

ATM PROJECT

CSC 340 Ethics and Software Engineering



April 14, 2021

Josiah luikham

Individual Project

Table of Contents

[I. Introduction 4](#_Toc70453524)

[A. Problem Statement 4](#_Toc70453525)

[B. System Proposal 4](#_Toc70453526)

[II. System Description 4](#_Toc70453527)

[III. System Requirements 5](#_Toc70453528)

[A. Functional Requirements 5](#_Toc70453529)

[B. Non-Functional Requirements 19](#_Toc70453530)

[IV. Use Case Diagram 20](#_Toc70453531)

[V. Class Diagram 21](#_Toc70453532)

[VI. Sequence Diagrams 22](#_Toc70453533)

[A. Check Balance 22](#_Toc70453534)

[B. Deposit Money 23](#_Toc70453535)

[C. Sign In 24](#_Toc70453536)

[D. Sign Out 25](#_Toc70453537)

[E. Transfer Money 26](#_Toc70453538)

[F. Withdraw Money 28](#_Toc70453539)

[VII. Activity Diagrams 30](#_Toc70453540)

[A. Check Balance 30](#_Toc70453541)

[B. Deposit Money 31](#_Toc70453542)

[C. Sign In 32](#_Toc70453543)

[D. Sign Out 33](#_Toc70453544)

[E. Transfer Money 34](#_Toc70453545)

[F. Withdraw Money 35](#_Toc70453546)

[VIII. State Diagrams 37](#_Toc70453547)

[A. Check Balance 37](#_Toc70453548)

[B. Deposit Money 38](#_Toc70453549)

[C. Sign In 39](#_Toc70453550)

[D. Sign Out 40](#_Toc70453551)

[E. Transfer Money 41](#_Toc70453552)

[F. Withdraw Money 42](#_Toc70453553)

[IX. Database Design 44](#_Toc70453554)

[A. ER Diagram 44](#_Toc70453555)

[B. Table Schema 45](#_Toc70453556)

[X. Conclusion 46](#_Toc70453557)

[XI. Data Dictionary 46](#_Toc70453558)

Table of Figures

[Figure 1: Main Menu 5](#_Toc70453559)

[Figure 2: Account List for Withdrawal 6](#_Toc70453560)

[Figure 3: Error Already Reached Limit 7](#_Toc70453561)

[Figure 4: Display Account Balance 7](#_Toc70453562)

[Figure 5: Withdrawal Menu 8](#_Toc70453563)

[Figure 6: Error, limit exceeded 8](#_Toc70453564)

[Figure 7: Error, Balance Exceeded 9](#_Toc70453565)

[Figure 8: Error, not enough cash 10](#_Toc70453566)

[Figure 9: Withdrawal Display Message 10](#_Toc70453567)

[Figure 10: Account List for Balance Checking 11](#_Toc70453568)

[Figure 11: Account List for Deposit 12](#_Toc70453569)

[Figure 12: Deposit Menu 13](#_Toc70453570)

[Figure 13: Deposit Display Message 13](#_Toc70453571)

[Figure 14: Login Menu 14](#_Toc70453572)

[Figure 15: Error, Only One Account 15](#_Toc70453573)

[Figure 16: Accountlist for Transferring From 16](#_Toc70453574)

[Figure 17: Accountlist for Transferring To 17](#_Toc70453575)

[Figure 18: Display Balances for Transfer 17](#_Toc70453576)

[Figure 19: Transfer Money Menu 18](#_Toc70453577)

[Figure 20: Transfer Money Display Message 19](#_Toc70453578)

[Figure 21: Use Case Diagram 20](#_Toc70453579)

[Figure 22: Class Diagram 21](#_Toc70453580)

[Figure 23: Check Balance Sequence Diagram 22](#_Toc70453581)

[Figure 24: Deposit Money Sequence Diagram 23](#_Toc70453582)

[Figure 25: Sign In Sequence Diagram 24](#_Toc70453583)

[Figure 26: Sign Out Sequence Diagram 25](#_Toc70453584)

[Figure 27: Transfer Money Sequence Diagram 26](#_Toc70453585)

[Figure 28: Withdraw Money Sequence diagram 28](#_Toc70453586)

[Figure 29: Check Balance Activity Diagram 30](#_Toc70453587)

[Figure 30: Deposit Money Activity Diagram 31](#_Toc70453588)

[Figure 31: Sign In Activity Diagram 32](#_Toc70453589)

[Figure 32: Sign Out Activity Diagram 33](#_Toc70453590)

[Figure 33: Transfer Money Activity Diagram 34](#_Toc70453591)

[Figure 34: Withdraw Money Activity Diagram 35](#_Toc70453592)

[Figure 35: Check Balance State Diagram 37](#_Toc70453593)

[Figure 36: Deposit Money State Diagram 38](#_Toc70453594)

[Figure 37: Sign In State Diagram 39](#_Toc70453595)

[Figure 38: Sign Out State Diagram 40](#_Toc70453596)

[Figure 39: Transfer Money State Diagram 41](#_Toc70453597)

[Figure 40: Withdraw Money State Diagram 42](#_Toc70453598)

[Figure 41: Entity Relationship Diagram 44](#_Toc70453599)

# Introduction

## Problem Statement

ZZZ Bank plans to provide ATM service to their customers. Currently they don’t have any ATM machines. They need a software system to provide withdrawal, deposit, balance checking, and transferring money services to their customers.

## System Proposal

We propose a software system to help ZZZ Bank with the desired functionality of the system.

# System Description

A new local bank ZZZ asks for your help to develop a software system for their multiple ATM machines. The machines will provide basic operations to their customers, including depositing money, withdrawing money, checking balances, and transferring money from one account to another. Each machine will be refilled to hold $100,000 cash daily for possible withdrawals. For security reason, each account can have at most $3000 in total for all the transactions (except for checking balance) through ATM machines each day. Effectiveness and efficiency are their primary requirements.

# System Requirements

## Functional Requirements

1. The system shall allow a customer to withdraw money from his/her banking accounts.
   1. The system shall display a user menu to the customer.

Table

Description automatically generated

Figure 1: Main Menu

* 1. The customer shall select “Withdraw Money” in the menu.
  2. At all input points the customer shall be able to cancel the operation and return to the user menu
  3. The system shall display a list of accounts of the customer to be selected for withdrawal.

Graphical user interface, text

Description automatically generated with medium confidence

Figure 2: Account List for Withdrawal

* + 1. The system shall retrieve the account information from the central database system with the customer’s id.
    2. The system shall format and display only account numbers to the customer.
  1. The customer shall select one account from the list for withdrawal.
  2. The system shall update the date of the daily transactions of the selected account.
     1. If the last date of the transactions is not today’s date, the system shall change the date of the transactions to today’s date and reset the daily transaction total to $0.
  3. The system shall check if the account’s daily transaction total have reached $3000 (see NR1).
     1. If the total has reached $3000 already, the system shall display an error message to the customer, and check if there is any other account to be selected for withdrawal.

Chart

Description automatically generated with medium confidence

Figure 3: Error Already Reached Limit

* + - 1. If there are more accounts to be selected, return Step 1.4.
      2. If there is not any other account to be selected, return to Step 1.1.
    1. If they haven’t reached $3000, the system shall continue Step 1.8.
  1. The System shall display the balance of the account to the customer

Chart

Description automatically generated

Figure 4: Display Account Balance

* 1. The customer shall indicate they are finished viewing
  2. The system shall request the customer to enter the amount to withdrawal.

Timeline

Description automatically generated with medium confidence

Figure 5: Withdrawal Menu

* 1. The customer shall enter an amount.
  2. The system shall check if the total of the amount plus the daily transaction total is greater than $3000 (see NR1).
     1. If the total is greater than $3000, the system shall display an error message to the customer, and return to Step 1.10.

Application

Description automatically generated with low confidence

Figure 6: Error, limit exceeded

* + 1. If the total is less than or equal to $3000, the system shall continue Step 1.13.
  1. The system shall verify if the amount is less than the current balance of the selected account.
     1. If the amount is greater than the balance, the system shall display an error message to the customer, and return to Step 1.10.

Chart

Description automatically generated with medium confidence

Figure 7: Error, Balance Exceeded

* + 1. If the amount is less than or equal to the current balance, the system shall continue Step 1.14.
  1. The system shall check if the machine has enough cash for the withdrawal.
     1. If there is not enough cash for the withdrawal, the system shall display an error message, and return to Step 1.6

A picture containing text

Description automatically generated

Figure 8: Error, not enough cash

* + 1. If there is enough cash for the withdrawal, the system shall continue Step 1.13.
  1. The system shall dispense the cash.
  2. The system shall deduct the amount from the balance of the account.
  3. The system shall display a message showing the amount withdrawn and old and new balance

A picture containing graphical user interface

Description automatically generated

Figure 9: Withdrawal Display Message

* 1. The system shall add the amount to the daily transaction total.
  2. The system shall update the account information in the database.
  3. The system shall store this transaction to the database.
  4. The system shall return to main menu.

1. The System shall allow a customer to check the balance of his/her banking accounts
   1. The system shall display a user menu to the customer.
   2. The customer shall select “check balance” in the menu.
   3. The system shall display a list of accounts of the customer to be selected for balance checking

Graphical user interface, text, application

Description automatically generated

Figure 10: Account List for Balance Checking

* + 1. . The system shall retrieve the account information from the central database system with the customer’s id.
    2. The system shall format and display only account numbers to the customer.
  1. The customer shall select one account from the list for balance checking.
  2. The system shall display the balance of the account on the screen
  3. The system shall allow a customer to indicate they are finished viewing
  4. The customer shall indicate they are finished
  5. The system shall return to the main menu

1. The system shall allow a customer to deposit money in the ATM
   1. The system shall display a user menu to the customer
   2. The customer shall select “deposit money” in the menu.
   3. The system shall display a list of accounts of the customer to be selected for depositing.

Graphical user interface, text

Description automatically generated

Figure 11: Account List for Deposit

* + 1. The system shall retrieve the account information from the central database system with the customer’s id.
    2. The system shall format and display only account numbers to the customer.
  1. The customer shall select one account for deposition from the menu
  2. The system shall display the balance of the account
  3. The customer Shall indicate they are finished viewing
  4. The System shall ask the customer to enter the amount of money to be deposited

Timeline

Description automatically generated

Figure 12: Deposit Menu

* 1. The customer shall enter the amount of money to be deposited
  2. The system shall add the amount to the balance of the account
  3. The system shall display a message showing old and new balance and the amount deposited

Table

Description automatically generated with low confidence

Figure 13: Deposit Display Message

* 1. The system shall update the account information in the database
  2. The system shall create a record of this transaction
  3. The system shall store this record in the database
  4. The system shall return to the main menu

1. The system shall allow a user to sign in to the atm
   1. The System shall ask the user to enter an ID and password

A picture containing chart

Description automatically generated

Figure 14: Login Menu

* + - 1. The system shall check the database to determine if the ID and password match an existing user
      2. If they do not match an error message will be displayed and the system will return to step 4.1
      3. If they match then continue step 4.1
  1. The user will be granted customer status
  2. a user menu will be displayed

1. The system shall allow a user to sign out of the atm
   1. The system shall display a user menu to the customer
   2. The customer shall select “sign out” from the menu.
   3. The system shall demote the customer to user status
   4. The system shall ask the user to enter an ID and password
2. The system shall allow a customer to transfer money from one account to another
   1. The system shall display a user menu to the customer.
   2. The customer shall select “transfer money” in the menu.
      1. The system shall retrieve the account information from the database
      2. The system shall check to make sure there is more than one account
         1. If there is only one account the system shall display an error message and return to step. 6.1

A picture containing chart

Description automatically generated

Figure 15: Error, Only One Account

* + - 1. If more than one account the system shall continue to 6.3
  1. The system shall display a list of accounts of the customer to be selected for transferring

From

Graphical user interface, text, application

Description automatically generated

Figure 16: Accountlist for Transferring From

* + 1. The system shall format and display only account numbers to the customer.
    2. The user shall select an account from the list for transferring money from
  1. The system shall update the date of the daily transactions of the selected transferring from account.
     1. If the last date of the transactions is not today’s date, the system shall change the date of the transactions to today’s date and reset the daily transaction total to $0.
  2. The system shall check if the account’s daily transaction total have reached $3000 (see NR1).
     1. If the total has reached $3000 already, the system shall display an error message to the customer and return to step 6.3
     2. If they haven’t reached $3000, the system shall continue Step 6.6
  3. The system shall display a list of accounts for transferring money to.

Graphical user interface, text, application, email

Description automatically generated

Figure 17: Accountlist for Transferring To

* + 1. The system shall format and display only account numbers to the customer.
    2. The user shall select an account from the list for transferring money to
  1. The System shall display the balances of the transferring to and from accounts

Text

Description automatically generated with low confidence

Figure 18: Display Balances for Transfer

* 1. The customer shall indicate they are finished viewing
  2. The system shall request the customer to enter the amount to withdrawal.

Timeline

Description automatically generated with low confidence

Figure 19: Transfer Money Menu

* 1. The customer shall enter an amount.
  2. The system shall check if the total of the amount plus the daily transaction total is greater than $3000 (see NR1).
     1. If the total is greater than $3000, the system shall display an error message to the customer, and return to Step 6.9
     2. If the total is less than or equal to $3000, the system shall continue Step 6.12.
  3. The system shall verify if the amount is less than the current balance of the selected transferring from account.
     1. If the amount is greater than the balance, the system shall display an error message to the customer, and return to Step 6.9.
     2. If the amount is less than or equal to the current balance, the system shall continue Step 6.13.
  4. The system shall deduct the amount from the balance of the transferring from account
  5. The system shall add the amount to the daily transaction total for the transferring from account
  6. The system shall add the amount to the balance of the transferring to account
  7. The system shall display the old and new balances to the customer and the amount transferred

Table

Description automatically generated

Figure 20: Transfer Money Display Message

* 1. The system shall update the account information in the database
  2. The system shall store this withdrawal transaction to the database.
  3. The system shall update the account information in the database
  4. The system shall store this deposit transaction to the database.
  5. The system shall return to main menu.

## Non-Functional Requirements

Non-functional Requirements:

1. The total of daily transactions (withdrawals ad money transfers) shall not exceed $3000 from same account.
2. The cash in a machine will be reset to $100000 every day. This action will be done manually by the employees of the bank (The software system is not responsible for this part.)
3. The amount of a deposit will not be added to the balance of an account immediately. The system will show the deposit amount in another way. The action for adding the amount to the balance will be done manually by the employees of the bank (The software system is not responsible for this part.)

# Use Case Diagram



Figure 21: Use Case Diagram

This Diagram is to show the functionality of the system using use cases. A use case is a summary of related scenarios that may take place. For instance, anything that happens when a customer is withdrawing money falls under the Withdraw Money use case.

The User Actor is the anyone who may interact with the System. The only scenario they take part in is signing in. Once a user signs in they are a Customer Actor and participate in the other actions.

# Class Diagram

This diagram shows the data stored within classes as well as the functions that each class will perform. The lines between classes represent relationships between classes. The numbers on the sides of the classes represents how many of this class will be related to how many of the other. “1…\*” means any number from 1 to indefinite.



Figure 22: Class Diagram

# Sequence Diagrams

These Diagrams Show the sequence of functions that will occur in each use case. The pillars represent the different classes that the functions will occur in as well as actors that will perform the initial call. Functions are represented by the arrows with names. A function is implemented in the class that the arrow is pointing toward. A function is called by the class or user that the arrow is coming from. The lower down the function is, the later in the sequence it occurs. A White box surrounding a function or functions means that the functions inside and below it only occur if the condition inside the box is met.

## Check Balance



Figure 23: Check Balance Sequence Diagram

In this Sequence diagram, the only function that is called is getBalance(). The rest is handled by the GUI.

## Deposit Money



Figure 24: Deposit Money Sequence Diagram

Here we see that a depositMoney() Function is called. Next, UpdateBalance(), updateAccountData(), create(), and saveTransaction() are called. UpdateBalance() is self explanatory. UpdateAccountData() will update this account’s information in the database. Create() is creating a new instance of a transaction object, and saveTransaction will save the new object to the database.

## Sign In



Figure 25: Sign In Sequence Diagram

The function checkCredentials() will verify that the user id and password inputs match a customer in the database. If this function returns a value of true, then a new customer object containing the customer’s information will be created. RetrieveAccounts() will retrieve the information for each of the customer’s accounts and create new Account objects to store them in.

## Sign Out



Figure 26: Sign Out Sequence Diagram

When the Sign out use case occurs, the objects that Exist in the program are destroyed. The Customer actor presses sign out, which will call he destroy functions. There are two functions for account destruction. DestroyAccs() destroys the list of accounts stored. DestroyAcc() destroys the account stored in the variable for whichever account was used last.

## Transfer Money



Figure 27: Transfer Money Sequence Diagram

The system first makes a call to make sure there is more than one account available to the customer. If this is true, WithdrawMoney() is called, an from within this function, updateTodaysDate() is called which checks that the current date stored in the system is accurate. checkDailyTransactions() verifies that the account which will be transferred from has not already reached its withdrawal limit for the day. VerifyDailyTransaction() ensures that the amount of money to be transferred will not exceed the limit. updateBalance() is self explanatory. updateDailyTransactionTotal() updates the daily withdrawal total on record for the transferring from account. UpdateAccountData() saves the new data for the accounts to the database. From here the program call for withdrawMoney() finishes, and the gui interfaced with by the Customer calls for a transaction object to be created and stored in the database.

## Withdraw Money



Figure 28: Withdraw Money Sequence diagram

The initial function called is withdrawMoney() this function will call the remaining functions within the account class. Many of the functions here are also present in the transfer money diagram. UpdateTodaysDate is called, followed by checkDailyTransactions(). The functions within the condition box execute only if the condition is met. If so, verifyDailyTransaction() is called. Another condition box is met. Then verifyAccountBalance() is called. Yet another condition box is met, and checkMachineCash() is called. Finally, after the last condition box is met, updateBalance(), updateDailyTransactionTotal() and updateAccountData() are called. A new transaction object is created and saveTransaction() is called.

# Activity Diagrams

These Diagrams show the structure of the functions that will implement the various use cases. The single circle connected to the line represents the beginning. The circle within a circle represents the end. Each function is mapped out between horizontal lines with a comment box telling its name. The white boxes tell what is happening inside the code. Diamond symbols show decision branches in the code.

## Check Balance



Figure 29: Check Balance Activity Diagram

The function getBalance() is very simple. It consists of simply retrieving the value for balance.

## Deposit Money



Figure 30: Deposit Money Activity Diagram

DepositMoney() is a simply a function which calls other functions. UpdateBalance() adds the variable “amount” to the variable “balance”. UpdateAccountData() accesses the database using sql to update the accounts stored there. Create() makes a new transaction object. SaveTransaction() also accesses the database using an sql statement, storing a new transaction there.

## Sign In



Figure 31: Sign In Activity Diagram

The checkCredentials() function first checks the database using an sql statement to determine whether there exists a customer with the credentials input by the user. If there is not, the system displays an error message. If there is, the system continues. A customer object is created with the data input by the user. The retrieveAccounts() function uses an sql statement to retrieve the data for all accounts in the database with the current customer ID. It then creates objects for each account and stores the objects in an arraylist.

## Sign Out



Figure 32: Sign Out Activity Diagram

This diagram shows the architecture for the functions in the sign out system. First, the customer object stored is set to null, then the account objects, and finally, the last transaction object created is set to null.

## Transfer Money



Figure 33: Transfer Money Activity Diagram

First, the checkIfMultAccounts() function checks if multiple accounts, if true it continues, if false, an error message is displayed. UpdateTodaysDate() checks if the day on file is today, if true it continues, if false it updates and sets the transaction total for the transferring from account to zero. CheckDailyTransactions() checks that the transaction total is not over the limit, if so it displays an error, if not it continues. VerifyDailyTransactions() checks if the amount input for transfer will put the total transaction amount over the limit for the day. If so, an error is displayed. If not, continue. VerifyAccountBalance() checks that the transferring from account has enough money to complete the transfer. If it does, continue. If not, an error is displayed. In updateBalance(), the balance of the transferring from account has the amount subtracted and the balance of the transferring to account has the amount added. UpdateDailyTransactionTotal() updates the transaction total for the day for the transferring from account. Update account data updates the info for both accounts in the database using an sql statement. A new transaction object is created and saveTransaction() stores the data for it in the database using an sql statement.

## Withdraw Money



Figure 34: Withdraw Money Activity Diagram

WithdrawMoney() is simply a function which calls other functions. UpdateTodaysDate(), checkDailyTransactions(), verifyDailyTransactions(), and verifyAccountBalance() functions all behave as described in the transfer money activity diagram. checkMachineCash() checks if there is enough money in the machine for the withdrawal. If not, an error is displayed. If there is, the system continues. The functions of updateBalance(), updateDailyTransactionTotal(), updateAccountData(), and saveTransaction() all occur just as explained in the transfer money diagram above.

# State Diagrams

These diagrams show the states that the system will be in during different parts of each use case. They are used when testing the System. Each square represents a state. The lines represent transitions to new states.

## Check Balance



Figure 35: Check Balance State Diagram

We can see here that in the check balance use case, the program is initially in a state of displaying the menu. Clicking check balance leads to the displaying account list state. When an account is selected, the program moves to the displaying balance state. Clicking ok leads the program back to the displaying menu state.

## Deposit Money



Figure 36: Deposit Money State Diagram

Initially the program is displaying menu. Clicking the deposit button causes the program to transition to displaying account list. Select account transitions the program to displaying balance. Clicking ok transitions the program to a window where the customer is asked to input an amount. Entering an amount leads the program to be in a state of updating the balance. The program transitions from here to displaying the old and new balance for the account. Clicking ok causes a transition to a state of updating account info in the database, this transitions to storing a record of the transaction, and finally, the program transitions to displaying the menu again.

## Sign In



Figure 37: Sign In State Diagram

Initially, the program is in a state of asking for credentials. Entering credentials causes the program to enter a state of verifying credentials. The program then transitions to retrieving account info, and from there transitions to displaying the menu.

## Sign Out



Figure 38: Sign Out State Diagram

Initially the program is in a state of displaying the menu. Selecting sign out causes the program to transition to a state of setting variables stored within it to default values. When this finishes, the program transitions to a state of asking the user to input credentials.

## Transfer Money



Figure 39: Transfer Money State Diagram

Initially the program is in a state of displaying the menu. Clicking transfer money causes a transition to checking if there is only one account for the current customer. If there is, the program returns to menu. If not, the program transitions to a state of displaying accounts for transferring from. Selecting one of these transitions the program to displaying accounts for transfer to. Selecting one of these transitions the program to updating the date. From here the program transitions to checking if the daily transaction total has already gone over 3k. If it has, the program returns to a state of displaying accounts for transferring from. If not, it transitions to a state of displaying the balance of the accounts. Clicking ok causes a transition to displaying an interface for entering an amount to transfer. Entering an amount causes the program to check if there is enough money in the balance of the relevant account. If there is not, the program transitions back to displaying an interface for entering an amount to transfer. If there is, the program enters a state of updating balance for the transferring from account, then a state of updating account info for the transferring from account. The program then enters the same states consecutively for the transferring to account. Afterward, the program enters a state of storing the transaction to the database. Finally, the program transitions back to displaying the menu.

## Withdraw Money



Figure 40: Withdraw Money State Diagram

All states in the withdraw money state diagram and how they work are explained in the transfer money Activity diagram description above, except for checking if machine has enough. The system will enter this state after checking if the amount is over the balance. If the machine does not have enough cash, the state transitions back to displaying withdrawal interface. If it does, the system continues. The State Diagram moves along but with only one account instead of two like in the transfer money diagram. Also, it does not incorporate some of the functions associated with the transfer money system.

# Database Design

## ER Diagram

This Diagram is a model of the database structure that will be used in the program. Rectangles represent tables with data related to their labels, ovals are the data types that will be present in the tables. Diamonds are relationships between tables. The “1” and “M” represent 1 and many. It signifies that the table with the 1 will be related to many instances of the tables with the m, and vice versa.



Figure 41: Entity Relationship Diagram

In this ER Diagram, we can see that the Customer table will contain the variables of id and password. The Account table will contain variables for dailyTransactionDate, dailyTransactionLimit, dailyTransactionTotal, customerID, balance, and accountID. The Transaction table will have variables for transactionType, date, toAccount, fromAccount, transactionID, amount, and accountID. Customer has a one to many relationship with Account. Account Has a one to many relationship with Transaction.

## Table Schema

The table schema is a representation of the data stored within the tables and how one tables data will relate to another.

|  |  |
| --- | --- |
| ID | password |

Customer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| dailyTransactionDate | accountID | balance | dailyTransactionLimit | customerID | dailyTransactionTotal |

Account

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| date | amount | accountID | transactionType | transactionID | fromAccount | toAccount |

Transaction

In this table schema we can see that the ID value in customer maps onto the customerID value in Account. The accountID variable in Account maps onto the accountID variable in transaction, as well as the fromAccount and toAccount variables.

# Conclusion

In this project, a simulated ATM System was created. It fulfilled the functional requirements of providing a Sign In System, a Sign Out System, a Withdraw Money System, a Transfer Money System, and a Deposit Money System. The system fulfilled the nonfunctional requirement that no more than 3000$ shall be withdrawn from an account per day. The second nonfunctional requirement is not related to software and irrelevant. During Implementation, I chose not to follow the last non-functional requirement. Whenever money is deposited, the software system updates the account with the new balance.

The project began with creation of the functional requirements. These allowed the creation of many diagrams which provided a map of how the system would be implemented. Using the diagrams, the system was implemented using visual studio and c#. In some cases, the implementation did not match the diagrams, because a better method was found in the process. The diagrams were updated accordingly. The database used for the project was made using HeidiSQL. The ATM program accesses the databases using SQL queries within the c# code.

# Data Dictionary

Functional Requirements: A requirement specifying something that the software must do.

Non-Functional Requirement: A requirement specifying something that the software must not violate.

Use Case Diagram: A diagram describing what a system does from the standpoint of an external observer. The emphasis is on *what* a system does rather than *how.*

Use Case: A summary of scenarios that may take place during a task.

Class Diagram: A Class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static -- they display what interacts but not what happens when they do interact.

Sequence Diagram: An interaction diagram that details how operations are carried out -- what messages are sent and when. Sequence diagrams are organized according to time. The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

Activity Diagram: A flowchart for a particular use case, which describes the way the code will work.

State Diagram: A diagram showing the various states the program will be in while it is running. It is typically used in testing.

Object: an instance of a class.

ArrayList: a list of objects which are stored together for convenience.