**NBKR Institute Of Science and Technology**

**Project Report**

**Project Title : SIMPLE ATM SIMULATION**

**Department: Computer Science Engineering**

**Section: E**

**Year:**  **I**

**Semester:** **II**

**Date: May 07 2025**

**Submitted to: Ashok Selva Kumar**

**Submitted by:**

**1. SK .HASMATH (24KB1A05HM)**

**2.RAYAPANENI JITHENDRA (24KB1A05FV)**

# **Acknowledgement:**

# We would like to express our sincere gratitude to everyone who supported us in completing our project titled “SIMPLE ATM SIMULATION.”

# First and foremost, we thank our respected guide/teacher [Ashok Selvan Kumar] for their valuable guidance, continuous support, and encouragement throughout the project. Their suggestions helped us improve our understanding and approach.

# We are also thankful to our institution [NBKRIST] for providing us with the necessary resources and an environment conducive to learning and development.

# We extend our heartfelt thanks to our friends for their support, motivation, and understanding during the entire project work.

# This project helped us gain hands-on experience in C programming, particularly in areas like control structures, switch-case, file handling, and user-defined functions involved in building real-time simulations like ATM systems.

# A very special thanks to all who contributed to the success of our project.

## **ABSTRACT:**

## The project titled “Simple ATM Simulation” is a console-based application developed in the C programming language. It simulates the fundamental operations of a real Automated Teller Machine (ATM), such as PIN authentication, balance inquiry, money deposit, and withdrawal. The system provides a user-friendly, menu-driven interface that uses switch-case statements for option selection and file handling to store and retrieve account balance information persistently.

## This project aims to demonstrate the practical application of basic C programming concepts including user-defined functions, conditional logic, loops, file input/output operations, and pointer usage. The user is first prompted to enter a secure PIN, after which they are given access to perform transactions. The balance is stored and updated using a text file (`account.txt`), allowing the simulation of persistent account management.

## The Simple ATM Simulation serves as an effective educational tool for understanding how real-life banking operations can be modeled programmatically. It emphasizes structured coding practices and real-world problem solving using C.

## Introduction

The Simple ATM Simulation project is a console-based C program that mimics the basic functionalities of an Automated Teller Machine (ATM). The system allows users to securely access their account using a PIN and perform basic transactions such as withdrawing money, depositing money, and checking account balance. This simulation demonstrates the use of fundamental programming concepts including switch-case structures and file input/output (I/O) operations.

## Objectives

* Implement a basic ATM-like interface in C.
* Secure access using a PIN system.
* Enable transactions: Withdraw, Deposit, and Balance Inquiry.
* Store and update account data using file operations.
* Practice use of switch-case statements to manage menu-driven choices.

## SYSTEM REQUIREMENTS :

**PIN Authentication:**

The user is required to enter a predefined PIN to access the ATM functionalities. Incorrect attempts are limited.

**Transaction Options (Using Switch-Case):**

Once authenticated, the user is presented with a menu:

* Check Balance
* Deposit Money
* Withdraw Money
* Exit

Each menu choice is handled via a switch statement for cleaner control flow.

**File I/O for Account Data:**

The account balance is read from and written to a text file (account.txt), simulating persistent storage.

## METHODOLOGY:

- Language Used: C

- Key Libraries: stdio.h, stdlib.h, string.h

- File Used: account.txt to store the balance.

- Security: Basic PIN protection (hardcoded for simplicity).

## PROJECT DESCRIPTION

1. **PROBLEM STATEMENT:**-

The main objective of this project is to design and implement a Simple ATM Simulation using the C programming language that enables users to perform essential banking operations through a secure and interactive command-line interface.

In real-world scenarios, ATMs are used for secure and instant access to banking services such as checking balance, depositing money, and withdrawing funds. To simulate such functionality in a simplified environment, this project must:

Prompt the user for a secure 4-digit PIN and restrict access on incorrect entries.

Allow authenticated users to choose from a menu of operations using a switch-case mechanism.

Support key functionalities:

Balance Check: View the current account balance.

Deposit Money: Add funds to the account.

Withdraw Money: Remove funds from the account, with validation to prevent overdraft.

Use file I/O operations to read the balance from and write the updated balance to a file, simulating persistent account data

1. **PROPOSED SOLUTION:-**

To simulate a basic ATM system, we propose a C program that ensures secure PIN-based access and allows users to perform key banking operations through a menu-driven interface. The program uses switch-case statements to manage user choices and file handling to store and update the account balance.

The balance is stored in a file (account.txt) and updated after every transaction to simulate persistent storage. The system includes basic validation to prevent overdrafts and invalid inputs. Separate functions are used for deposit, withdrawal, and balance checking to maintain modularity and clarity.

1. **KEY FEATURES:**

PIN Authentication:

Ensures secure access by verifying a 4-digit PIN before allowing any transactions.

Menu-Driven Interface:

Uses a `switch-case` structure to let users choose between balance check, deposit, withdrawal, or exit.

File I/O Operations:

Reads the account balance from a file and updates it after every transaction to simulate data persistence.

Modular Functions:

Separate functions for deposit, withdrawal, and balance check make the code organized and reusable.

Input Validation:

Prevents invalid operations like withdrawing more than the balance or entering negative amounts.

**ALGORITHM** :

Step 1: Start the program.

Step 2: Prompt the user to enter a 4-digit PIN.

Step 3: If the PIN is incorrect, deny access and exit.

Step 4: If the PIN is correct, display the main menu with options:

Check Balance

Deposit Money

Withdraw Money

Exit

Step 5: Read the balance from account.txt.

Step 6: Use a switch-case to handle the user’s choice:

Check Balance: Display the current balance.

Deposit: Ask for deposit amount, add to balance, and update the file.

Withdraw: Ask for amount, check if sufficient balance, subtract and update the file.

Exit: Display thank you message and end the program.

Step 7: Loop the menu until the user chooses to exit.

Step 8: End the program

**PROGRAM CODE:**

#include <stdio.h>

#include <stdlib.h>

void checkBalance(int balance) {

printf("Current Balance: ₹%d\n", balance);

}

void deposit(int \*balance) {

int amount;

printf("Enter amount to deposit: ₹");

scanf("%d", &amount);

\*balance += amount; // Add deposit to balance

printf("Deposit successful! New Balance: ₹%d\n", \*balance);

}

void withdraw(int \*balance) {

int amount;

printf("Enter amount to withdraw: ₹");

scanf("%d", &amount);

if (amount > \*balance) {

printf("Insufficient balance!\n");

} else {

\*balance -= amount; // Deduct withdrawal

printf("Withdrawal successful! Remaining Balance: ₹%d\n", \*balance);

}

}

int main() {

int enteredPin, savedPin, balance = 10000, choice;

// Step 1: Handle PIN creation or reading

printf("No PIN found. Please create a new 4-digit PIN: ");

scanf("%d", &savedPin);

printf("PIN created successfully!\n");

// Ask for PIN

printf("Enter your PIN: ");

scanf("%d", &enteredPin);

if (enteredPin != savedPin) {

printf("Incorrect PIN. Access Denied.\n");

return 0;

}

// Step 2: ATM Menu

do {

printf("\nATM Menu:\n");

printf("1. Check Balance\n");

printf("2. Deposit Money\n");

printf("3. Withdraw Money\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

checkBalance(balance);

break;

case 2:

deposit(&balance);

break;

case 3:

withdraw(&balance);

break;

case 4:

printf("Thank you for using the ATM!\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 4);

return 0;

}

**OUTPUT SCREEN SHOT :**

## 2025-05-08 07:50:29.560000

**TESTING AND VALIDATION:**

1. Functional Testing:

Each core feature of the ATM system was tested to ensure correct behavior:

* PIN Verification:

\* ✅ Correct PIN grants access.

\* ❌ Incorrect PIN denies access.

Check Balance:

Displays the current balance accurately from `account.txt`.

Deposit Money:

Adds valid positive amounts to the balance.

Rejects zero or negative inputs.

Withdraw Money:

Deducts amount if sufficient balance is available.

Displays error if withdrawal exceeds balance

Rejects invalid or negative inputs.

Exit Option:

Exits the program gracefully after displaying a thank-you message.

2. File I/O Testing:

\* Verified that `account.txt` is created if it does not exist.

\* Confirmed that balance is read and updated correctly after every transaction.

\* Ensured data persists between program runs.

3. Input Validation Testing:

\* Non-numeric inputs were handled with manual safeguards.

\* Negative and zero amounts for deposit/withdraw were blocked.

4. Edge Case Testing:

\* Withdrawing the entire balance sets it to zero without error.

\* Depositing and then withdrawing immediately reflects correct updated values.

\* No crash when balance file is missing or empty on first run.

5. Validation Strategy:

Manual Testing:

Inputs were provided via keyboard to simulate real user behavior.

Console Output Verification: Every action’s result was manually checked.

File Inspection:The contents of `account.txt` were reviewed after each transaction to ensure correct updates.

## Conclusion

This project demonstrates a simple simulation of ATM functionalities using core C programming concepts. It highlights user interaction, control structures (switch-case), and persistent storage using file operations. This project can be extended further to support multiple users, enhanced security, and GUI interfaces.

**REFERENCE:**

1. Algorithms and Data Structures: The Basic Toolbox by Kurt Mehlhorn and Peter

Sanders

2. C Data Structures and Algorithms by Alfred V. Aho, Jeffrey D. Ullman, and John E.

Hopcroft

3. Problem Solving with Algorithms and Data Structures" by Brad Miller and David

Ranum

4. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L.

Rivest, and Clifford Stein

**WEBSITES:**

**<http://www.geeksforgeeks.org>**