

# Online Bank Design Documentation

Zhihao Gu U55787297

Pengchao Yuan U50962567

Huiwen He U96391336

## Function Analysis

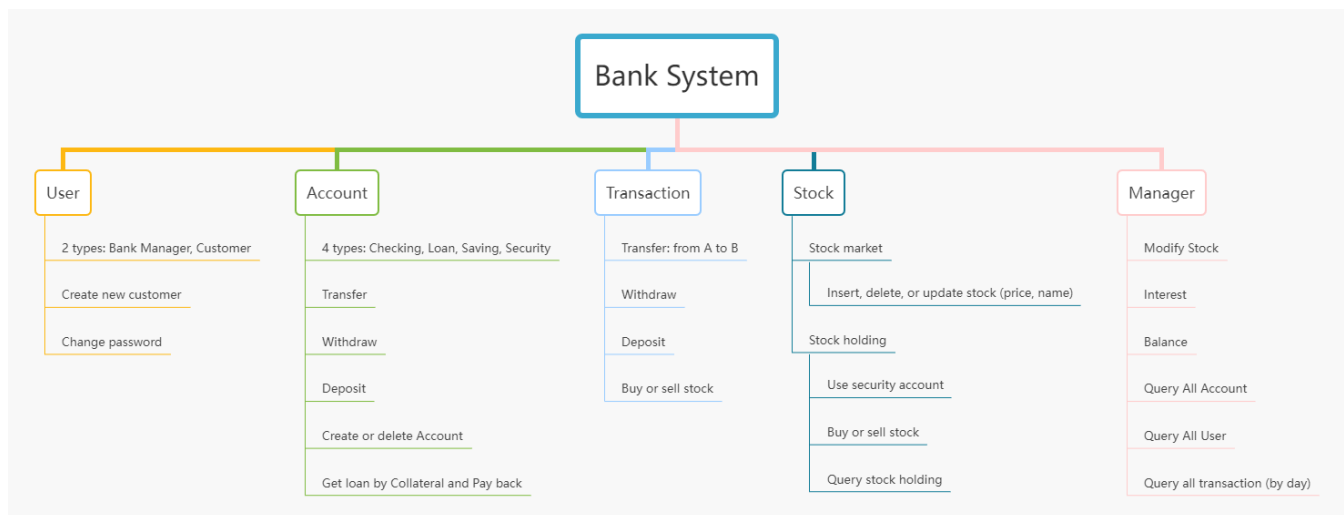
The function can be divided into manager and customer. For detailed information, please refer to bank.pdf.

### 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.





stock	
 PK	sid
name	
price	
quantity	

collateral	
 PK	id
 FK	uid
name	
value	

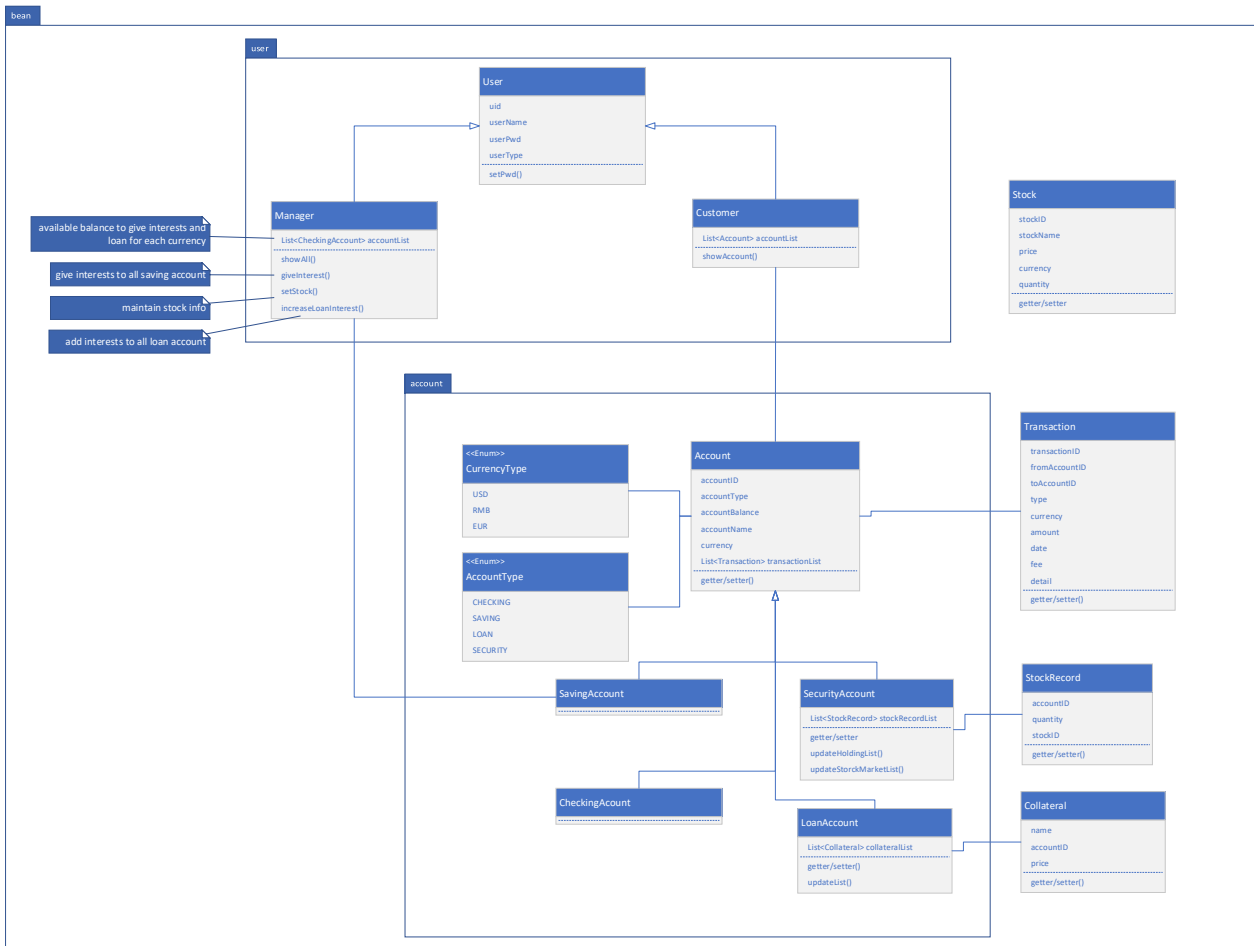
stockHolding	
 PK	uid
 PK	sid
quantity	

account	
 PK	aid
 FK	uid
name	
type	
balance	
currency	
status	

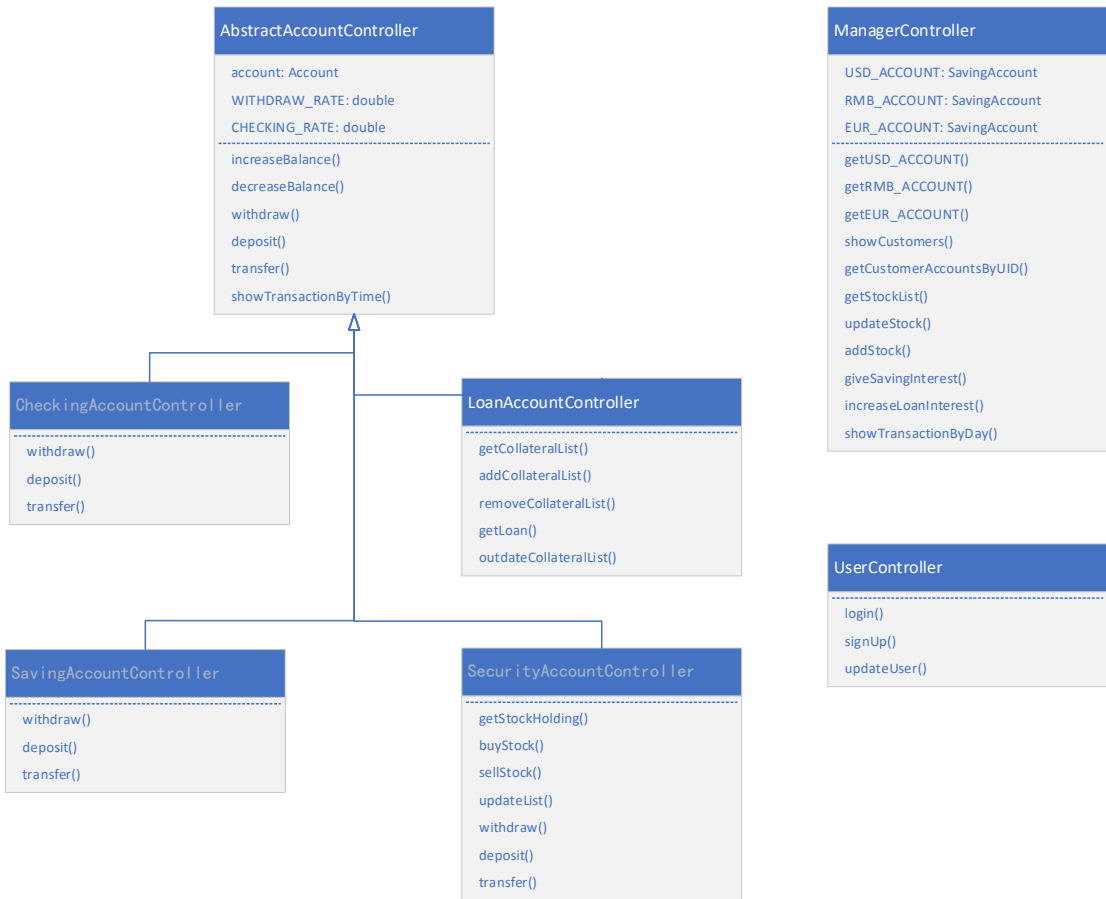
user	
 PK	uid
type	
username	
pwd	
status	

transaction	
 PK	tid
 FK	uid
 FK	fromaid
 FK	toaid
type	
currency	
amount	
fee	
detail	
time	

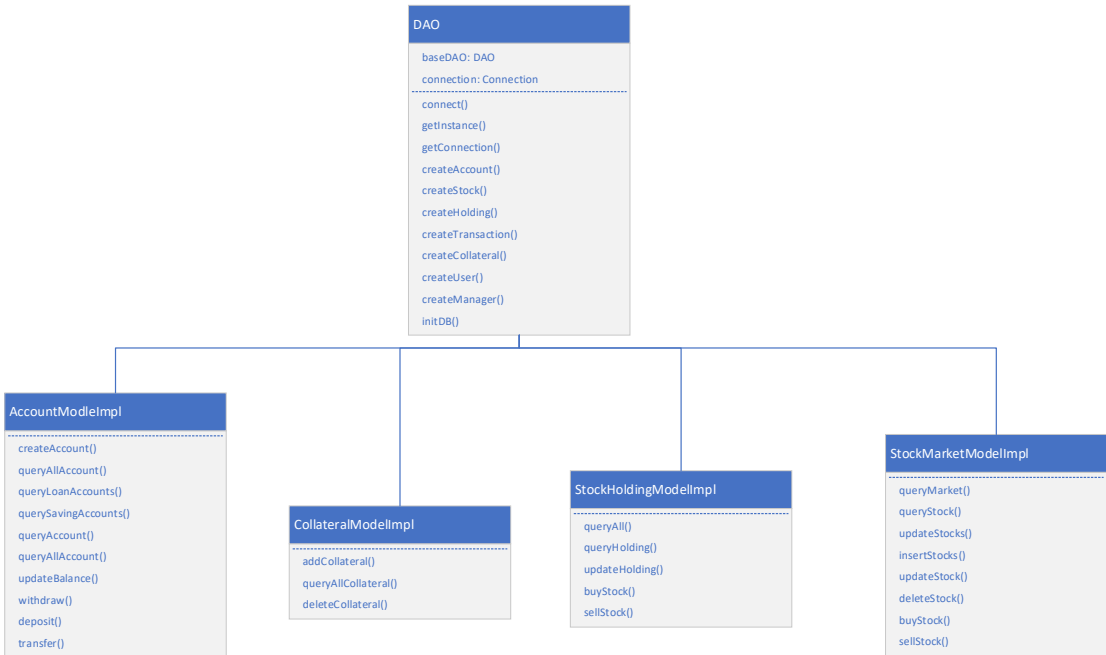
## Beans for entity



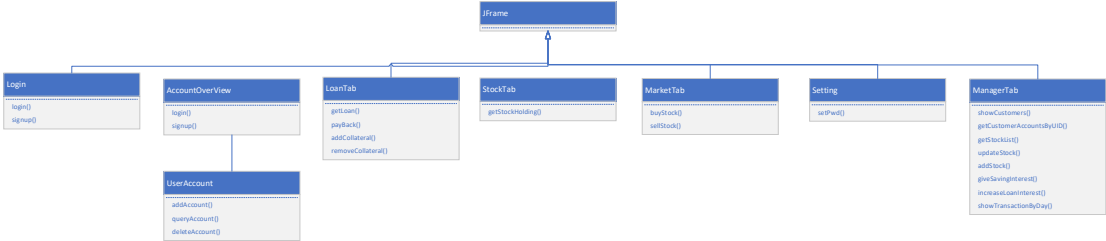
Controller



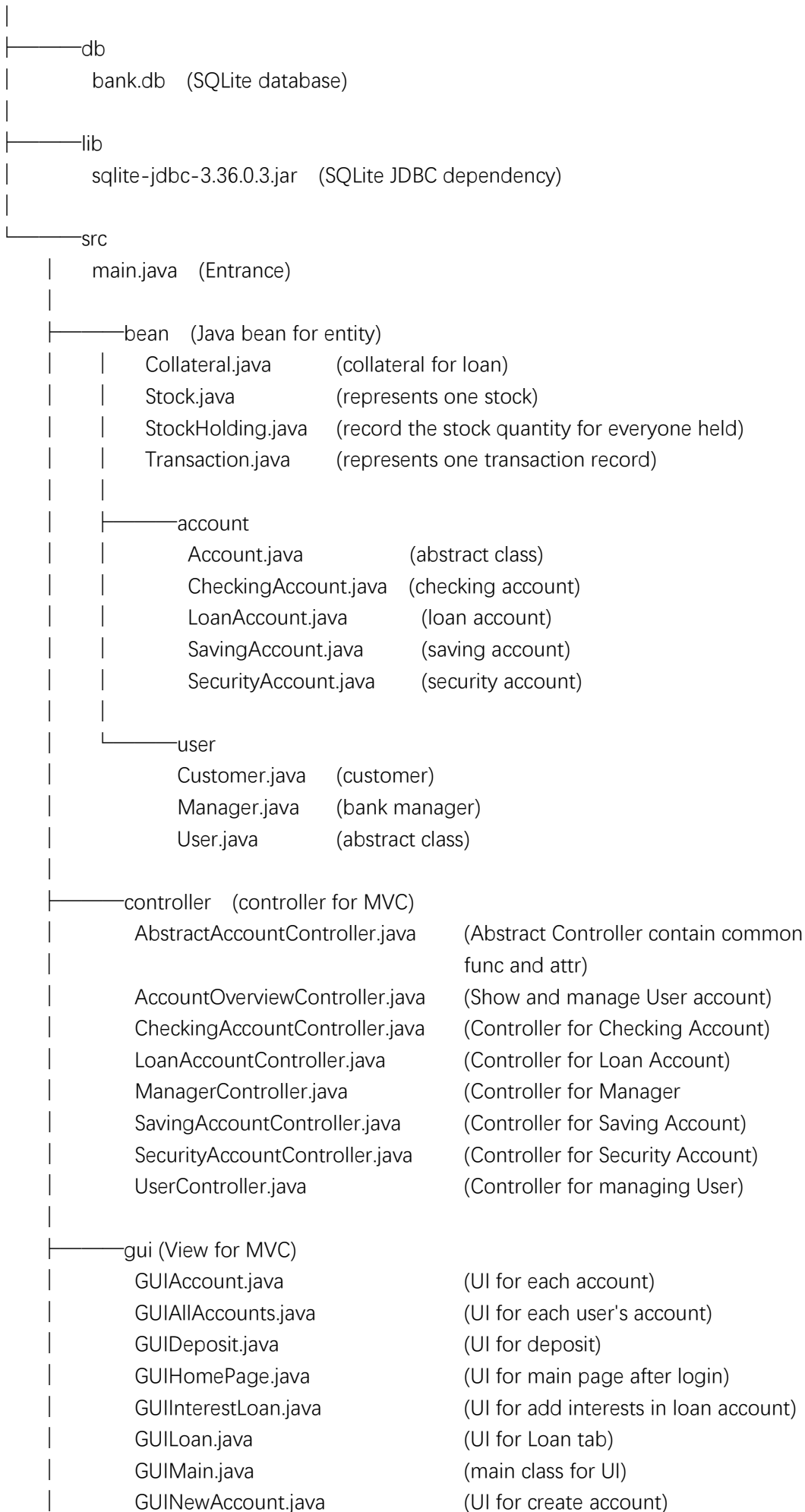
Model



UI



## Detail about each class:



	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.