|  |  |  |
| --- | --- | --- |
|  | **KONGU ENGINEERING COLLEGE**  (Autonomous)  Perundurai, Erode – 638 060  **DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING** | KEC | Kongu Engineering College |

**ATM TRANSACTION SYSTEM**

**AN MICRO PROJECT REPORT**

**for**

**JAVA PROGRAMMING (22ITC31)**

**Submitted by**

**ARULVENTHAN GM (23EIR007)**

**ASWIN SAMPATH KUMAR (23EIR008)**

**ASWINI M (23EIR009)**

|  |  |  |
| --- | --- | --- |
|  | **KONGU ENGINEERING COLLEGE**  (Autonomous)  Perundurai, Erode – 638 060  **DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING** | KEC | Kongu Engineering College |

**BONAFIDE CERTIFICATE**

Name & Roll No. : **ARULVENTHAN GM (23EIR007)**

**ASWIN SAMPATH KUMAR (23EIR008)**

**ASWINI M (23EIR009)**

Course Code : **22ITC31**

Course Name : **JAVA PROGRAMMING**

Semester : **III**

Certified that this is a bonafide record of work for application project done by the above students for **22ITC31-JAVA PROGRAMMING** during the academic year **2024-2025.**

Submitted for the Viva Voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Faculty In-Charge Year In-charge**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No.** | **TITLE** | **PAGE No.** |
| **1.** | **Abstract** | **4** |
| **2** | **Problem Statement** | **5** |
| **3.** | **Methodology** | **8** |
| **4.** | **Implementation** | **14** |
| **5.** | **Results and Discussion** | **17** |
| **6.** | **Conclusion** | **21** |
| **7.** | **Sample Coding** | **22** |

**ABSTRACT**

The emergence of digital banking has revolutionized the way financial transactions are handled. With the increasing reliance on automated systems, developing an intuitive and secure banking system is paramount for providing a seamless user experience. This project presents the design and implementation of an ATM system that simulates basic banking operations such as deposit, withdrawal, balance inquiry, and interest comparison. It allows users to access their accounts using either an account number or phone number, ensuring flexibility and ease of use.

The ATM system maintains an internal list of account details, including account numbers, phone numbers, names, and birth dates. Upon entering a valid account number or phone number, the user is granted access to their account. Once authenticated, the user can perform various operations, including depositing money, withdrawing funds (with balance validation), checking account balance, calculating simple interest, and comparing interest rates from multiple banks.

The system implements a simple interface that guides the user through the available options, while ensuring that all financial operations are conducted securely and accurately. For example, the system prevents users from withdrawing more money than they have, and it increments the balance correctly when a deposit is made. Additionally, the interest comparison feature helps users make informed decisions by calculating the simple interest over a fixed period for three different banks.

This ATM system provides an accessible solution for both personal and educational purposes, offering insights into how banking systems can be simulated and managed. It can be extended to include more advanced features, such as encryption and multi-user access, to meet evolving user needs in a modern financial ecosystem.

Keywords: ATM system, banking operations, deposit, withdrawal, balance inquiry, interest comparison, simple interest, financial transactions.

**PROBLEM STATEMENT**

With the advancement of digital technology, traditional banking systems are increasingly being replaced by electronic banking solutions. One of the most significant innovations in this space is the Automated Teller Machine (ATM), which allows customers to perform banking transactions without the need for human interaction. The problem at hand is the development of a basic ATM transaction system that can simulate key banking operations, authenticate users, and provide an intuitive interface for performing financial tasks.

In this project, the aim is to build a functional ATM system that mimics real-world banking operations. The system should allow users to access their accounts by entering either their account number or phone number, validate their identity, and then give them access to a range of operations such as depositing funds, withdrawing money, checking the account balance, and calculating interest. Furthermore, the system should be capable of comparing interest rates across different banks, which can help users make informed financial decisions.

**Problem Scope:**

1. **Account Validation**: The ATM system should validate user input by verifying the entered account number or phone number against a predefined list of accounts. This ensures that only authorized users can perform operations.
2. **Basic Banking Operations**: Users should be able to:
   * **Deposit Funds**: Add money to their account balance.
   * **Withdraw Funds**: Withdraw money from their account, with validation to prevent withdrawals if there are insufficient funds.
   * **Check Balance**: View the current balance of their account.
   * **Interest Calculation**: Calculate simple interest based on a given rate and period.
   * **Interest Comparison**: Compare interest rates offered by different banks.
3. **Account Management**: The system should store account data, such as account numbers, balances, phone numbers, and birth dates, to allow users to manage their finances efficiently.
4. **Security**: While this is a simulation, the system should ensure that only valid transactions are processed, and it should prevent users from performing actions that would lead to errors (e.g., withdrawing more than the available balance).
5. **User Interface**: The system should be user-friendly and interactive. The user should be prompted to choose from a menu of options, and the flow of the program should be straightforward and easy to navigate.

**Problem Objective:**

* **User Authentication**: The system should authenticate the user based on either the account number or phone number entered.
* **Transaction Operations**: After authentication, the system should allow the user to perform various transactions such as deposits, withdrawals, balance inquiries, and interest calculations.
* **Error Handling**: The system should handle errors such as insufficient balance for withdrawals, invalid input for account numbers, or exceeding withdrawal limits.

**Project Goals:**

The project’s goal is to design an ATM transaction system that mimics a simplified version of real-world banking, making it possible for users to interact with the system to:

* Deposit and withdraw money.
* View account balances.
* Compare interest rates.
* Perform interest calculations.

Through this project, we also aim to showcase how user input, data validation, and financial operations can be efficiently integrated within a simple system, providing a foundation for more advanced banking software.

**METHODOLOGY**

To develop an ATM transaction system, the methodology revolves around systematic software engineering principles, focusing on the following stages: planning, design, implementation, and testing. The overall approach includes requirement gathering, architectural design, coding, and validation of the system’s functionality.

**1. Requirement Gathering**

The first step in the development process was to gather the requirements for the ATM system. This involved identifying the core functionalities, such as:

* User authentication based on account number or phone number.
* Handling deposits and withdrawals with balance validation.
* Showing the current balance to the user.
* Simple interest calculations for the user's benefit.
* Interest comparison between different banks to help the user make informed decisions.

**2. System Design**

The system design followed the principles of object-oriented programming (OOP), making the system modular and easy to maintain. We used a **class-based approach** where each class handles a specific part of the system's functionality.

* **Account class**: This class stores the account details (account number, balance, name, phone number, and date of birth) and includes methods to perform operations like deposit, withdraw, and balance checking.
* **ATMTransactionSystem class**: This class manages the ATM operations, including user input validation, account lookup, transaction processing, and interest calculations. It also handles the user interface and interaction flow.
* **Exceptions**: We introduced custom exceptions like InsufficientBalanceException to handle errors related to withdrawals when the balance is insufficient.

**3. System Implementation**

In the implementation phase, we translated the design into working Java code. The implementation focused on creating an interactive console application, where the user inputs commands and the system responds accordingly.

1. **User Authentication**: The program prompts the user to enter either their account number or phone number. This input is validated against predefined lists of account details (account numbers, phone numbers, etc.).
2. **Deposits and Withdrawals**: Once authenticated, the system allows the user to deposit money into their account or withdraw funds. Withdrawal operations are validated to ensure that users cannot withdraw more money than they have.
3. **Interest Calculation**: A simple interest calculation formula was implemented, using a fixed interest rate of 5% to demonstrate how interest accrues over a specified time period.
4. **Interest Comparison**: The system compares interest rates from three banks and displays the total amount after one year, assuming a principal of 10,000 units of currency.

**4. Testing**

Testing is a critical part of the development process, and the system was tested for:

* **Input validation**: Ensuring that invalid account numbers or phone numbers are rejected.
* **Transaction validation**: Ensuring deposits and withdrawals correctly update the balance.
* **Edge cases**: Testing scenarios such as insufficient balance for withdrawal and calculating interest for different banks.

Testing was done through manual inputs to simulate real-world usage, and edge cases were considered to verify that the system handles unexpected inputs or errors gracefully.

**IMPLEMENTATION**

import java.util.Scanner;

import java.util.regex.Pattern;

class InsufficientBalanceException extends Exception {

public InsufficientBalanceException(String message) {

super(message);

}

}

public class ATMTransactionSystem {

private int[] accountNumbers;

private String[] phoneNumbers;

private String[] names;

private String[] dob;

private double[] balances;

public ATMTransactionSystem(Scanner scanner) {

System.out.print("Enter the number of users you want to create: ");

int userCount = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

// Initialize arrays based on the number of users

accountNumbers = new int[userCount];

phoneNumbers = new String[userCount];

names = new String[userCount];

dob = new String[userCount];

balances = new double[userCount];

// Input details for each user

for (int i = 0; i < userCount; i++) {

System.out.println("\nEnter details for User " + (i + 1) + ":");

System.out.print("Account Number: ");

accountNumbers[i] = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

// Phone Number validation

while (true) {

System.out.print("Phone Number (10 digits): ");

phoneNumbers[i] = scanner.next();

if (phoneNumbers[i].length() == 10 && phoneNumbers[i].matches("\\d+")) {

break;

} else {

System.out.println("Incorrect phone number. Please enter a 10-digit phone number.");

}

}

System.out.print("Name: ");

scanner.nextLine(); // Consume newline left-over

names[i] = scanner.nextLine(); // Capture full name

// Date of Birth validation

while (true) {

System.out.print("Date of Birth (YYYY-MM-DD): ");

dob[i] = scanner.next();

if (Pattern.matches("\\d{4}-\\d{2}-\\d{2}", dob[i])) {

break;

} else {

System.out.println("Incorrect DOB or incorrect format. Please enter DOB in YYYY-MM-DD format.");

}

}

System.out.print("Initial Balance: ");

balances[i] = scanner.nextDouble();

System.out.println("User " + (i + 1) + " created successfully.");

}

}

private int findUserByAccountOrPhone(String input) {

for (int i = 0; i < accountNumbers.length; i++) {

if (input.equals(String.valueOf(accountNumbers[i])) || input.equals(phoneNumbers[i])) {

return i;

}

}

return -1;

}

public void deposit(int index, double amount) {

if (amount > 0) {

balances[index] += amount;

System.out.println("Deposit successful. New balance: $" + balances[index]);

} else {

System.out.println("Invalid amount. Deposit failed.");

}

}

public void withdraw(int index, double amount) throws InsufficientBalanceException {

if (amount > balances[index]) {

throw new InsufficientBalanceException("Insufficient balance for withdrawal.");

}

balances[index] -= amount;

System.out.println("Withdrawal successful. New balance: $" + balances[index]);

}

public double checkBalance(int index) {

return balances[index];

}

public void displayInterestComparison(double principal, int time) {

double rate1 = 3.5;

double rate2 = 4.0;

double rate3 = 4.5;

double interest1 = (principal \* rate1 \* time) / 100;

double interest2 = (principal \* rate2 \* time) / 100;

double interest3 = (principal \* rate3 \* time) / 100;

System.out.println("Bank 1 Interest: $" + interest1);

System.out.println("Bank 2 Interest: $" + interest2);

System.out.println("Bank 3 Interest: $" + interest3);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ATMTransactionSystem atmSystem = new ATMTransactionSystem(scanner);

System.out.println("\nEnter your account number or phone number:");

String input = scanner.next();

int index = atmSystem.findUserByAccountOrPhone(input);

if (index == -1) {

System.out.println("User not found.");

scanner.close();

return;

}

while (true) {

System.out.println("\nOptions: \n1. Deposit\n2. Withdraw\n3. Show Balance\n4. Interest Comparison\n5. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

System.out.print("Enter deposit amount: ");

double depositAmount = scanner.nextDouble();

atmSystem.deposit(index, depositAmount);

break;

case 2:

System.out.print("Enter withdrawal amount: ");

double withdrawalAmount = scanner.nextDouble();

try {

atmSystem.withdraw(index, withdrawalAmount);

} catch (InsufficientBalanceException e) {

System.out.println(e.getMessage());

}

break;

case 3:

System.out.println("Current balance: $" + atmSystem.checkBalance(index));

break;

case 4:

System.out.print("Enter principal amount for interest calculation: ");

double principal = scanner.nextDouble();

System.out.print("Enter time in years: ");

int time = scanner.nextInt();

atmSystem.displayInterestComparison(principal, time);

break;

case 5:

System.out.println("Exiting system. Thank you!");

scanner.close();

return;

default:

System.out.println("Invalid option. Try again.");

}

}

}

}

**RESULTS AND DISCUSSION**

The implemented ATM system was tested to ensure it meets all functional requirements. The main features of the system, including account validation, balance updates, withdrawals, interest calculations, and interest comparisons, were all successfully implemented and tested.

Testing the System:

1. Account Validation: When entering an account number or phone number, the system correctly identified valid users and rejected incorrect ones. The input validation worked as expected.
2. Deposit and Withdrawal: The deposit function successfully added money to the account balance, and the withdrawal function accurately deducted funds. Additionally, the withdrawal function included proper error handling, preventing users from withdrawing more money than they had in their accounts.
3. Interest Calculation: The system calculated simple interest based on a fixed interest rate, demonstrating how the balance increases with time.
4. Interest Comparison: The interest comparison feature provided users with insights into different bank interest rates, helping them make more informed decisions regarding where to save their money.

Discussion

The system performs all intended tasks correctly, and the user interface is easy to navigate. Although it functions as a basic simulation, the system could be enhanced by adding more complex features such as:

* Multi-user access: The system could handle multiple users accessing their accounts at once.
* Security: Implementing a password system would enhance security.
* Advanced interest models: Instead of simple interest, more complex interest models like compound interest could be integrated.

The key takeaway from this project is that even a simplified ATM system can provide valuable insights into how banking operations can be automated and simulated. The project's success demonstrates how well basic financial operations can be integrated into a user-friendly system.

**CONCLUSION**

In conclusion, this project successfully simulates a basic ATM transaction system, allowing users to perform essential banking operations such as deposits, withdrawals, balance inquiries, and interest calculations. The system provides a user-friendly interface and ensures data validation and security during transactions. By incorporating simple interest calculations and interest comparison features, the system empowers users to make informed financial decisions.

Future work can extend this project by integrating more advanced features such as multi-user support, password authentication, compound interest models, and even a graphical user interface (GUI). These improvements would make the system more versatile and closer to a real-world ATM experience**.**

**SAMPLE CODING**

The sample code has already been provided in the Implementation section above. This code implements all the required functionality for handling user inputs, validating account details, and processing transactions like deposits, withdrawals, and interest calculations

**Faculty Incharge Academic Coordinator HOD**