

**Oscar Villasana**

**Benjamin wodhams**

**ryan wolfe**

Automated Teller Machine (ATM)

# Problem Statement

ATM’s are very important to today’s way of living because they allow people to retrieve money they have in their bank account without the need of a teller. This greatly reduces the amount of time it takes to get to get money. ATM’s are located all over modern cities making it the fastest way to get money without the need of going to the bank.

Our project was simulating an Automated Teller Machine (ATM). The ATM will include accounts of users, a feature to create accounts, and the amount of money it holds. Each User has at least one bank account, a name, username, password, address, age, birthday and a user ID. There were 2 types of accounts: Checking account and savings account. We decided to go with these two types of bank accounts because they are the most common. Each account has a balance and a pin number for security. Each account will have its unique features. An account will have an account name, withdraw amount and a deposit amount. These features depend on the type of account. For example, a checking account will have a maximum daily withdraw amount of 300 dollars where a Business account will have no limit. Savings and credit accounts are typically monitored so you don’t withdraw cash and we will take that into account. Some users will also be considered VIP which allow them to withdraw and deposit a larger amount of money than a non-VIP member.

The users are able to login to their account or create and account if they are a new user. Once they have done this, they can see their current accounts and they can deposit money or withdraw money. If a new user doesn’t have any accounts, a new GUI window will popup where they will enter their information, and they will then be able to login. We will introduce serialization in order to keep each user’s information saved even if the application is closed.

# System boundary

The ATM simulation has features for everyone. If there is a new user, the new user will enter their information such as: name, username, password, address, age, phone number and birthday. The ATM will check to see if another user does not have the same username. We excluded password strength as a real ATM/Bank would require you to do as it was a much more difficult thing to do. Once a valid username is chosen, a new account has been created and the new user is ready to login. Once a user logs into their account, they will see their information as well as their bank accounts with their balances. The user will be able to withdraw of deposit money from their account once they enter the correct account pin. We added a security feature that when the wrong pin is entered 4 times, the system shuts down to prevent unauthorized users to access the money. Our main focus was security and that’s why we required two layers of protection. The first one was logging in with a username and password. Once they were in, the second security feature asked for the account pin to handle the money. We excluded issues of the ATM being down. We also excluded the issue of only being able to withdraw factors of 20 just to make the program more interactable. In a real ATM, you may only withdraw an amount that is divisible by 20 as the ATM only has 20 dollar bills. We also decided to remove the transfer money to a different user as it would require the user sender to know the user receiver’s information. We also decided to create 3 separate GUI in order to maintain the program running soother so there wouldn’t be many popups for all the different functions. This makes the program run smoother.

|  |
| --- |
| USER |
| - name: String  - accountList: ArrayList < Account >  - age: int  - birthday: String  - address: String  - VIP: boolean  - UserID: int  - username: String  - password: String |
| + getAccountList(): ArrayList<Account>  + setName(n: String): void  + getName( ): String  + setAge(a: String): void  + getAge( ): String  + setBirthday(b: String): void  + getBirthday( ): String  + setAddress(a: String): void  + getAddress( ): String  + setVIP(b: Boolean): void  + getVIP( ): boolean |

# Domain analysis (UML)

|  |
| --- |
| ATM |
| - userList: ArrayList<User>  - moneyAvailable: double  - moneyWithdrawn: double |
| + getUserList(): ArrayList<User>  + setMoneyAvailable(money: double): void  + getMoneyAvailable( ): double  + setMoneyWithdrawn(money: double): void  + getMoneyWithdrawn( ): void  + createAccount( ): void |

|  |
| --- |
| ACCOUNT |
| - accountNumber: double  - balance: double  - pin: int |
| + getAccountNumber( ): void  + setAccountNumber(n: double): void  + getBalance( ): void  + setBalance(n: double): void  + transferMoney(a: double): void |

|  |
| --- |
| SAVINGS |
| - maxDeposit: double  - minWithdraw: double  - minDailyWithdraw: double |
| + getMaxDeposit(): double  + setMaxDeposit(n: double): void  + getMinWithdraw(): double  +setMinWithdraw(n:double): void  + getMaxDailyWithdraw(): double  + setMaxDailyWithdraw(n:double):void |

|  |
| --- |
| CHECKING |
| - maxDeposit: double  - minWithdraw: double  -maxDailyWithdraw: double |
| + getMaxDeposit(): double  + setMaxDeposit(n: double): void  + getMinWithdraw(): double  +setMinWithdraw(n:double): void  + getMaxDailyWithdraw(): double  + setMaxDailyWithdraw(n:double):void |

# State model

The class with the biggest state change was the GUI for the ATM. We decided it would be better to create 3 separate GUIs. One for the user login, user register and finally the main ATM GUI. By doing this, we can make the program run smoother.

User logs out

User Login

User enters correct username/password

User Information

User selects Account

Withdraw or Deposit

Correct PIN

Display Successful Transaction

User Information Input

User selects Register

Since our program demanded many functions to operate, it would have been somewhat uncomfortable for the user to interact with multiple pop up frames popping out for everything the user did. We created a GUI so the user can login and when they do sign in, there is a separate GUI that handles all the transactions the user desires.

# Interaction Analysis

CHANGE/SET

PIN

DEPOSIT

MONEY

WITHDRAW

MONEY

USER

LOGIN

ATM

GUI

LOGIN TO

EXISTING

ACCOUNT

CREATE ACCOUNT

VIEW AND SELECT ACCOUNT

# Implementation

# testing & results

# Extensibility