Contents

Overview

For our project, we decided to implement a banking system. We plan to support Savings, Checking and several types of loans within the application. We will have two types of User Interfaces currently planned: one is an ATM and the other is a web based banking interface. The user side of the application will invoke RMI to call methods on the Server.

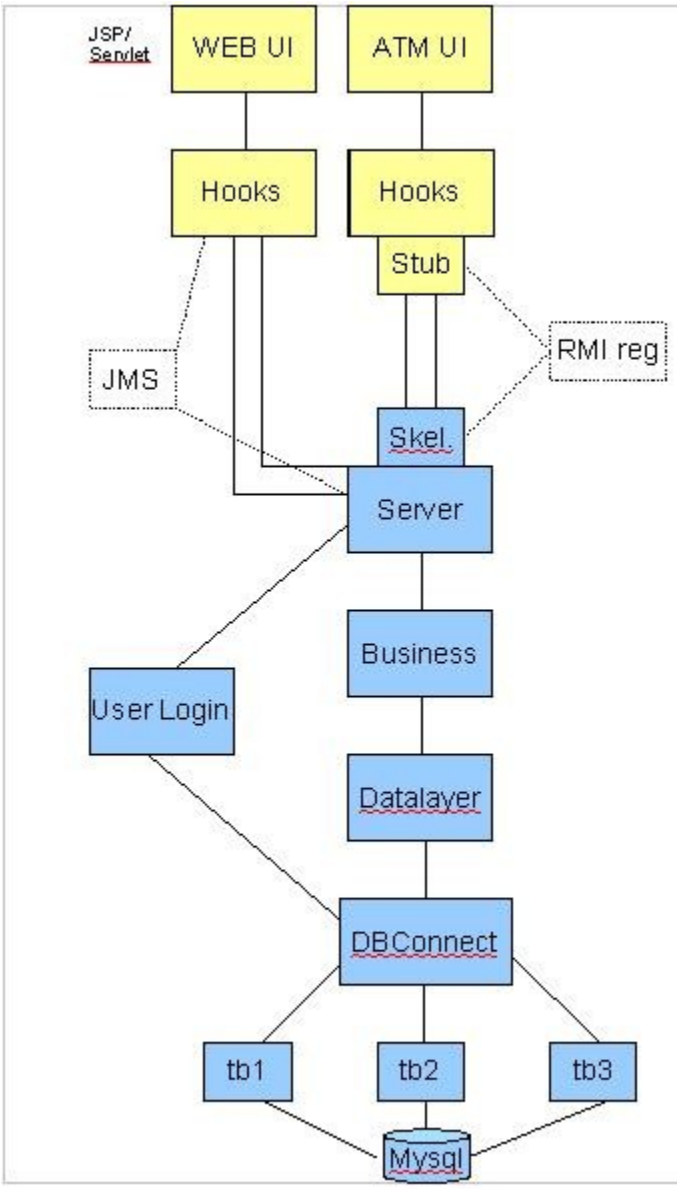
All the user data (names, account numbers, etc) will be stored in a MySQL database. This database will be accessible by the Server side of our program. There will be multiple clients connecting simultaneously to the server over a network connection. The server will parse and analyze user input, query the database, and return output back to the user.

Advanced functionality at the ATM will consist of the ability of users with various loans to make payments to those loans, monitor the existing balance on the loan, and withdraw cash from existing credit lines provided through said loans. From the web UI the user will be able to view account and loan transaction histories and transfer funds. Authentication at the ATM will consist of an account or card number and PIN, and also at the web page (username and password).

We have chosen to use a MySQL database because it is free, and can handle synchronicity by itself (very important for any database, but especially for a bank).

The bulk of the 'Business' portion of our application will be built using Java and RMI to facilitate remote access. The web interface will use Java Server Pages and probably PHP.

Architecture



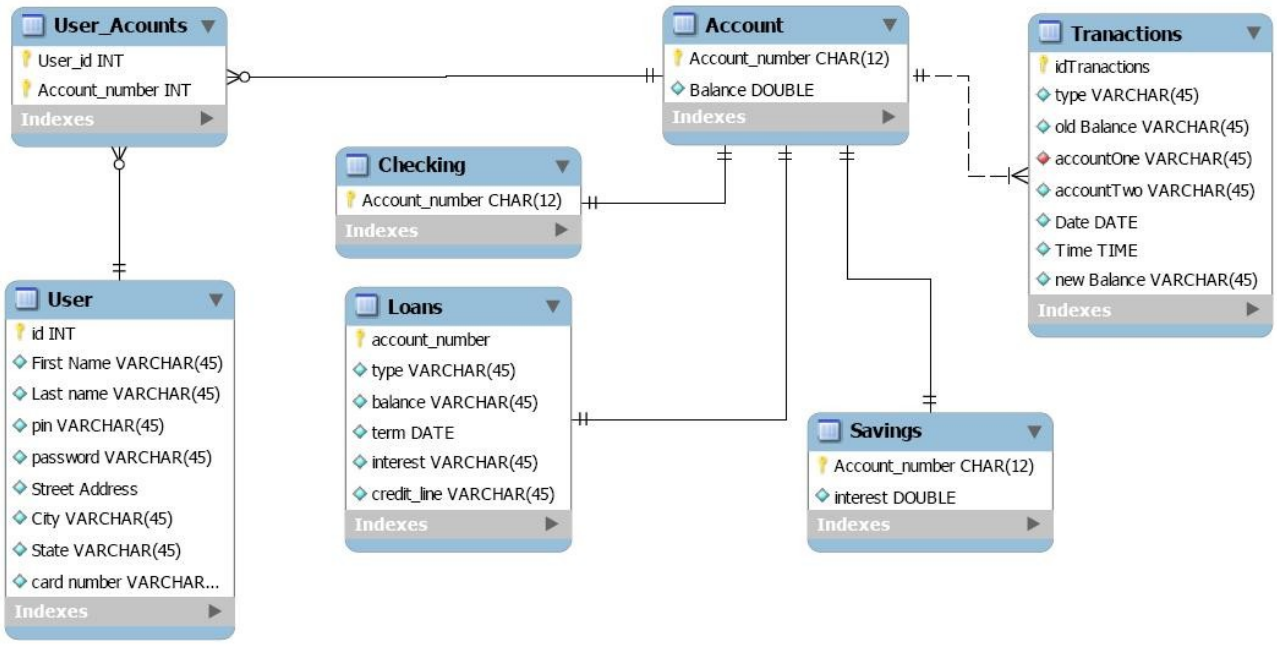
Architecture Notes:

- The DBConnect and tb1-3 elements will provide access to the database and function as our DBMS.
- The Data Layer classes/objects will interface between the main portion of the server and the DBMS.
- The Business and Authentication (not shown) elements will perform the main 'heavy lifting' functions on the server side of the application.

- The Server and skeleton classes will provide networked access to the server side of the application.
- JMS and RMI Registry will be used to facilitate communications between the client and server sides of the application.
- The “hooks” and stub class elements exist to provide the connectivity portion of the client side of the application.
- The Web UI and ATM UI elements shown are place holders for the GUI the user will interact with on deployment.

Note: This is a ‘first blush’ take on our class/object structure, few of these elements have actually been designed and as such this diagram is subject to change (will be noted in the revision history).

Data Design



Data Dictonary

We will be using a MySQL database in order to provide data persistence functionality to our banking application. The database will be stored server-side at the same physical location, and the server will use standard ODBC drivers to connect from the program to the database to send queries and retrieve the results.

Our current design calls for three main tables – a table of users, a table of accounts, and a table of transactions.

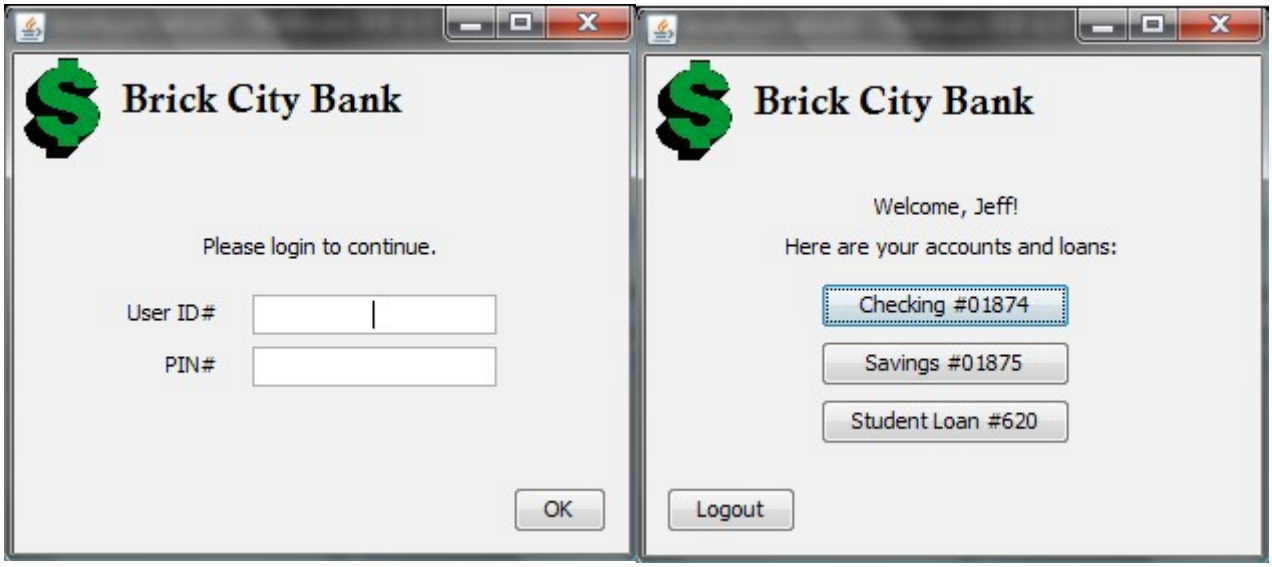
The users table will contain the following columns:

- id – integer, starts from 0 and counts up, used as the primary key
- firstname – 1{character}45, first name of the user
- lastname – 1{character}45, last name of the user
- pin – 6{[0-9] }6, used as an identification code for ATM transactions
- password – 1{character}45, used as an identification code for web banking transactions
- address – 1{character}120, street address of the user
- city – 1{character}45, city the user lives in
- state – 2{[A-Z]}2, 2-letter state code for the user
- cardnumber – 12{[0-9]}12, the ATM card number of the user
- The accounts table will contain the following columns:
  account_number – 12{[0-9]}12, the account number of the user, used as the primary key
- balance – double, the current balance of the account
- It has three subtables, checking, savings, and loans, used for checking accounts, savings accounts, and loans respectively.
- The savings table has one additional column:
- interest – double, the current interest rate (nominal APR) of the account
- The loan table has five additional columns:
- type – 1{character}45, lists the type of loan
- balance – double, the current debt remaining on the loan
- term – date, when the loan is due
- interest – double, the current interest rate (nominal APR) of the loan
- credit_line – double, the current credit line remaining in the loan for some loan types
- The many-to-many relationship between accounts and users is mediated by a user_accounts table, which takes id from users and account_number from accounts as foreign keys and associates each pair in a row, allowing for one account to have multiple users (in the case of married couples, etc.) or one user to have multiple accounts (for example, checking and savings).
- The transactions table contains a history of all transactions which have taken place within the bank's system, and will contain the following columns:
- idTransactions – integer, starts from 0 and counts up, used as the primary key
- type – 1{character}45, lists the type of transaction
- amount – double, the amount of money involved in the transaction
- accountOne – 12{[0-9]}12, foreign key from accounts, the first account involved in the transaction
- accountTwo – 12{[0-9]}12, foreign key from accounts, the second account involved in some kinds of transactions. Left null in others.
- date – date, the date on which the transaction occurred
- time – time, the time at which the transaction occurred

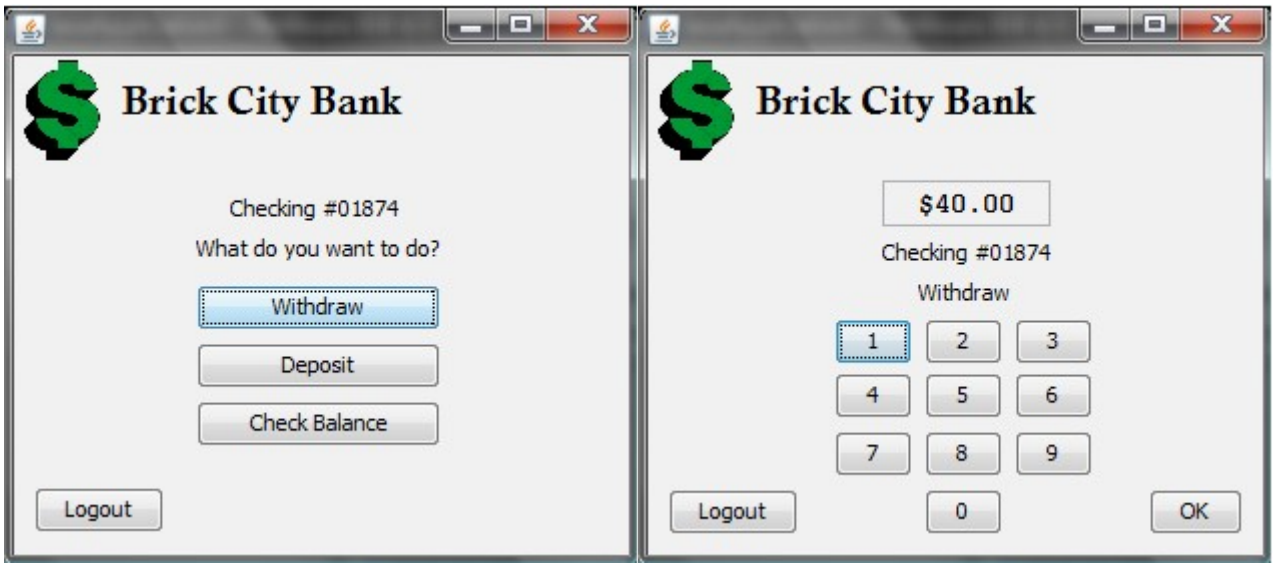
User Interface

ATM

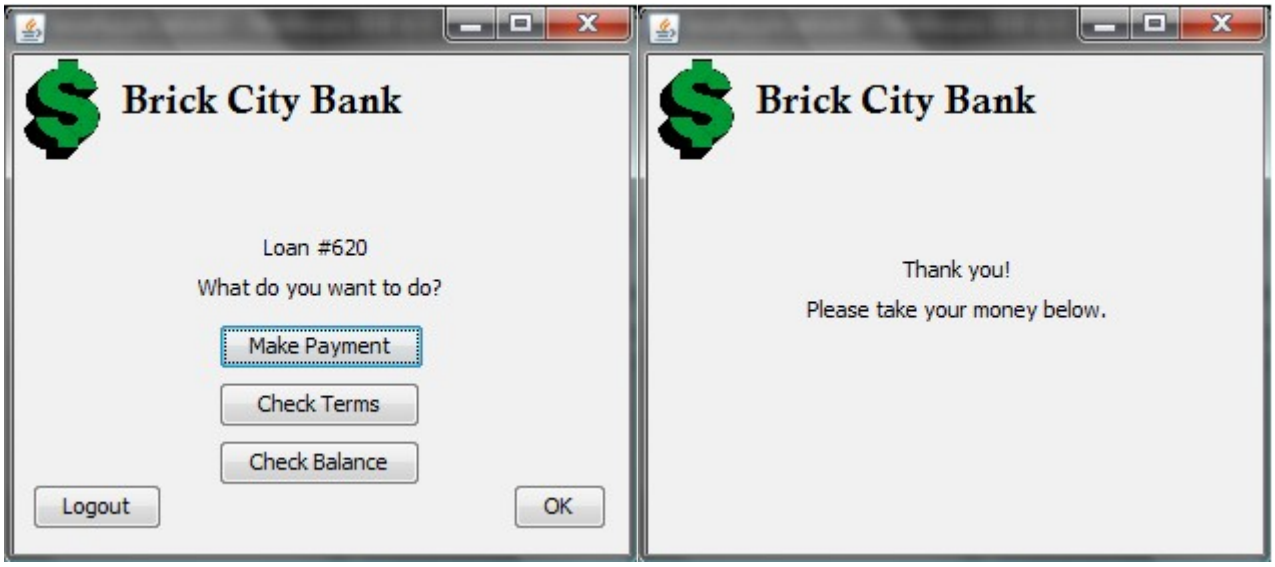
Our ATM interface is currently being designed. Below we have several mock-ups of the purposed screen layouts.



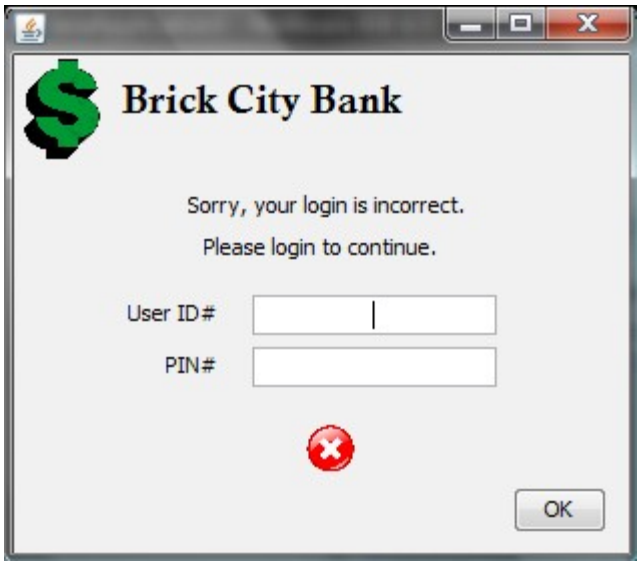
The Log-in Screen and Welcome Screens



A sample checking account and withdrawal Screen



The Loan and Exit screens



The incorrect log-in screen

Web Interface

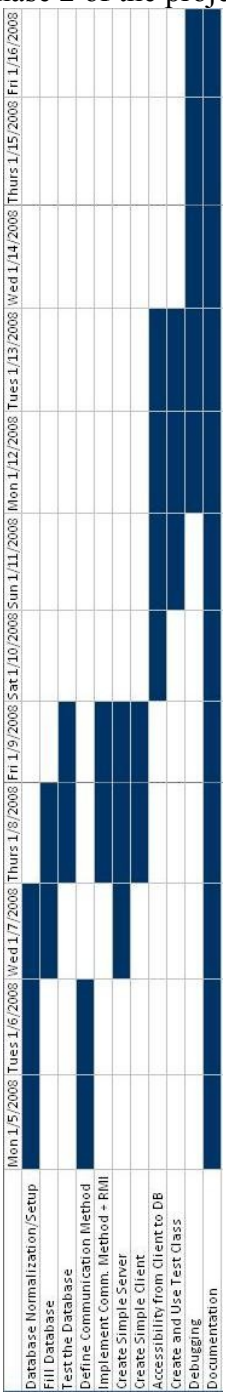
Currently the web interface for our system has yet to be designed. This section will be updated as needed throughout the design and implementation phases.

Team Assignments

Jeff Mueller - User Interface design and Implementation
Caroline Malnoe – Connectivity Implementation
Louis Duke – Documentation, Quality Assurance and core program implmentation
William Manville – Data layer and DBMS Implementation
Ira Mertes – Database Design and Implementation

Timeline

Below is our proposed timeline for phase 2 of the project.



Revision History

Version 1.0 – Initial design document created