|  PHAN LOAI THEO DANH MUC                                 |
| THU NHAP:                                                |
|   - Luong          : 300,00 VND                          |
| CHI TIEU:                                                |
|   - xay dung co ngoi : 4.000,00 VND                      |
------------------------------------------------------------
| GIAO DICH GAN DAY (5 giao dich moi nhat)                 |
| [-] 22/10/2025 xay dung co ngoi : 4.000,00 mua nha       |
| [+] 22/10/2025 Luong          : 300,00 thu nhap hang thang   |
============================================================
================================================================================
KET THUC BAO CAO TAI KHOAN: ACC_60f7071b
================================================================================


================================================================================
TAI KHOAN ID: ACC_00ed0ebe - khai
================================================================================
Khong co giao dich nao cho tai khoan nay!
================================================================================
```

Financial overview report

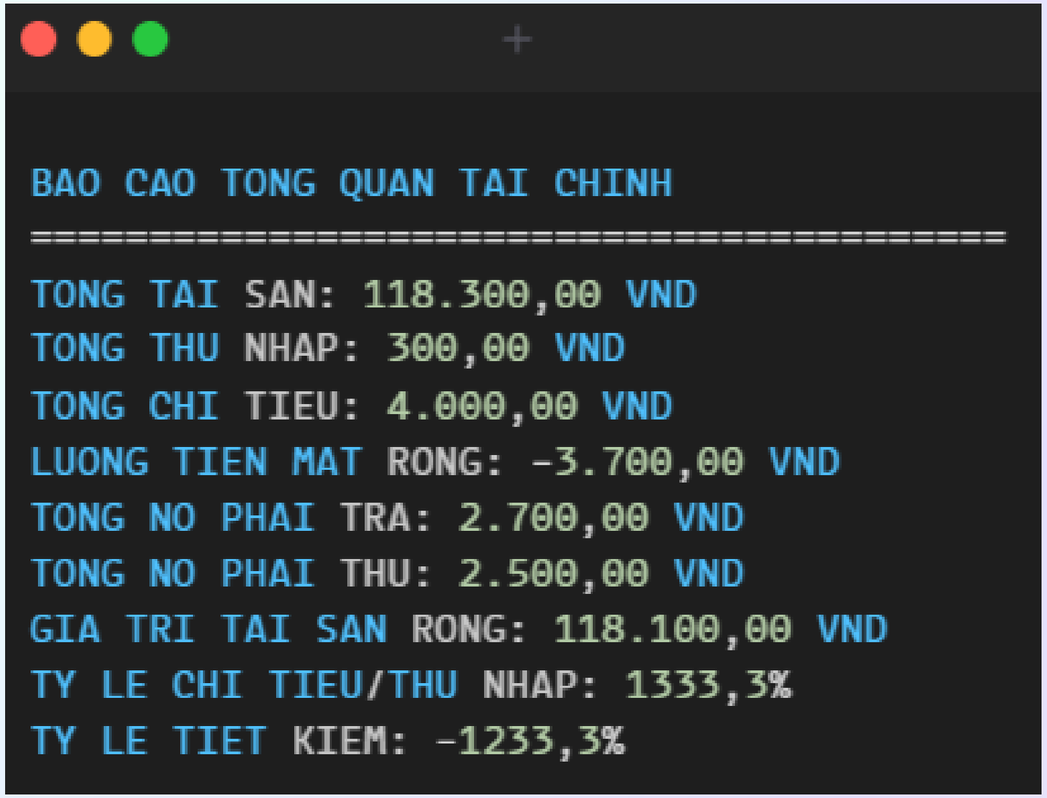
Function: Summarizes total assets, income, expenses, and cash flow

- **Key financial indicators:**
  - Total assets
  - Net cash flow (Income - Expenses)
  - Net worth (Assets + Receivables - Payables)
  - Spending/income ratio
  - Savings rate

```java
public void generateFinancialOverview() {
    System.out.println("\nBAO CAO TONG QUAN TAI CHINH");
    System.out.println("========================================");

    double totalBalance = accountService.getTotalBalance();
    double totalIncome = transactionService.getTotalIncome();
    double totalExpense = transactionService.getTotalExpense();
    double netCashFlow = totalIncome - totalExpense;
    double totalLoans = loanService.getTotalLoanAmount();
    double totalLendings = loanService.getTotalLendingAmount();
    double netWorth = totalBalance + totalLendings - totalLoans;

    System.out.printf("TONG TAI SAN: %,.2f VND\n", totalBalance);
    System.out.printf("TONG THU NHAP: %,.2f VND\n", totalIncome);
    System.out.printf("TONG CHI TIEU: %,.2f VND\n", totalExpense);
    System.out.printf("LUONG TIEN MAT RONG: %,.2f VND\n", netCashFlow);
    System.out.printf("TONG NO PHAI TRA: %,.2f VND\n", totalLoans);
    System.out.printf("TONG NO PHAI THU: %,.2f VND\n", totalLendings);
    System.out.printf("GIA TRI TAI SAN RONG: %,.2f VND\n", netWorth);

    // Phân tích tỷ lệ
    if (totalIncome > 0) {
        double expenseRatio = (totalExpense / totalIncome) * 100.0;
        double savingsRatio = 100.0 - expenseRatio;
        System.out.printf("TY LE CHI TIEU/THU NHAP: %.1f%%%n", expenseRatio);
        System.out.printf("TY LE TIET KIEM: %.1f%%%n", savingsRatio);
    } else {
        System.out.println("TY LE CHI TIEU/THU NHAP: N/A (khong co du lieu thu nhap)");
        System.out.println("TY LE TIET KIEM: N/A (khong co du lieu thu nhap)");
    }
}
```

Financial overview report

Result:



Income and expenditure report

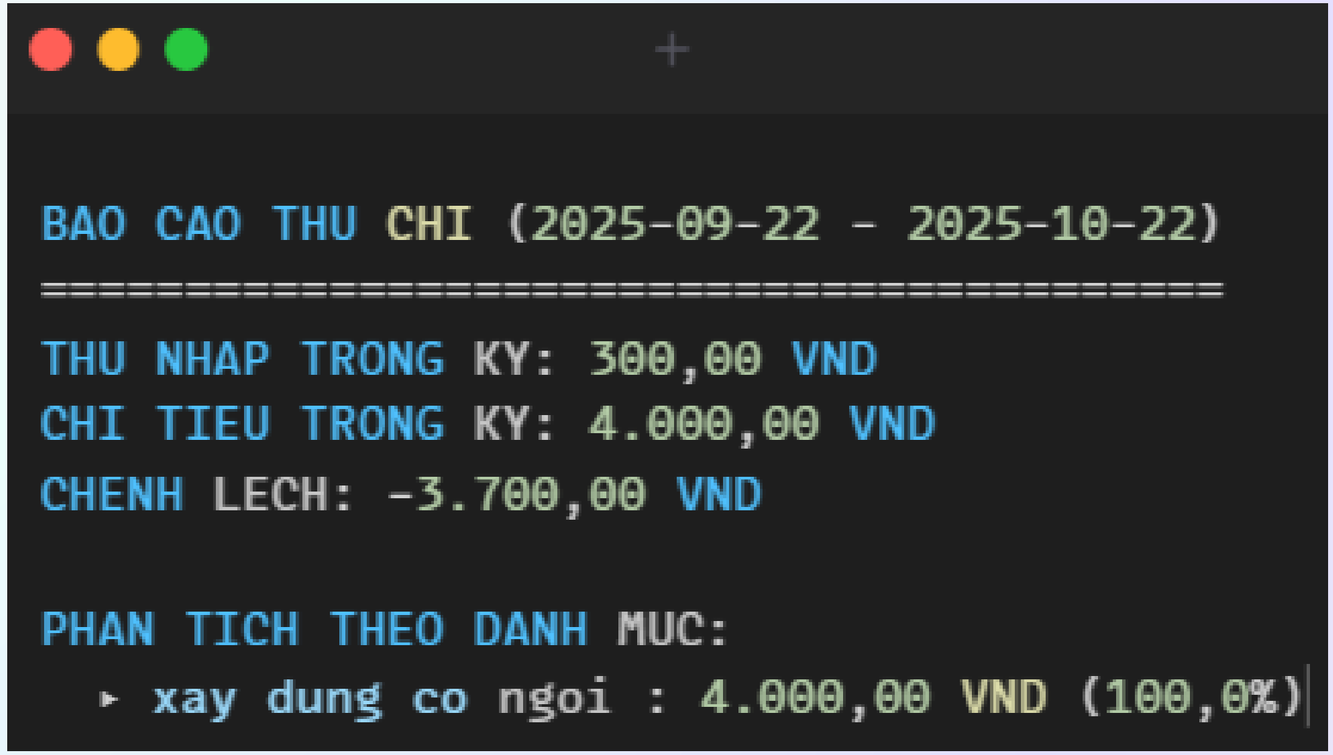
Function: Consolidates all loan-related entries.

- **Track over time:** Can select specific time periods
- **Category analysis:** Shows % of spending for each category
- **Income-expense comparison:** Clearly shows the difference

```java
// Báo cáo thu nhập và chi tiêu theo khoảng thời gian
public void generateIncomeExpenseReport(LocalDate startDate, LocalDate endDate) {
    System.out.println("\nBAO CAO THU CHI (" + startDate + " - " + endDate + ")");
    System.out.println("========================================");

    List<Transaction> transactions = transactionService.getAllTransactions().stream()
            .filter(tx -> !tx.getDate().toLocalDate().isBefore(startDate) &&
                    !tx.getDate().toLocalDate().isAfter(endDate))
            .collect(Collectors.toList());

    double periodIncome = transactions.stream()
            .filter(tx -> tx.getType() == TransactionType.INCOME)
            .mapToDouble(Transaction::getAmount)
            .sum();

    double periodExpense = transactions.stream()
            .filter(tx -> tx.getType() == TransactionType.EXPENSE)
            .mapToDouble(Transaction::getAmount)
            .sum();

    System.out.printf("THU NHAP TRONG KY: %,.2f VND\n", periodIncome);
    System.out.printf("CHI TIEU TRONG KY: %,.2f VND\n", periodExpense);
    System.out.printf("CHENH LECH: %,.2f VND\n", periodIncome - periodExpense);

    // Phân tích theo danh mục
    System.out.println("\nPHAN TICH THEO DANH MUC:");
    Map<String, Double> expenseByCategory = transactions.stream()
            .filter(tx -> tx.getType() == TransactionType.EXPENSE)
            .collect(Collectors.groupingBy(
                    tx -> tx.getCategory().getName(),
                    Collectors.summingDouble(Transaction::getAmount)
            ));

    expenseByCategory.entrySet().stream()
            .sorted(Map.Entry.<String, Double>comparingByValue().reversed())
            .forEach(entry -> {
                double percentage = (entry.getValue() / periodExpense) * 100;
                System.out.printf("  • %-15s: %,.2f VND (%.1f%%)\n",
                        entry.getKey(), entry.getValue(), percentage);
            });
}
```

Income and expenditure report

Result



Loan and lending report

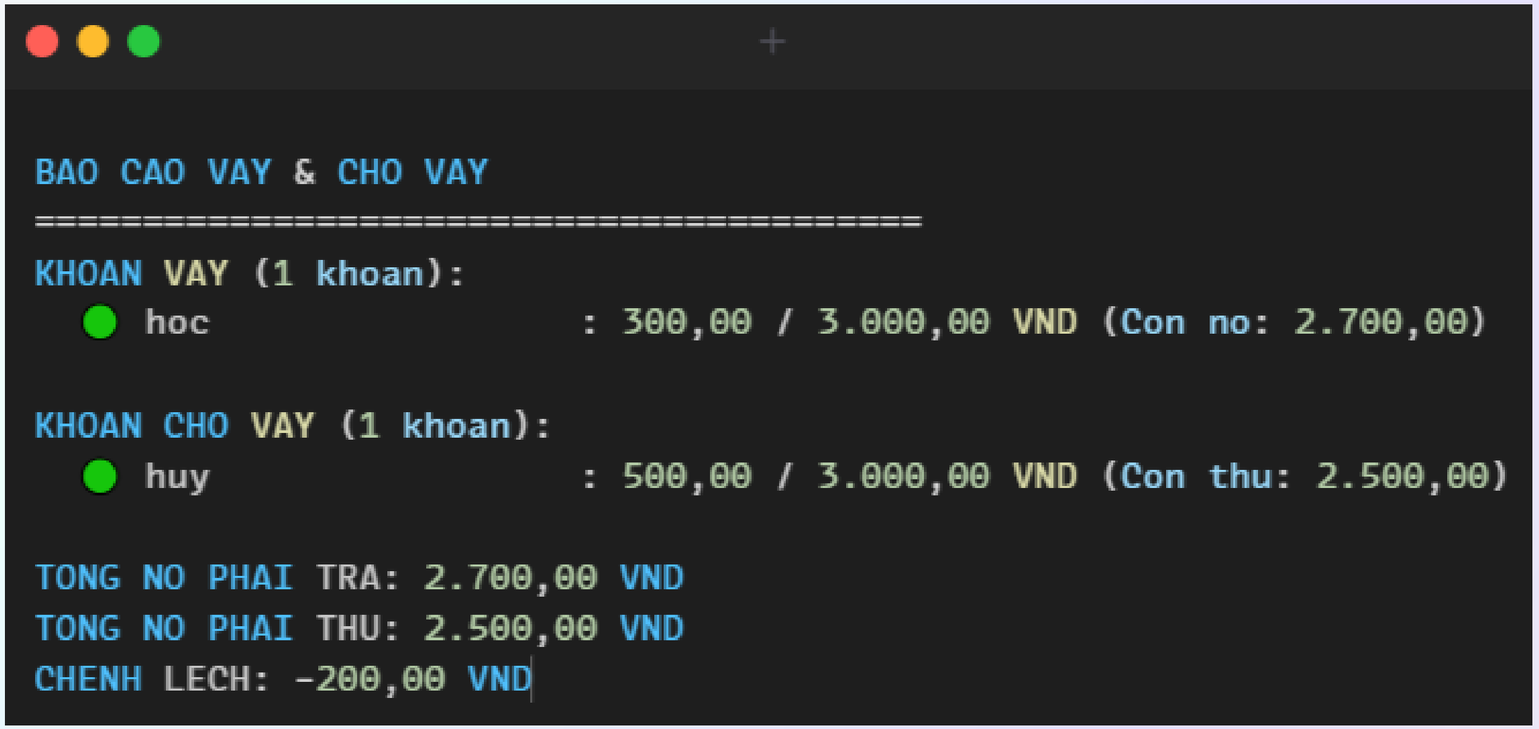
Function: Display an overview of loans and borrowings, including status and remaining amounts

- **Visual status:** Red/green icon for overdue/active
- **Payment progress:** Display amount paid/total amount
- **Risk summary:** Count of overdue accounts

```java
// Báo cáo khoản vay và cho vay
public void generateLoanReport() {
    System.out.println("\nBAO CAO VAY & CHO VAY");
    System.out.println("=====================================");

    // Khoản vay
    List<Loan> loans = loanService.getAllLoans();
    System.out.println("KHOAN VAY (" + loans.size() + " khoan):");
    loans.forEach(loan → {
        String statusIcon = loan.isOverdue() ? "🔴" : "🟢";
        System.out.printf("  %s %-20s: %,.2f / %,.2f VND (Con no: %,.2f)\n",
                statusIcon, loan.getLenderName(),
                loan.getPrincipalAmount() - loan.getRemainingAmount(),
                loan.getPrincipalAmount(), loan.getRemainingAmount());
    });

    // Khoản cho vay
    List<Lending> lendings = loanService.getAllLendings();
    System.out.println("\nKHOAN CHO VAY (" + lendings.size() + " khoan):");
    lendings.forEach(lending → {
        String statusIcon = lending.isOverdue() ? "🔴" : "🟢";
        System.out.printf("  %s %-20s: %,.2f / %,.2f VND (Con thu: %,.2f)\n",
                statusIcon, lending.getBorrowerName(),
                lending.getPrincipalAmount() - lending.getRemainingAmount(),
                lending.getPrincipalAmount(), lending.getRemainingAmount());
    });

    // Tổng hợp
    double totalDebt = loanService.getTotalLoanAmount();
    double totalReceivable = loanService.getTotalLendingAmount();
    System.out.printf("\nTONG NO PHAI TRA: %,.2f VND\n", totalDebt);
    System.out.printf("TONG NO PHAI THU: %,.2f VND\n", totalReceivable);
    System.out.printf("CHENH LECH: %,.2f VND\n", totalReceivable - totalDebt);
}
```

Loan and lending report

Result



```
BAO CAO VAY & CHO VAY
=====================================
KHOAN VAY (1 khoan):
  🟢 hoc                  : 300,00 / 3.000,00 VND (Con no: 2.700,00)

KHOAN CHO VAY (1 khoan):
  🟢 huy                  : 500,00 / 3.000,00 VND (Con thu: 2.500,00)

TONG NO PHAI TRA: 2.700,00 VND
TONG NO PHAI THU: 2.500,00 VND
CHENH LECH: -200,00 VND
```

Monthly report

Function: Report income and expenses and daily balance fluctuations within the month.

- **Daily fluctuations:** Track balances day by day
- **Focus on the month:** Automatically calculate the first/last day of the month

```java
// Báo cáo hàng tháng
public void generateMonthlyReport(int year, int month) {
    LocalDate startDate = LocalDate.of(year, month, 1);
    LocalDate endDate = startDate.withDayOfMonth(startDate.lengthOfMonth());

    System.out.println("\nBAO CAO THANG " + month + "/" + year);
    System.out.println("========================================");

    generateIncomeExpenseReport(startDate, endDate);

    // Thống kê hàng ngày
    System.out.println("\nBIEN DONG HANG NGAY:");
    Map<LocalDate, Double> dailyBalance = new TreeMap<>();

    // Tính toán số dư hàng ngày (đơn giản)
    double runningBalance = accountService.getTotalBalance();
    for (int day = startDate.lengthOfMonth(); day >= 1; day--) {
        LocalDate currentDate = LocalDate.of(year, month, day);
        dailyBalance.put(currentDate, runningBalance);
        // Giả lập biến động (trong thực tế cần tính toán chính xác)
        runningBalance -= 100000; // Giả định mỗi ngày giảm 100k
    }

    dailyBalance.entrySet().stream()
            .limit(10) // Hiển thị 10 ngày đầu
            .forEach(entry -> {
                System.out.printf("  %s: %,.2f VND\n", entry.getKey(), entry.getValue());
            });
}
```

Monthly report

Result

```
BAO CAO THANG 10/2025
========================================

BAO CAO THU CHI (2025-10-01 - 2025-10-31)
========================================
THU NHAP TRONG KY: 300,00 VND
CHI TIEU TRONG KY: 4.000,00 VND
CHENH LECH: -3.700,00 VND

PHAN TICH THEO DANH MUC:
  ▸ xay dung co ngoi : 4.000,00 VND (100,0%)

BIEN DONG HANG NGAY:
  2025-10-01: -2.881.700,00 VND
  2025-10-02: -2.781.700,00 VND
  2025-10-03: -2.681.700,00 VND
  2025-10-04: -2.581.700,00 VND
  2025-10-05: -2.481.700,00 VND
  2025-10-06: -2.381.700,00 VND
  2025-10-07: -2.281.700,00 VND
  2025-10-08: -2.181.700,00 VND
  2025-10-09: -2.081.700,00 VND
  2025-10-10: -1.981.700,00 VND
```

Export CSV data

Function: Export account and transaction data to CSV format.

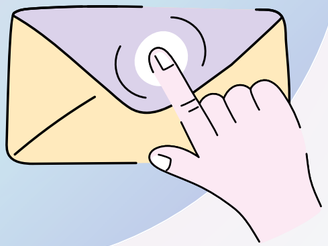
- **Standard format:** CSV easily importable into Excel/Google Sheets
- **Complete data:** Both account and transactions
- **Clear structure:** Header + consistent data forma

```java
// Xuất dữ liệu đơn giản (có thể mở rộng để xuất file)
public void exportToCSV() {
    System.out.println("\nXUAT DU LIEU (CSV Format):");
    System.out.println("====================================");

    // Xuất danh sách tài khoản
    System.out.println("TAI_KHOAN_ID,TEN,LOAI,SO_DU");
    accountService.getAllAccounts().forEach(acc -> {
        System.out.printf("%s,%s,%s,%.2f\n",
                acc.getAccountId(), acc.getAccountName(),
                acc.getAccountType(), acc.getBalance());
    });

    // Xuất danh sách giao dịch
    System.out.println("\nGIAO_DICH_ID,TAI_KHOAN,LOAI,SOTIEN,DANHMUC,NGAY");
    transactionService.getAllTransactions().forEach(tx -> {
        System.out.printf("%s,%s,%s,%.2f,%s,%s\n",
                tx.getTransactionId(), tx.getAccountId(),
                tx.getType().name(), tx.getAmount(),
                tx.getCategory().getName(), tx.getFormattedDate());
    });
}
```

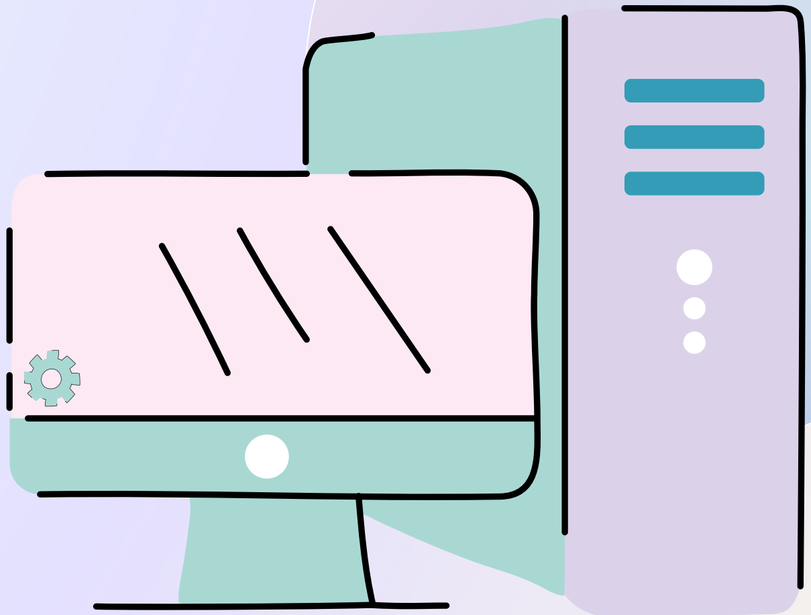
Export CSV data

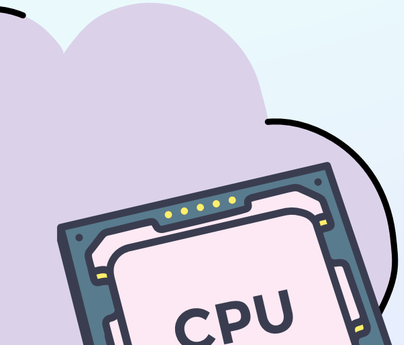
Result

```
XUAT DU LIEU (CSV Format):
========================================
TAI_KHOAN_ID,TEN,LOAI,SO_DU
ACC_60f7071b,nhat,BANK,48300,00
ACC_00ed0ebe,khai ,SAVINGS,70000,00

GIAO_DICH_ID,TAI_KHOAN,LOAI,SOTIEN,DANHMUC,NGAY
TRX_3b305507,ACC_60f7071b,INCOME,300,00,Luong,22/10/2025 14:53:42
TRX_954bf5fe,ACC_60f7071b,EXPENSE,4000,00,xay dung co ngoi ,22/10/2025 14:54:39
```



4. Summary

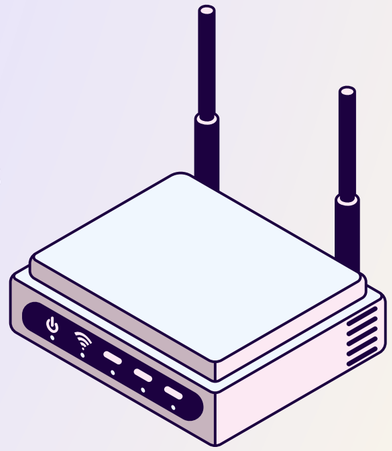




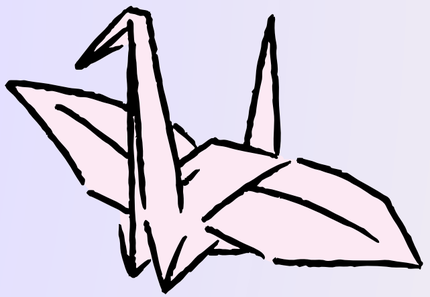
4.1. Self-Assessment

Strengths:

- Clear architecture: MVC-based package structure, easy to maintain and extend
- Feature completeness: Covers most core personal finance features
- Robust error handling: Custom exception mechanisms
- User-friendly interface: Aesthetic console layout using Unicode characters
- Strong validation: Comprehensive input verification
- Vietnamese support: Full Vietnamese UI and notifications
- Accurate calculations: Reliable financial and interest computations

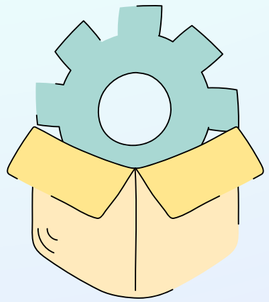


4.1. Self-Assessment



Limitations:

- Data persistence: No file/database storage yet
- Interface: Console-only, no GUI support
- Security: No encryption for sensitive data
- Performance: Not optimized for large data volumes
- Missing advanced features: Budgeting, financial goals, automated alerts



4.2. Future Developmen

Short Term

1. Data Storage:

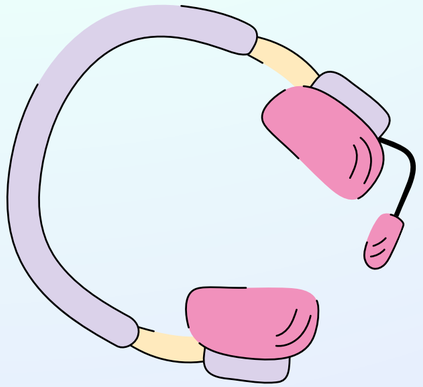
- Database integration: MySQL/PostgreSQL for persistent storage
- File storage: Support for JSON/XML or embedded DB (SQLite)

2 Interface Improvements:

- Desktop GUI: JavaFX/Swing for improved UX
- Web application: Spring Boot + React/Angular

3 New Features:

- Budget management
- Payment reminders
- Trend analysis & visualization
- Multi-currency support
- Bank statement import (CSV/OFX)



4.2. Future Developmen

medium term

1. Enhanced Security:

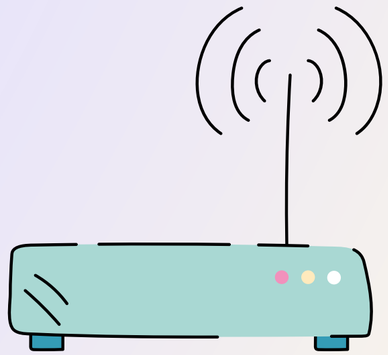
- User authentication & access control
- Data encryption

2. Analytics:

- Spending trend charts
- Financial forecasting
- Behavior analysis

3. Cross-platform:

- Web-based version (Spring Boot)
- Mobile app (Android)



4.2. Future Developmen

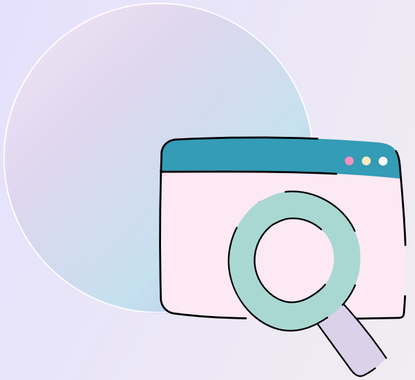
Long Term

1. Smart Features:

- AI-based transaction categorization
- Financial optimization suggestions
- Smart alerts

2. External Integration:

- Public API for third-party systems
- Plugin ecosystem
- Bank statement import automation



3. Community Features:

- Expense benchmarking against average users
- Knowledge sharing and saving tips

Thank You