**AUTOMATED TELLER MACHINE (ATM) SIMULATION USING JAVA**

**TABLE OF CONTENTS**

[Introduction 3](#_Toc139715947)

[Discussion 3](#_Toc139715948)

[Conclusion 12](#_Toc139715949)

[Reference list 13](#_Toc139715950)

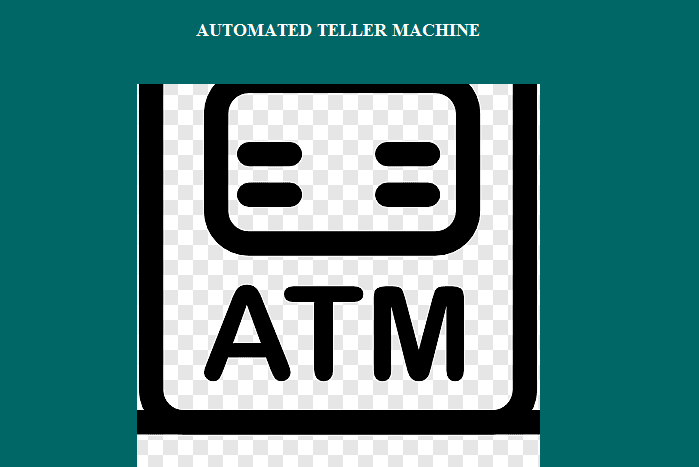
[Appendices 14](#_Toc139715951)

# Introduction

A piece of software called an Automated Teller Machine (ATM) emulator uses the programming language known as Java to simulate the abilities and processes of an ATM. Users can simulate accessing their financial institution accounts, carrying out operations, and managing their money in a virtual setting. Considering the ability to provide quick and safe entry to banking services, ATMs have become an integral component of our everyday life. Developers can develop a virtual version of this innovation by simulating an ATM using Java, so it allows buyers to comprehend the specifics of the banking industry while learning how to make use of an ATM.This project's goal is to give students practical experience with and knowledge of the basic functions of the ATM system, such as account identification, balance enquiries, cash withdrawals, deposits, and money transfers. Users may improve their Java programming abilities while learning more about how ATMs work by creating this simulation.

# Discussion

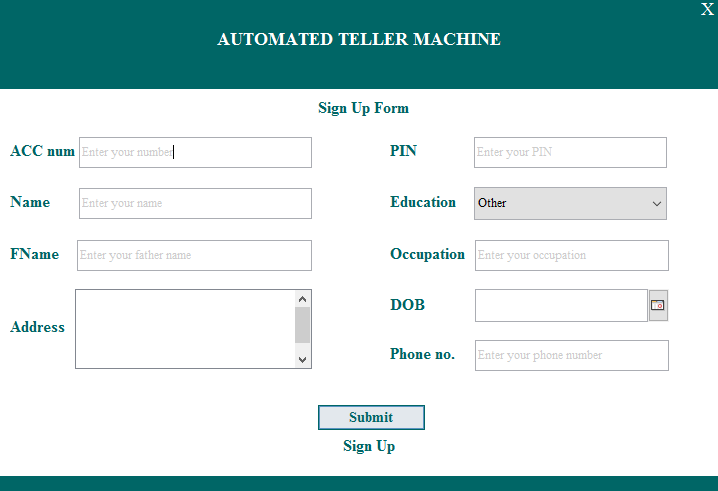
Users must enter their own account credentials, such as a PIN (Personal Identification Number) or card number, into the ATM simulation in order to verify themselves and receive access to their accounts.Users have permission to utilize a variety of managing accounts functions, including being able to monitor transaction histories, check the status of accounts, and update their private information.Individuals who have enough money in their bank accounts can ask to withdraw a particular amount of money (Akinnuwesi*et al.* 2023). The withdrawn sum should be appropriately removed from an account's balance in the simulation.Users can deposit money into their accounts by entering the desired amount. The revised account balance should correspondingly be reflected in the simulation.Users have the option of transferring money among their own accounts or to specific accounts at the same bank. The simulation ought to verify the availability of money and revise the sender's and recipient's account balances.To maintain the confidentiality and integrity of user information, the simulation should employ security features such session timeouts, account lockouts after several failed login attempts, and encryption of important data (Akinola Kayode*et al.* 2019).Users can better grasp the underlying procedures and logic involved in banking systems by creating an ATM simulation in Java. It allows developers to improve their abilities in areas like object-oriented programming, user input validation, and data manipulation. It gives a realistic application of Java programming ideas.

****

**Figure 1: First page of ATM**

(Source: self-created in Netbeans)

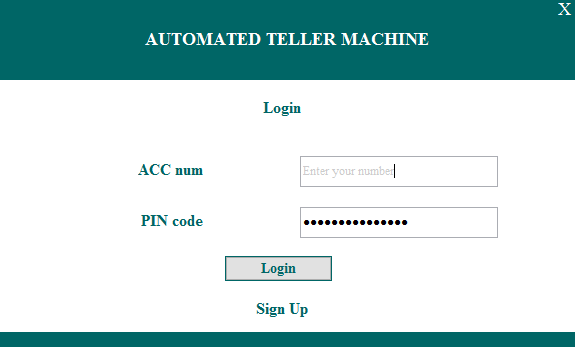
The first page of the Java ATM simulation is seen in this example. The user is given an option of checking in or departing (Oko*et al.* 2022). If the user decides to log in, it runs the login() function to check the password, account information and PIN of the person who is logged in.



**Figure 2: Sign up form**

(Source: self-created in Netbeans)

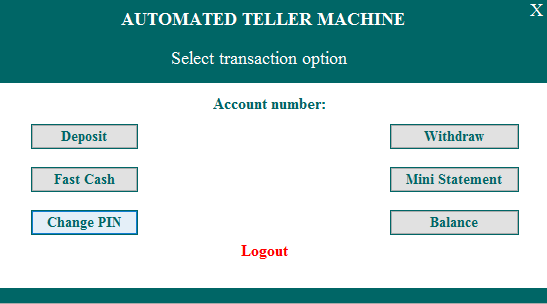
In this illustration, currently enhanced the ATM simulation by including a registration form. The signUp() function occurs once a user picks the "Sign Up" alternative (Sanusi*et al.* 2022). The member must then enter their 4-digit PIN for the account's password as well as their full name, year of birth, mailing address, and cell phone number.



**Figure 3: Login form**

(Source: self-created in Netbeans)

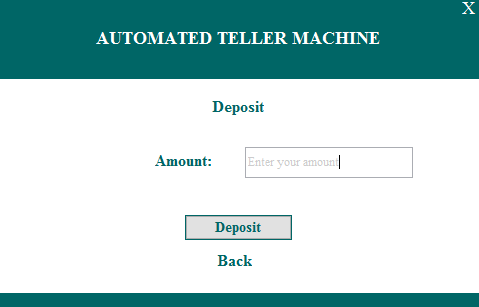
In this illustration, the components of the ATM simulation include a login form. The "Login" option causes the login() function to be invoked (Wang*et al* 2020). The individual is required to provide their PIN and the number of their account. The authorization logic then gets carried out through the use of the authenticate() function.



**Figure 4: Main menu page**

(Source: self-created in Netbeans)

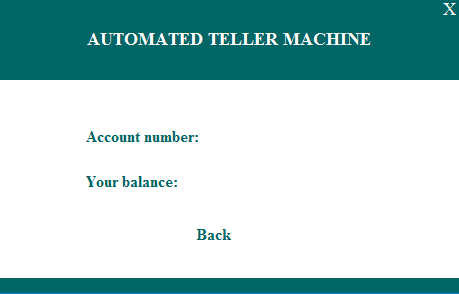
In this case, the showMainMenu() function is called once the user logs in or registers. The primary screen is shown, offering a few options including checking the balance, pulling money out, putting funds in, or leaving the ATM.



**Figure 5: Deposit money page**

(Source: self-created in Netbeans)

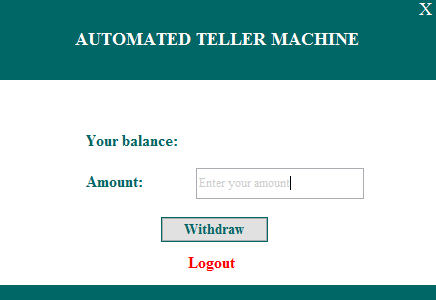
In this example, when the user picks the "Deposit Cash" choice from the default menu, the depositCash() function is executed.



**Figure 6: Checking balance page**

(Source: self-created in Netbeans)

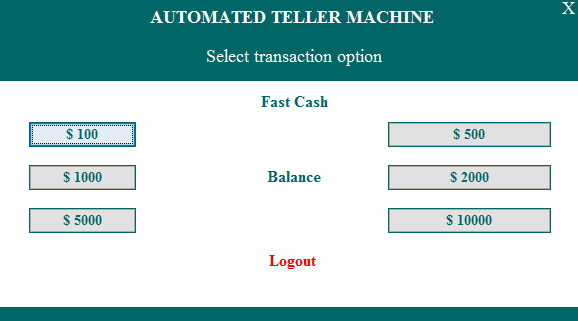
The checkBalance() function is invoked in this example when the user chooses the "Check Balance" option from the main menu.



**Figure 7: Withdrawing money page**

(Source: self-created in Netbeans)

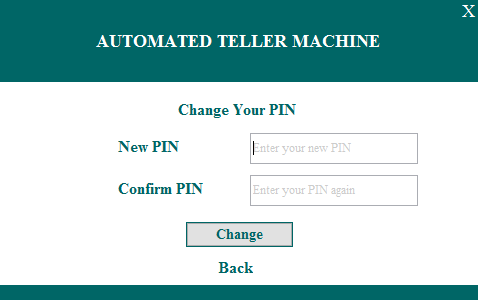
The withdrawCash() function is invoked in this example when the user chooses the "Withdraw Cash" option from the main menu.



**Figure 8: Fast cash page**

(Source: self-created in Netbeans)

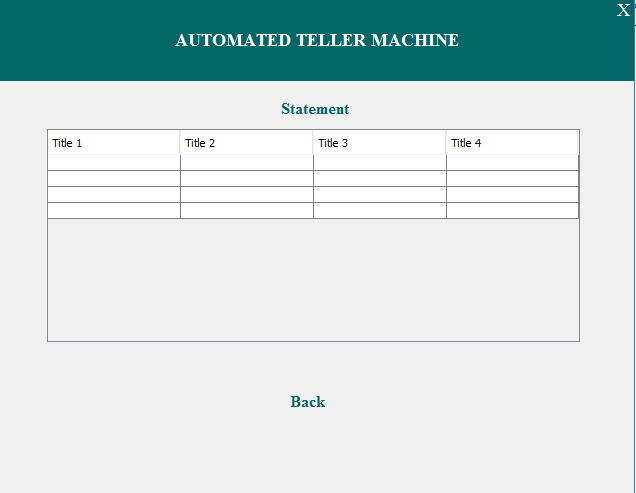
The fastCash() function is invoked in this example when the user chooses the "Fast Cash" option from the main menu.**[*Refer to Appendix 1*]**



**Figure 9: Change PIN page**

(Source: self-created in Netbeans)

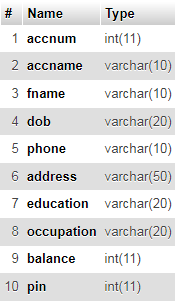
The changePIN() function is invoked in this example when the user chooses the "Change PIN" option from the main menu.

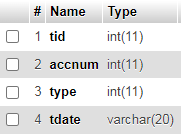


**Figure 10: Mini statement page**

(Source: self-created in Netbeans)

When the user chooses "Mini Statement" from the main menu, the miniStatement() function is invoked.

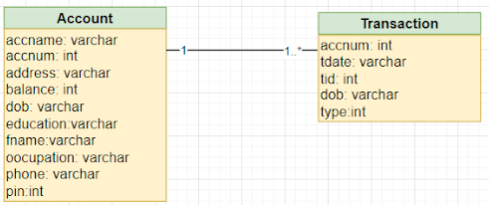




**Figure 11: Database created**

(Source: self-created in Netbeans)

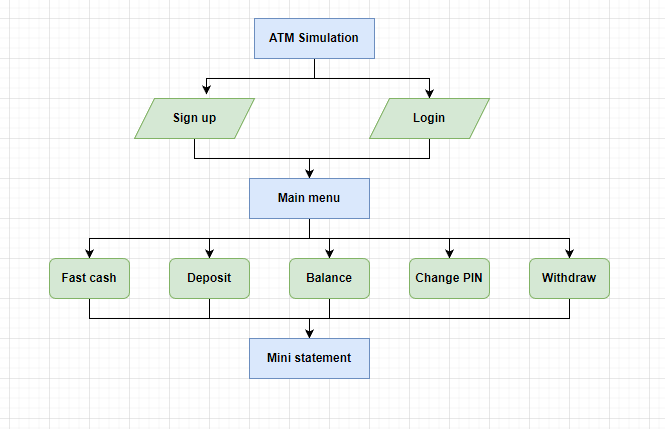
Database is created for the ATM simulator for storing values. Two types of table are created here. One table is for the account status and another is for the transaction status.



**Figure 12: UML diagram**

(Source: self-created in Draw.io)

In this representation, UML diagram is created here for the two tables, one is for the account database part and another is for the transaction part.



**Figure 13: Flow chart**

(Source: self-created in Draw.io)

This is the representation of flow chart.

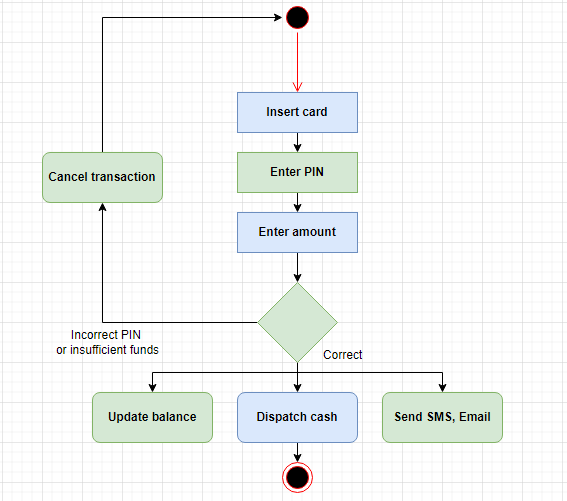
**Pseudo code**

Step 1: The user will have to create account or login into existing account.

Step 2: In the main menu page, there are various options.

Step 3: Fast cash, withdrawal, checking balance, deposit money and PIN can be changed.

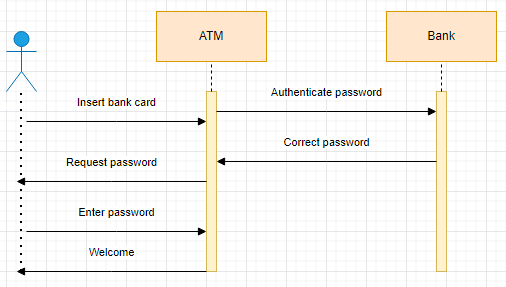
Step 4: A mini statement will be generated to check the information of the account.



**Figure 14: Activity diagram**

(Source: self-created in Draw.io)

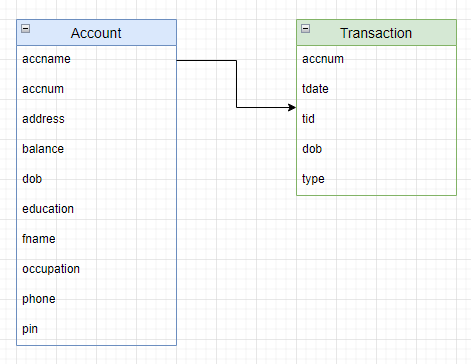
This is the representation of activity diagram.



**Figure 15: Sequence diagram**

(Source: self-created in Draw.io)

This is the representation of sequence diagram.



**Figure 16: Class diagram**

(Source: self-created in Draw.io)

This is the representation of class diagram.

# Conclusion

In conclusion, the Java-based simulation of an automated teller machine (ATM) offers both consumers and developers a useful learning opportunity. This project helps people to comprehend the inner workings of banking technology and learn useful knowledge about managing their funds by building a virtual depiction of an ATM system. User authentication, account administration, cash withdrawal, cash deposit, and fund transfers are all included in the simulation. These features allow users to practice and become familiar with the banking system in a secure and regulated setting by simulating the activities of an actual ATM.The ATM simulation may be created in Java with a number of advantages. Java is a popular programming language renowned for its stability, scalability, and cross-platform compatibility. It gives programmers the resources and libraries needed to implement the security and functionality requirements for an ATM simulation.

# Reference list

**Journals**

Akinnuwesi, B.A., Uzoka, F.M.E. and Obasa, A.P., 2023. Enhancing automated teller machine with multi-lingual and multi-denominational software functionalities. *International Journal of Electronic Finance*, *7*(2), pp.97-114.

Akinola Kayode, E., Adekunle, Y.A., Adebayo, A.O. and Okolie, S.O., 2019. Multi-factor authentication model for integrating iris recognition into an automated teller machine. *International Journal of Computer Applications*, *181*(45), pp.1-8.

Oko, S. and Oruh, J., 2022. Enhanced ATM security system using biometrics. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), p.352.

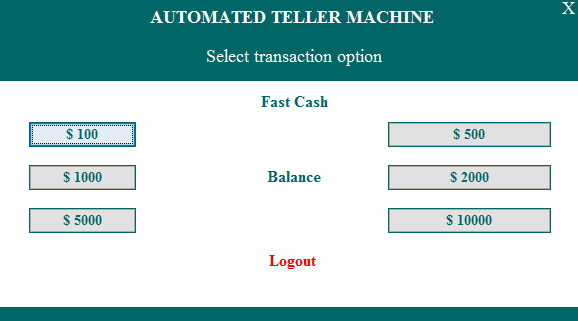
Sanusi, B.A., Ogunshile, E., Aydin, M., Olabiyisi, S.O. and Oyediran, M.O., 2022, August. DEVELOPMENT OF COMMUNICATING STREAM XMachine TOOL FOR MODELING AND GENERATING TEST CASES FOR AUTOMATED TELLER MACHINE. In *CS & IT Conference Proceedings* (Vol. 12, No. 14). CS & IT Conference Proceedings.

Wang, Y., Zhang, Y., Sheu, P.C., Li, X. and Guo, H., 2020. The formal design model of an automatic teller machine (ATM). *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *2*(1), pp.102-131.

Yahaya, S.N., Idakwo, H.O. and Nnegha, I., Development of Automated Teller Machine Simulated Software.

# Appendices

**Appendix 1: Figure 8: Fast cash page**



(Source: self-created in Netbeans)