Online Bank Design Documentation

Zhihao Gu U55787297 Pengchao Yuan U50962567 Huiwen He U96391336

Function Analysis

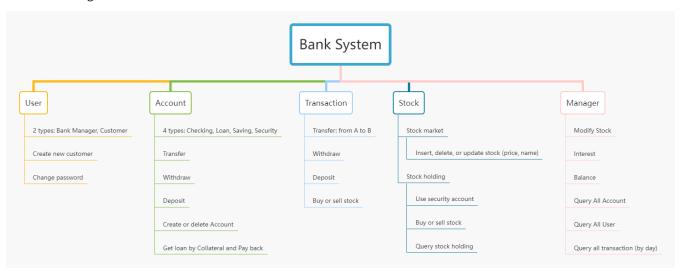
The function can be divided into manager and customer. For detailed information, please refer to bank.pdf.

Customer:

- 1. Open/close account with a specific currency.
- 2. Get loan by adding collaterals and get collaterals back by paying money back through loan account.
- 3. View account balance, deposit/withdraw money, transfer money to specific account.
- 4. Buy/sell stock using security account (Security account has special requirement).

Manager:

- 1. View all accounts transactions.
- 2. Charge fees for account open, transfer money, withdraw.
- 3. Give interests to saving accounts which have a minimum balance.
- 4. Manage the stock market: add stock, adjust stock price.
- 5. Charge interests to all loan account.



Design

Design pattern:

1. MVC pattern

The project is organized by MVC pattern.

- V: View. All things about GUI.
- M: Model. All things about DB.
- C: Controller. Connect the M and V, which means all the logic things are there.

Reason: If needed, View and DB can be easily switched, for example, changing the SQLite to Oracle to full fill the high demand.

2. Singleton Pattern

For DAO.java. Double-checked locking

Reason: Connection with database is expensive. There is no need to create a new connection.

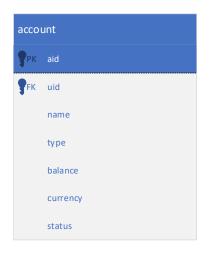
Database: SQLite

- 1. Using keys and foreign keys to make sure all data are valid.
- 2. Using delete mark instead of deleting real data for data recovery and rollback.

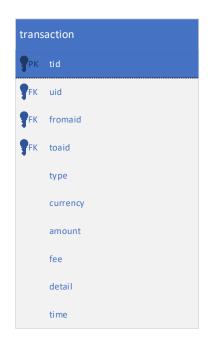




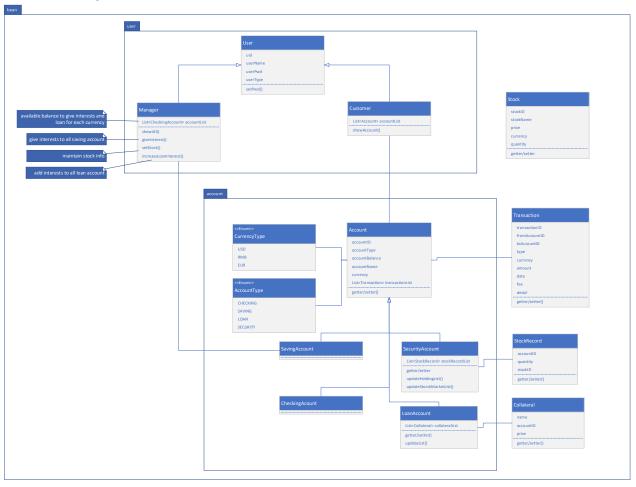




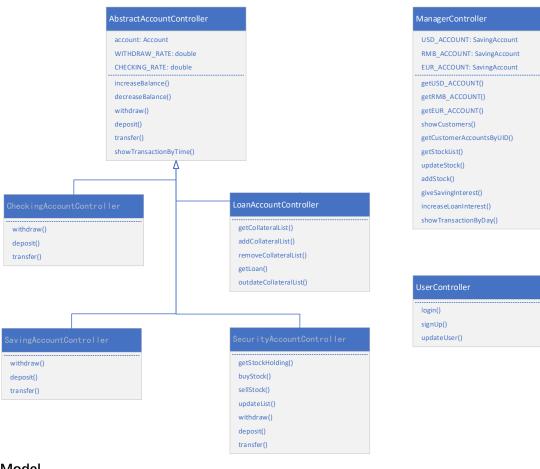




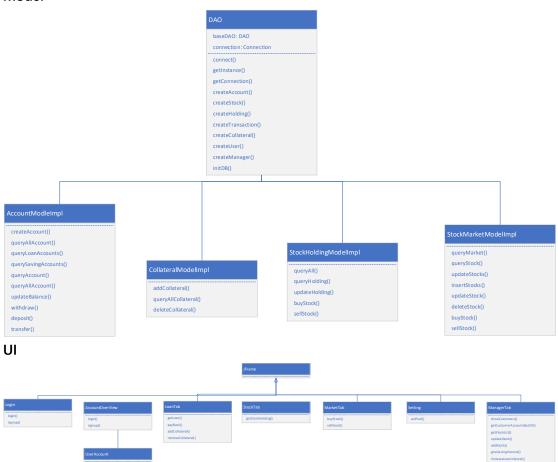
Beans for entity



Controller



Model



Detail about each class: -db bank.db (SQLite database) -lib sglite-jdbc-3.36.0.3.jar (SQLite JDBC dependency) -src main.java (Entrance) -bean (Java bean for entity) Collateral.java (collateral for loan) Stock.java (represents one stock) StockHolding.java (record the stock quantity for everyone held) Transaction.java (represents one transaction record) account Account.java (abstract class) CheckingAccount.java (checking account) LoanAccount.java (loan account) SavingAccount.java (saving account) SecurityAccount.java (security account) -user Customer.java (customer) Manager.java (bank manager) User.java (abstract class) controller (controller for MVC) AbstractAccountController.java (Abstract Controller contain common func and attr) AccountOverviewController.java (Show and manage User account) CheckingAccountController.java (Controller for Checking Account) LoanAccountController.java (Controller for Loan Account) ManagerController.java (Controller for Manager SavingAccountController.java (Controller for Saving Account) SecurityAccountController.java (Controller for Security Account) UserController.java (Controller for managing User) gui (View for MVC) GUIAccount.java (UI for each account) GUIAIIAccounts.java (UI for each user's account) GUIDeposit.java (UI for deposit) (UI for main page after login) GUIHomePage.java GUIInterestLoan.java (UI for add interests in loan account) GUILoan.java (UI for Loan tab)

(main class for UI)

(UI for create account)

GUIMain.java

GUINewAccount.java

GUIRegistration.java (UI for user registration)
GUISetting.java (UI for user setting)
GUIStock.java (UI for set a stock)
GUIStockHolding.java (UI for user's stock)

GUIStockManagement.java (UI for stock management)
GUITransaction.java (UI for each transaction)
GUITransactionOverView.java (UI for transaction overview)
GUITransfer.java (UI for money transfer)

GUIUser.java (UI for user info)

GUIUserOverView.java (UI for showing all users)
GUIUserStock.java (UI for showing user)
GUIViewAccount.java (UI for showing account)

GUIWithdraw.java (UI for withdraw)

-model (model for MVC)

AccountModel.java (Account model: create or delete account, withdraw, deposit, query)

AccountModelImpl.java (implementation of AccountModel)
CollateralModel.java (Collateral model: query, add, delete)
CollateralModelImpl.java (implementation of CollateralModel)
DAO.java (Connect to database, create table)

StockHoldingModel.java (Stock holding model: query, buy or sell stock)

StockHoldingModelImpl.java (implementation of StockHoldingModel)

StockMarketModel.java (Stock market model: query, update or delete stock, buy or sell stock)

StockMarketModelImpl.java (implementation of StockMarketModel)
TransactionModel.java (Transaction model: insert, query)
TransactionModelImpl.java (implementation of TransactionModel)
UserModel.java (User sign in, sign up, update password)

UserModelImpl.java (implementation of UserModel)

Extendibility:

The project can be update in the future:

A better UI.

A better storage solution with encryption.

More ways to view Transactions and monitor accounts.