

# Project Report

## ATM Interface in Java (Console Based Application)

**Submitted to:** Sudhanshu Kumar

**Submitted by:** Tushar Gupta (cstushar18@gmail.com)

**Date:** 08-June-2023

### **Abstract:**

This project aims to develop a console-based ATM interface in Java, designed to provide users with an easy-to-use and secure banking experience. The project comprises five different classes, namely account holder, account, bank transaction, bank, and the particular ATM of the bank.

The system provides user authentication, account details, transaction history, deposit, withdraw, transfer, and quit functionality. The project is developed using core Java and the domain of the project is banking.

The project was completed within the given timeline and fulfilled all the requirements specified in the problem statement. The system is scalable and secure, ensuring a seamless and secure banking experience for users.

# ACKNOWLEDGEMENT

First of all, I would like to thank my mentor **Mr. Hyder Abbas, Mr. Nitin M & Mr Navin** for helping me to acquire in-depth knowledge of “Java Programming Language”. At the same time, he gave me the opportunity to learn something new related to our module like constructors, methods, arrays, JDBC etc. Beside from my lecturer, I like to thank my other INEURON Support staff for helping to understand the assignment related questions more clearly. They gave their best for completing this report on time. I thank them for their efforts.

# ***Table of Contents:***

## ***1. Introduction***

### ***1.1 Problem Statement***

### ***1.2 Objectives***

### ***1.3 Scope***

### ***1.4 Overview***

## ***2. Explanation***

## ***3. System Requirements***

### ***3.1 Hardware Requirements***

### ***3.2 Software Requirements***

## ***4. System Design***

### ***4.1 High-Level Design***

### ***4.2 Low-Level Design***

### ***4.3 Wireframe Design***

## ***5. Implementation***

### ***5.1 Technologies Used***

### ***5.2 Functionalities Implemented***

## ***6. Conclusion***

### ***6.1 Project Summary***

### ***6.2 Limitations***

### ***6.3 Future Work***

## ***7. References***

# 1. Introduction:

## 1.1 Problem Statement:

The problem statement of this project is to develop a console-based ATM interface in Java that provides users with a secure and easy-to-use banking experience. The system must have user authentication, account details, transaction history, deposit, withdraw, transfer, and quit functionality.

## 1.2 Objectives:

*The objectives of this project are as follows:*

- To develop a console-based ATM interface in Java that provides users with a secure and easy-to-use banking experience.
- To implement user authentication, account details, transaction history, deposit, withdraw, transfer, and quit functionality.
- To ensure the scalability and security of the system.

## 1.3 Scope:

The scope of this project is to develop a console-based ATM interface in Java that provides users with a secure and easy-to-use banking experience. The system must have user authentication, account details, transaction history, deposit, withdraw, transfer, and quit functionality. The system must be scalable and secure to ensure a seamless and secure banking experience for users.

## 1.4 Overview:

This project comprises five different classes, namely account holder, account, bank transaction, bank, and the particular ATM of the bank. The system provides user authentication, account details, transaction history, deposit, withdraw, transfer, and quit functionality. The project is developed using core Java and the domain of the project is banking.

## 2. EXPLANATIONS

In this documentation we have given explanations of how to interact successfully with this ATM (Automated Teller Machine). We have explained here step by step so that it will surely help users to become more user friendly with it. Below are our explanations:

### First Things First:

Before execute this program users need to do some works so that it will run properly into their system. First they need to make sure their system is having “JDK”. If they don’t have it then they can download from this below link:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Depending on their system (Windows 64bit/32bit) they need to download and install. Then they need to add the “JAVA” files to their system “PATH” so that the system can run the program from CMD (Command Prompt). The path will show something like this “C:\Program Files (x86)\Java\jre1.8.0\_25\bin;” Now just add the address besides the current path directory and save it.

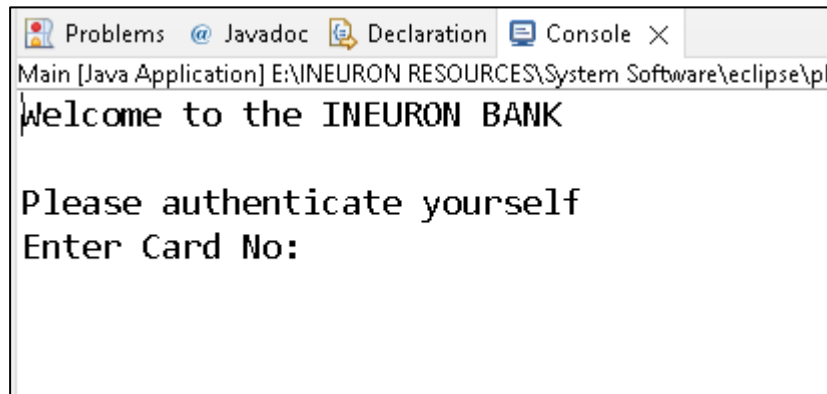
The other way they can execute this program in to download the IDE (Integrated Development Environment) on their system. They can download ECLIPSE or NETBEANS depending on the windows (32bit/64bit). Below is the link:

<b>NETBEANS:</b>	<a href="https://netbeans.org/downloads/">https://netbeans.org/downloads/</a>
<b>ECLIPSE:</b>	<a href="http://www.eclipse.org/downloads/">http://www.eclipse.org/downloads/</a>

We developed this program using “ECLIPSE”.

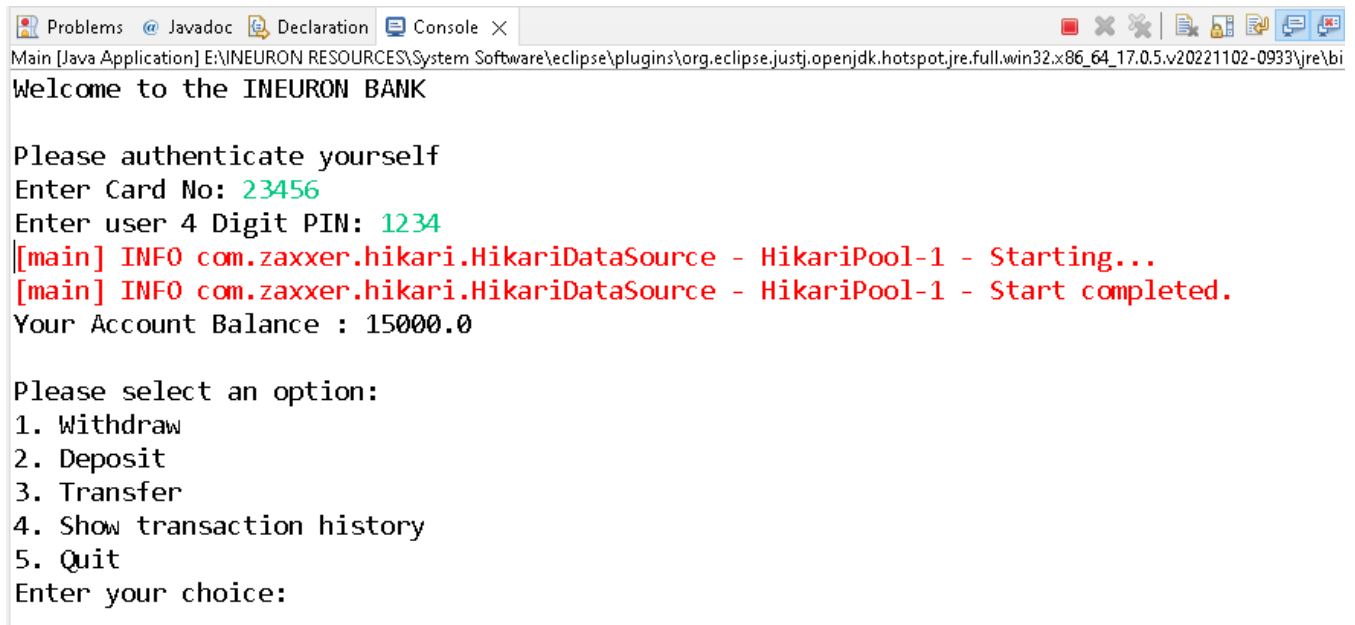
### ***Execution Procedures:***

When user executes this program it will show the details (Welcome to the ATM, Please Enter your card & pin) from the text file in the console.



**FIGURE 1: CONSOLE**

If they enter invalid card no or pin the system will show the warning message. Ref. Figure2



**FIGURE 2: AUTHENTICATION SUCCESSFUL**

```
<terminated> Main [Java Application] E:\INEURON RESOURCES\System Software\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5
Welcome to the INEURON BANK

Please authenticate yourself
Enter Card No: 12345
Enter user 4 Digit PIN: 123
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
Invalid credentials, Please try again
```

**FIGURE 3: AUTHENTICATION FAILED**

If the user enters valid username and password then it will show the “MENU” on their screen. Ref. Figure3

***Function (Balance):***

This function is auto call function, when user got authenticate then system will show the “BALANCE” available in the users account.

```
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
Your Account Balance : 15000.0

Please select an option:
1. Withdraw
2. Deposit
3. Transfer
4. Exit
```

**FIGURE 4: CURRENT BALANCE**

### ***Function (Deposit):***

To deposit into the account user select the option “**DEPOSIT**” from the menu. They need to enter the amount to deposit into their account.

```
<terminated> Main [Java Application] E:\INEURON RESOURCES\Sys
Welcome to the INEURON BANK

Please authenticate yourself
Enter Card No: 23456
Enter user 4 Digit PIN: 1234
[main] INFO com.zaxxer.hikari.HikariDataSource - Hi
[main] INFO com.zaxxer.hikari.HikariDataSource - Hi
Your Account Balance : 15000.0

Please select an option:
1. Withdraw
2. Deposit
3. Transfer
4. Show transaction history
5. Quit
Enter your choice: 2
Enter amount you want to deposit:
2000
Deposit successful. Your new balance is 17000.0
```

**FIGURE 5: DEPOSIT AMOUNT**

### ***Function (Withdraw):***

To withdraw from the account user select the option “**WITHDRAW**”. Where user enters the total amount to withdraw from their account.

```
Problems @ Javadoc Declaration Console X
Main [Java Application] E:\INEURON RESOURCES\System Software\eclipse\plu
Welcome to the INEURON BANK

Please authenticate yourself
Enter Card No: 23456
Enter user 4 Digit PIN: 1234
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool
Your Account Balance : 17000.0

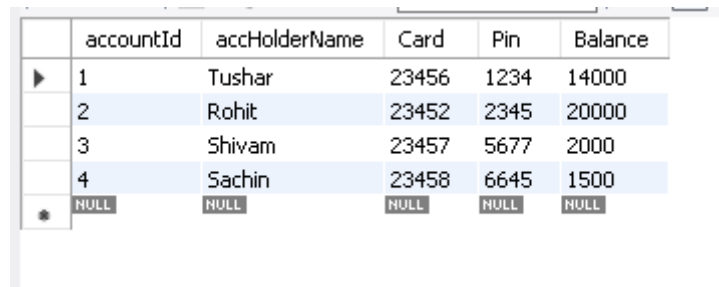
Please select an option:
1. Withdraw
2. Deposit
3. Transfer
4. Show transaction history
5. Quit
Enter your choice: 1
Enter amount you want to withdraw:
3000
Withdrawal successful. Your new balance is 14000.0
```

**FIGURE 6: WITHDRAW**



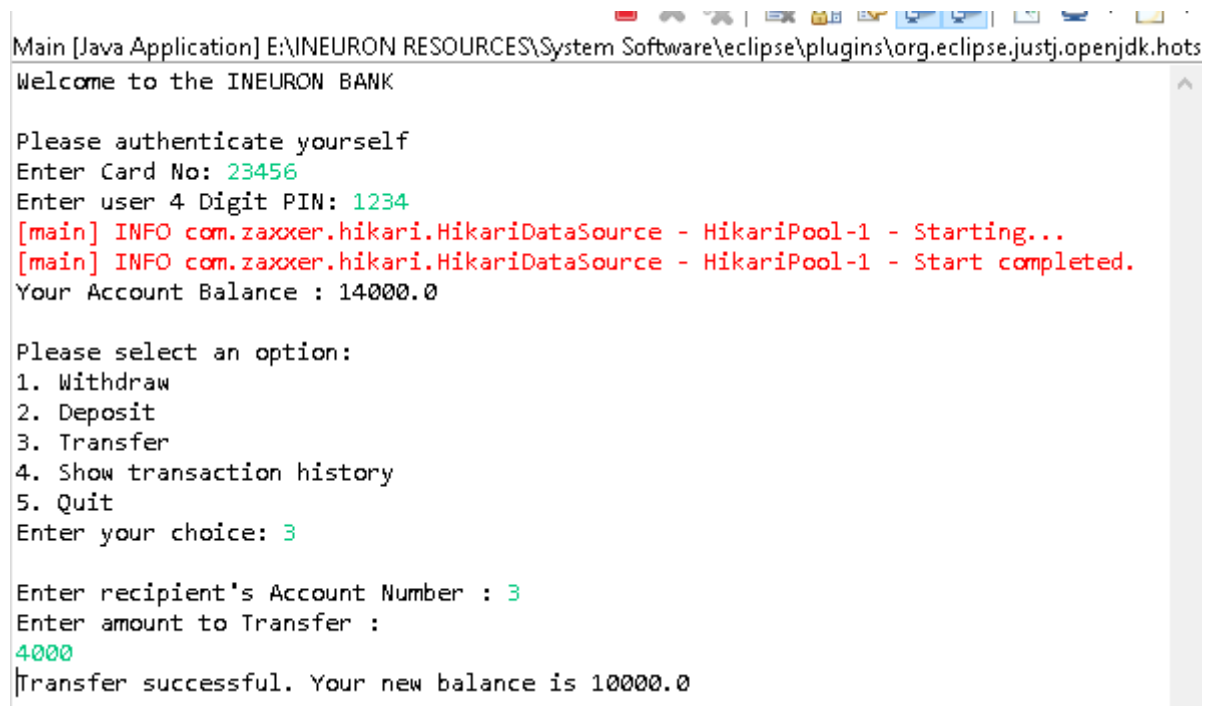
### ***Function (Transfer):***

User can also transfer amount to other users account. For that, user needs to select the option “TRANSFER”. The user needs to have other user account no and the total amount for transfer. As for example other users like “Shivam” having Balance 2000 in his account. If Shivam receive more from other user it will add to his account and the total account will become 6000.



	accountId	accHolderName	Card	Pin	Balance
▶	1	Tushar	23456	1234	14000
	2	Rohit	23452	2345	20000
	3	Shivam	23457	5677	2000
	4	Sachin	23458	6645	1500
*	NULL	NULL	NULL	NULL	NULL

**FIGURE 7: DATABASE BEFORE TRANSFER**



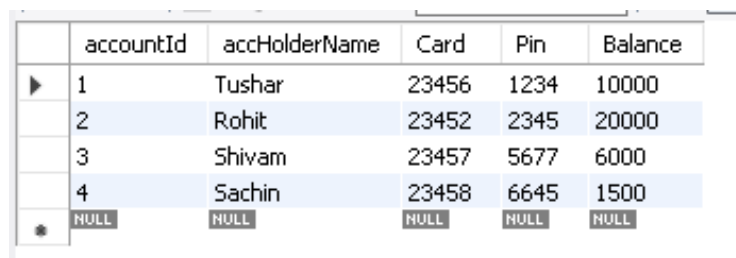
```
Main [Java Application] E:\INEURON RESOURCES\System Software\eclipse\plugins\org.eclipse.justj.openjdk.hot
Welcome to the INEURON BANK

Please authenticate yourself
Enter Card No: 23456
Enter user 4 Digit PIN: 1234
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
Your Account Balance : 14000.0

Please select an option:
1. Withdraw
2. Deposit
3. Transfer
4. Show transaction history
5. Quit
Enter your choice: 3

Enter recipient's Account Number : 3
Enter amount to Transfer :
4000
Transfer successful. Your new balance is 10000.0
```

**FIGURE 8: TRANSFER TRANSACTION**

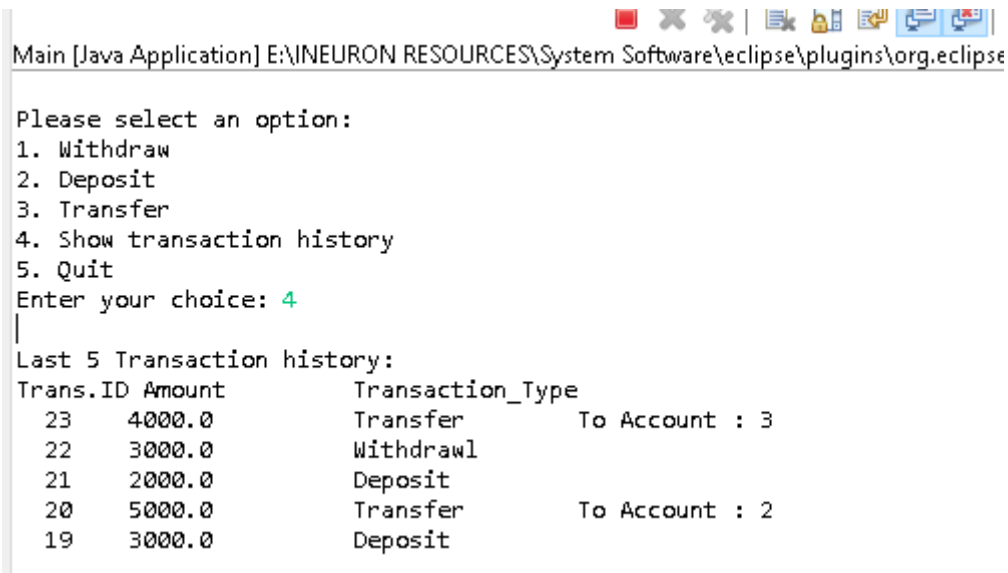


	accountId	accHolderName	Card	Pin	Balance
▶	1	Tushar	23456	1234	10000
	2	Rohit	23452	2345	20000
	3	Shivam	23457	5677	6000
	4	Sachin	23458	6645	1500
*	NULL	NULL	NULL	NULL	NULL

**FIGURE 9: DATABASE AFTER SUCCESSFUL TRANSFER**

### ***Function (Show Transaction History):***

User can also check its last 5 transaction history using this function. Transaction history also shows the type of transaction done by the user. For example: If user done a withdrawal, history shows the amount of withdrawn take by the user from the ATM.



```
Main [Java Application] E:\INEURON RESOURCES\System Software\eclipse\plugins\org.eclipse
Please select an option:
1. Withdraw
2. Deposit
3. Transfer
4. Show transaction history
5. Quit
Enter your choice: 4
|
Last 5 Transaction history:
Trans.ID Amount      Transaction_Type
  23   4000.0      Transfer      To Account : 3
  22   3000.0      Withdrawl
  21   2000.0      Deposit
  20   5000.0      Transfer      To Account : 2
  19   3000.0      Deposit
```

**FIGURE 8: SHOW TRANSACTION HISTORY**

## 3. *System Requirements:*

### *3.1 Hardware Requirements:*

- Processor: Intel Core i3 or higher
- RAM: 4 GB or higher
- Hard Disk: 500 GB or higher

### *3.2 Software Requirements:*

- Operating System: Windows 10 or higher
- Java Development Kit (JDK) version 8 or higher
- Eclipse IDE for Java Developers or any other Java IDE
- MySQL Connector J Jar Driver
- Hikari\_CP Connection Pool Jar from Maven Repository

## 4. *System Design:*

### *4.1 High-Level Design:*

The ATM interface project can be designed with the following modules:

#### **User authentication module:**

This module is responsible for authenticating the user with their user id and pin. If the authentication is successful, it will proceed to the next module. Otherwise, it will prompt the user to enter the correct user id and pin.

#### **Account module:**

This module is responsible for displaying the account details of the user. It includes the account balance, account number, and other details.

#### **Transaction module:**

This module is responsible for performing all the banking transactions, including deposit, withdraw, transfer, and viewing transaction history.

#### **ATM module:**

This module is responsible for integrating all the other modules and providing the user interface to access the functionalities of the ATM.

## ***4.2 Low-Level Design***

### ***User authentication module:***

The user authentication module can be designed with a class named "UserAuthentication." This class should contain two methods, "authenticateUser()" and "validateUserCredentials()". The "authenticateUser()" method should prompt the user to enter the user id and pin, while the "validateUserCredentials()" method should validate the user credentials against the user account database.

### ***Account module:***

The account module can be designed with a class named "AccountDetails." This class should contain methods like "getAccountBalance()", "getAccountNumber()", and other account details. These methods should fetch the account details from the account database.

### ***Transaction module:***

The transaction module can be designed with a class named "Transaction." This class should contain methods like "deposit()", "withdraw()", "transfer()", and "viewTransactionHistory()". The "deposit()" method should deposit the specified amount into the user's account, while the "withdraw()" method should withdraw the specified amount from the user's account. The "transfer()" method should transfer the specified amount from the user's account to another user's account. The "viewTransactionHistory()" method should fetch the transaction history from the transaction database.

### ***ATM module:***

The ATM module can be designed with a class named "ATM." This class should contain the main method, which should integrate all the other modules and provide the user interface to access the functionalities of the ATM. The "main()" method should call the "authenticateUser()" method of the "UserAuthentication" class to authenticate the user. If the authentication is successful, it should proceed to the next module. Otherwise, it should prompt the user to enter the correct user id and pin. Once the user is authenticated, it should display the account details using the "getAccountDetails()" method of the "AccountDetails" class. The user should be able to perform all the banking transactions using the methods of the "Transaction" class. Finally, the user should be able to quit the ATM interface by selecting the "quit" option.

## ***4.3 Wireframe Design***

The wireframe design of the ATM interface system architecture will consist of different screens, including the login screen, account details screen, transaction screen, and confirmation screen. The login screen will prompt the user to enter their user id and pin, while the account details screen will display the account balance, account number, and other details. The transaction screen will provide options to deposit, withdraw, transfer, and view transaction history. The confirmation screen will confirm the user's selected transaction and provide a summary of the transaction details.

Overall, the system architecture design and wireframe design will ensure the smooth functioning of the ATM interface, providing users with a seamless banking experience.

## 5. Implementation:

### 5.1 Technologies Used:

*The implementation of the ATM interface in Java (Console Based Application) utilizes the following technologies:*

- **Java:** The core programming language used to develop the entire project.
- **Eclipse IDE:** An Integrated Development Environment that provides tools for Java development, including code editing, debugging, and compilation.
- **MySQL:** A relational database management system used to store and manage user account details, transaction history, and other relevant data.
- **JDBC (Java Database Connectivity):** A Java API used for connecting to and interacting with the MySQL database.
- **Hikari\_CP from Maven Repository:** A Connection pooling API used to manage connection data Source.
- **Git:** A version control system used for tracking changes made to the project's source code.

*These technologies were chosen for their compatibility with Java development and their ability to support the required functionalities of the ATM interface.*

### 5.2 Functionalities Implemented:

*The ATM interface project successfully implements the following functionalities:*

**1. User Authentication:** *The system prompts the user to enter their user ID and PIN for authentication. If the provided credentials are valid, the user gains access to the ATM functionalities. Otherwise, an authentication failure message is displayed.*

**2. Show Transaction History:** *Users can view their transaction history, which displays details such as transaction type (withdrawal, deposit, or transfer) and the corresponding transaction amounts.*

**3. Withdraw:** *Users can initiate cash withdrawals by specifying the desired amount. The system verifies the availability of sufficient funds in the user's account and updates the account balance accordingly.*

**4. Deposit:** Users can deposit money into their account by entering the amount they wish to deposit. The system records the transaction and updates the account balance accordingly.

**5. Transfer:** Users can transfer funds from their account to another user's account. They provide the recipient's account details and the desired amount to be transferred. The system validates the transaction, deducts the transferred amount from the sender's account, and updates the balances of both accounts accordingly.

**6. Quit:** Users have the option to exit the ATM interface, terminating their banking session.

These functionalities were implemented using Java programming concepts such as classes, methods, and conditional statements, along with the integration of database operations to store and retrieve user account information and transaction history.

**The successful implementation of these functionalities ensures that users can perform essential banking operations through the console-based ATM interface.**

## 6. Conclusion

### 6.1 Project Summary

The project "ATM Interface in Java (Console Based Application)" aims to develop a user-friendly and secure banking experience through a console-based ATM interface. The project consists of five essential classes: account holder, account, bank transaction, bank, and the specific ATM of the bank.

To use the program, the user is prompted to enter their user ID and PIN. Upon successful authentication, the user gains access to various functionalities that are commonly found in an ATM. These functionalities include:

**1. Show Transaction History:** Users can view their transaction history, which provides details of their past activities such as withdrawals, deposits, and transfers.

**2. Withdraw:** Users can withdraw cash from their account by specifying the desired amount. The system ensures the availability of funds and updates the account balance accordingly.

**3. Deposit:** Users can deposit money into their account by entering the amount they wish to deposit. The system records the transaction and updates the account balance accordingly.

**4. Transfer:** Users can transfer funds from their account to another user's account. They need to provide the recipient's account details and the desired amount to be transferred. The system ensures the validity of the transaction and updates the account balances of both parties involved.

**5. Quit:** *Users have the option to exit the ATM interface, concluding their banking session.*

*By implementing these functionalities, the project successfully provides a comprehensive ATM experience in a console-based environment.*

Overall, the project achieves its objective of developing a user-friendly and secure ATM interface using Java. It offers users a range of banking operations, including transaction history viewing, cash withdrawals, deposits, fund transfers, and the ability to conclude their session. The project demonstrates proficiency in programming concepts and addresses the requirements outlined in the problem statement.

## **6.2 Limitations:**

*Despite the successful implementation of the ATM interface in Java (Console Based Application), there are a few limitations to be aware of:*

**1. Console Interface:** The project is designed as a console-based application, which may limit the user experience compared to a graphical user interface (GUI). The absence of visual elements and interactive components may make the interface less intuitive for some users.

**2. Security Measures:** While the project aims to provide a secure banking experience, it is important to note that the level of security implemented may not be on par with real-world banking systems. Additional security measures such as encryption, secure network communication protocols, and multi-factor authentication could be considered for future enhancements.

**3. Scalability:** The current implementation of the ATM interface may not be optimized for handling a large number of concurrent users or extensive transaction volumes. It may require further enhancements to ensure scalability and performance under increased system load.

## **6.3 Future Work:**

*To improve and expand the functionality of the ATM interface in Java (Console Based Application), the following areas can be considered for future work:*

**1. Graphical User Interface (GUI):** Enhance the user experience by developing a graphical user interface that provides a visually appealing and intuitive interface for users to interact with the ATM functionalities.

**2. Enhanced Security Features:** Implement additional security measures to ensure a higher level of data protection and user authentication. This could include encryption of sensitive information, secure communication protocols, and the integration of biometric authentication methods.

**3. Transaction Validation and Error Handling:** Implement robust validation mechanisms to ensure the accuracy and integrity of transactions. Enhance error handling capabilities to provide informative error messages and handle exceptional scenarios effectively.

**4. Account Management Features:** Extend the functionalities to include features such as creating new user accounts, updating user information, and managing account settings.

**5. Integration with Banking Systems:** Explore the possibility of integrating the ATM interface with real banking systems to enable seamless connectivity and access to live account information, transaction records, and other banking services.

**6. Performance Optimization:** Fine-tune the application to improve performance and efficiency, making it more responsive and capable of handling a larger user base and higher transaction volumes.

*By addressing these future work areas, the ATM interface can evolve into a more comprehensive and feature-rich banking application, providing users with an enhanced and secure banking experience.*

## **7. References:**

1.	Eclipse IDE:	➤ <a href="https://www.eclipse.org/downloads/">https://www.eclipse.org/downloads/</a>
2.	Java JDK	➤ <a href="https://www.oracle.com/in/java/technologies/downloads/">https://www.oracle.com/in/java/technologies/downloads/</a>
3.	MySQL Server	➤ <a href="https://dev.mysql.com/downloads/mysql/">https://dev.mysql.com/downloads/mysql/</a>
4.	MySQL connector J Driver	➤ <a href="https://dev.mysql.com/downloads/connector/j/">https://dev.mysql.com/downloads/connector/j/</a>
5.	Hikari Cp Driver:	➤ <a href="https://mvnrepository.com/artifact/com.zaxxer/HikariCP">https://mvnrepository.com/artifact/com.zaxxer/HikariCP</a>