ATM MACHINE PROJECT REPORT

SUBMITTED BY: KRISHNA MEWADA

REGISTRATION NO:25MIM10138

PROJECT TITLE:

ATM MACHINE DESIGN, DEVELOPMENTM AND ANALYSIS

ACADEMIC YEAR:2025-26

Introduction:

An Automated Teller Machine (ATM) is an electronic banking device that allows customers to perform financial transactions without visiting a bank branch.

It provides services like cash withdrawal, balance inquiry, fund transfer, mini statements, and PIN change.

ATMs help reduce crowd in banks and offer 24×7 availability, increasing convenience for users.

They are connected to a secure banking network that verifies user identity using ATM cards and PIN authentication.

Modern ATMs support advanced features like cash deposit, mobile recharge, bill payment, and cardless transactions.

The ATM system uses hardware (card reader, keypad, cash dispenser) and software to ensure smooth and secure transactions.

ATMs improve banking efficiency and support the digitization of the financial sector by enabling quick self-service operations.

With increasing digital payments, ATMs remain essential for providing reliable cash access to customers.

Problem Statement

Users need a quick and reliable way to withdraw cash without visiting banks.

Manual transactions in banks take time and cause long queues.

Ensuring secure money transactions is difficult without automation.

Users need 24×7 financial services which banks alone cannot provide.

The system must prevent fraud through PIN verification and secure communication.

Banks require an automated system to reduce workload and improve customer service.

Functional Requirements

User authentication via card number and PIN.

Cash withdrawal with limit validation.

Balance inquiry and mini-statement generation.

Fund transfer between accounts.

PIN change option.

Cash deposit (optional feature).

Transaction receipt printing.

Display of transaction status (success/failure).

Error handling (invalid PIN, insufficient balance, card blocked).

Non-Functional Requirements

Security: Data encryption, secure PIN handling.

Performance: Fast response time (<3 seconds for main operations).

Availability: System uptime of 99%.

Reliability: Accurate and error-free transaction processing.

Usability: Simple interface for all age groups.

Maintainability: Easy to update and fix bugs.

Scalability: Able to support increasing number of transactions.

Portability: Should support different banks’ ATM networks.

System Architecture

ATM device interacts with user using screen, keypad, card reader.

ATM communicates with the bank server through a secure network.

Bank server verifies user credentials and account details.

Transaction processor handles the debit/credit operations.

Database maintains user account information.

Security layer encrypts communication between ATM and bank server.

System Architecture Diagram

```
+——————————+
|      User          |
+—————+—————+
         |
         v
+——————————+
|   ATM Machine UI      |
| (Screen, Keypad etc.) |
+—————+—————+
         |
         v
+———————————+
| Transaction Processor |
+—————+——————+
         |
         v
+———————————+

|   Bank Server         |


+—————+——————+
         |
         v
+———————————+
|   Account Database    |
+————————
```

Design Diagrams

A) Case Diagram

```
+———————+
      |     User      |
      +———+———+
            |
   ————————————————————
   |           |          |         |
Withdraw Cash   Check Balance Fund Transfer PIN Change
```

B) Workflow Diagram

```
Start
  |
Insert Card
  |
Enter PIN

[PIN Valid?] – No –> Error Message –> End
   |
  Yes
   |
Select Transaction
   |
Perform Transaction
                   |

Print Receipt
   |
Remove Card
   |
End
```

C) Sequence Diagram

```
User      ATM Machine       Bank Server       Database
 |            |                 |                 |
 | Insert Card|                 |                 |
 |———–—>|                 |                 |
 |            | Validate PIN    |                 |
 |            |———————> |                 |
 |            |                 | Check PIN       |
 |            |                 |———————> |
 |            |                 |     Result      |
 |          | <———————|              |
 | Select Txn |              |
|————>| Process Txn    |              |

 |          |———————> | Update Balance |
 |          |              |———————> |
 |          |              |    OK        |
 |          | <———————|              |
 |          | Show Result    |              |
```

D) Class Diagram

```
+—————————+
|     ATM        |
+—————————+
| - atmID          |
| - location       |
+—————————+
| + authenticate()  |
| + processTxn()    |
+—————————+

+—————————+
|    Account      |
+—————————+
| - accNumber      |
| - accHolder      |
| - balance        |

+————————+
| + update Balance() |
+————————+

+————————+
|    Transaction    |
+————————+
| - txnID          |
| - amount         |
| - date           |
+————————+
| + validate()     |
```

E) Component Diagram

```
+————————+      +————————+
|  ATM Interface  |      | Bank Server API  |
+————————+      +————————+
          \               /
           \             /
            +——————+
            |  Network  |
            +——————+
```

F) Entity–Relationship (ER) Diagram

```
+———+         +————+        +———–+
| Customer | 1 — * |   Account    | 1 — *| Transaction |
+———+         +————+        +———–+
| custID   |         | accNumber    |        | txnID       |
| name     |         | balance      |        | date        |
+———+         +————+        +———–+
```

Design Decisions and Rationale

Modular approach chosen for easy maintenance.

PIN authentication ensures security and prevents unauthorized access.

Client-server architecture selected for centralized control.

Encrypted communication used to prevent data theft.

Database normalization reduces redundancy and improves performance.

Simple UI design improves usability for general users.

Scalable architecture for supporting large transactions.

Implementation Details

Developed using Java/Python (or your chosen language).

Database created using MySQL/Oracle/SQLite.

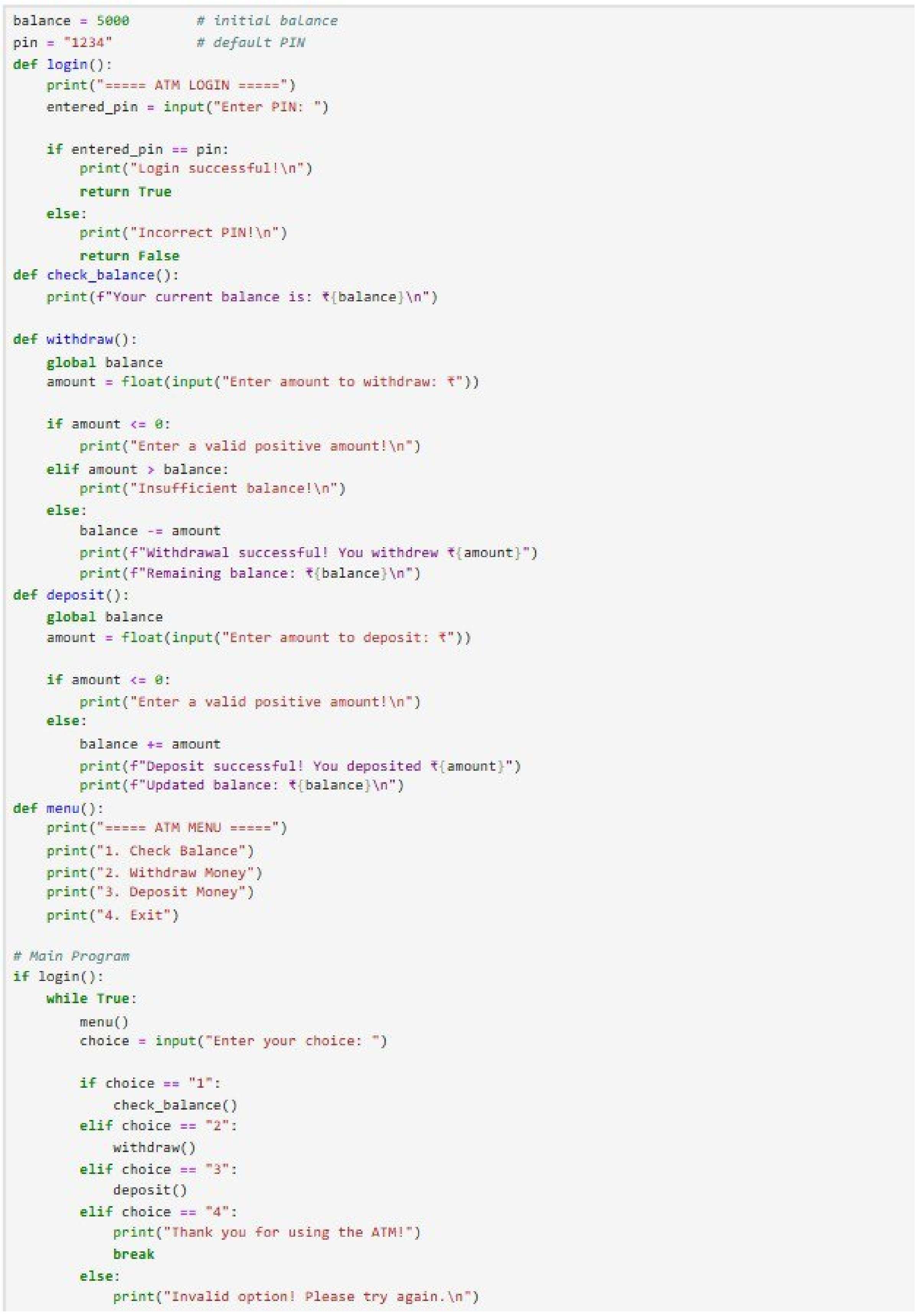
GUI implemented with simple menu-based screens.

API calls used to simulate communication with bank servers.

Error handling implemented for invalid PIN, insufficient funds, etc.

Testing done using sample accounts and dummy transaction data.

SCREENSHOT AND RESULT:



```
balance = 5000        # initial balance
pin = "1234"          # default PIN
def login():
    print("===== ATM LOGIN =====")
    entered_pin = input("Enter PIN: ")

    if entered_pin == pin:
        print("Login successful!\n")
        return True
    else:
        print("Incorrect PIN!\n")
        return False
def check_balance():
    print(f"Your current balance is: ₹{balance}\n")

def withdraw():
    global balance
    amount = float(input("Enter amount to withdraw: ₹"))

    if amount <= 0:
        print("Enter a valid positive amount!\n")
    elif amount > balance:
        print("Insufficient balance!\n")
    else:
        balance -= amount
        print(f"Withdrawal successful! You withdrew ₹{amount}")
        print(f"Remaining balance: ₹{balance}\n")
def deposit():
    global balance
    amount = float(input("Enter amount to deposit: ₹"))

    if amount <= 0:
        print("Enter a valid positive amount!\n")
    else:
        balance += amount
        print(f"Deposit successful! You deposited ₹{amount}")
        print(f"Updated balance: ₹{balance}\n")
def menu():
    print("===== ATM MENU =====")
    print("1. Check Balance")
    print("2. Withdraw Money")
    print("3. Deposit Money")
    print("4. Exit")

# Main Program
if login():
    while True:
        menu()
        choice = input("Enter your choice: ")

        if choice == "1":
            check_balance()
        elif choice == "2":
            withdraw()
        elif choice == "3":
            deposit()
        elif choice == "4":
            print("Thank you for using the ATM!")
            break
        else:
            print("Invalid option! Please try again.\n")
```

```
===== ATM LOGIN =====
Enter PIN:  1234
Login successful!

===== ATM MENU =====
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Enter your choice:  1
Your current balance is: ₹5000

===== ATM MENU =====
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Enter your choice:  2000000
Invalid option! Please try again.

===== ATM MENU =====
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Enter your choice:  3
Enter amount to deposit: ₹ 10000
Deposit successful! You deposited ₹10000.0
Updated balance: ₹15000.0

===== ATM MENU =====
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Enter your choice:  4
Thank you for using the ATM!
```

Testing Approach

Unit testing: Testing each module individually.

Integration testing: Checking interaction between ATM and server.

System testing: Complete ATM workflow testing.

Security testing: Testing PIN validation and encryption.

Performance testing: Checking response speed for each transaction.

User acceptance testing (UAT): Ensuring system is easy to use.

Challenges Faced

Ensuring secure communication between ATM and server.

Handling multiple error scenarios (invalid PIN, network issues).

Designing simple but effective UI screens.

Implementing database transactions safely.

Maintaining system performance under load.

Learning & Key Takeaways

Understanding of real-world banking systems.

Better knowledge of client-server architecture.

Improved problem-solving and debugging skills.

Hands-on experience with diagrams and requirement analysis.

Knowledge of secure authentication and database handling.

Future Enhancements

Add biometric authentication (fingerprint/face).

Support for cardless withdrawals using mobile OTP.

Add QR-based transactions.

Cloud-based ATM management.

Deposit automation and cheque deposit.

AI-based fraud detection.

References

Banking technology textbooks.

Research papers on ATM security.

Online documentation for Java/Python/MySQL.

Articles about ATM architecture and network security.

RBI guidelines for ATM operations.