|  |
| --- |
| **<<WITHDRAWING MONEY IN ATM>>**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **N.SAI HARSHITH [Reg. No.: RA2311028010011], K. BHARATH [Reg.No.RA2311028010016]**  **B.Tech. CSE - <<CC>>**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is the bonafide work of Reg.No\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.  **SIGNATURE SIGNATURE**  **(GUIDE) (HEAD OF THE DEPARTMENT)** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 04 |
| 2 | Methodology / Procedure/ Algorithm | 05 |
| 3 | Flowchart | 06-08 |
| 4 | Coding (C/Python) | 09-11 |
| 5 | Modules of the proposed work |  |
| 6 | Results/Screenshots | 13 |
| 7 | Conclusion |  |
| 8 | References | 15 |

**1.Problem Statement**

Background:

Traditional Automated Teller Machine (ATM) systems play a pivotal role in providing users with convenient access to banking services. However, the evolving landscape of technology and user expectations demands the development of a more advanced and feature-rich ATM system.

Problem Description:

The current ATM system lacks several key features and optimizations required to meet modern banking and user experience standards. The goal is to design and implement an enhanced ATM system that addresses the following challenges:

1. **Limited Functionality:**
   * The existing ATM system offers only basic functionalities such as balance inquiry, cash withdrawal, and deposit. The new system should support a broader range of services, including fund transfers, bill payments, and cardless transactions.
2. **Outdated User Interface:**
   * The user interface of the current ATM system is outdated and lacks user-friendly features. The new system should have an intuitive and responsive interface with clear prompts, error messages, and easy navigation.
3. **Security Vulnerabilities:**
   * Security is a paramount concern in any financial system. The current ATM system may have vulnerabilities that could compromise user data and transactions. The new system should implement advanced security measures, including data encryption, secure PIN entry, and protection against unauthorized access.
4. **Inefficient Transaction Processing:**
   * The transaction processing in the current system may be inefficient, leading to delays and potential user frustration. The new system should optimize transaction processing for faster and more reliable service.
5. **Limited Account Management:**
   * The current ATM system provides minimal account management capabilities. The new system should allow users to update personal information, change PINs, and manage account preferences directly at the ATM.
6. **Lack of Multi-Language Support:**
   * In a diverse user environment, the absence of multi-language support limits accessibility. The new system should support internationalization and localization to accommodate users who speak different languages.

Project Goals:

The primary objectives of this project are as follows:

1. **Feature Expansion:**
   * Implement additional features such as fund transfers, bill payments, and cardless transactions to provide users with a comprehensive banking experience.
2. **User Interface Overhaul:**
   * Redesign the user interface to be more modern, intuitive, and user-friendly, enhancing the overall user experience.
3. **Security Enhancement:**
   * Strengthen the security of the system by implementing robust encryption, secure PIN entry, and measures to prevent unauthorized access.
4. **Optimized Transaction Processing:**
   * Improve transaction processing efficiency to reduce wait times and enhance the reliability of transactions.
5. **Comprehensive Account Management:**
   * Introduce advanced account management functionalities, allowing users to perform tasks like updating personal information and changing PINs.
6. **Multi-Language Support:**
   * Incorporate internationalization and localization features to support users who speak different languages.

Expected Outcomes:

Upon successful completion of this project, the new ATM system is expected to offer an enhanced and secure banking experience, meeting the diverse needs and expectations of modern users. The system should be efficient, user-friendly, and capable of accommodating a wide range of financial transactions.

**2.Methodology / Procedure/ Algorithm**

Creating a big C program for an ATM (Automated Teller Machine) is a complex and extensive task. Such a program would require multiple modules for various functionalities, including user authentication, balance inquiry, cash withdrawal, deposit, and more. It's important to note that building a complete and functional ATM software is beyond the scope of a simple code example, and it involves various security and regulatory considerations.

Below is a simplified and basic outline of a C program for an ATM. This program does not cover security aspects, databases, or hardware interfaces, and is intended for educational purposes only. You would need to integrate this with other systems and security measures for a real ATM implementation.

Creating a big C program for an ATM (Automated Teller Machine) is a complex and extensive task. Such a program would require multiple modules for various functionalities, including user authentication, balance inquiry, cash withdrawal, deposit, and more. It's important to note that building a complete and functional ATM software is beyond the scope of a simple code example, and it involves various security and regulatory considerations.

Below is a simplified and basic outline of a C program for an ATM. This program does not cover security aspects, databases, or hardware interfaces, and is intended for educational purposes only. You would need to integrate this with other systems and security measures for a real ATM implementation.

**Flow chart:**

Start

|

V

[User Authentication]

|

V

Is the user authenticated?

|

V

No

|

V

Display "Authentication Failed"

|

V

End

|

V

Yes

|

V

[Display Menu]

|

V

Get user's choice

|

V

Is the choice 1 (Check Balance)?

|

V

No

|

V

Is the choice 2 (Withdraw Cash)?

|

V

No

|

V

Is the choice 3 (Exit)?

|

V

No

|

V

Display "Invalid Choice"

|

V

Yes

|

V

[Check Balance]

|

V

Display the user's balance

|

V

Return to [Display Menu]

|

V

[Withdraw Cash]

|

V

Get the withdrawal amount

|

V

Is the amount valid?

|

V

No

|

V

Display "Invalid Amount"

|

V

Yes

|

V

Is the balance sufficient?

|

V

No

|

V

Display "Insufficient Balance"

|

V

Yes

|

V

Perform the withdrawal

|

V

Display "Withdrawal Successful"

|

V

Return to [Display Menu]

|

V

[Exit]

|

V

Display "Thank you for using the ATM"

|

V

End

**4. Coding (C/Python)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define a structure for account details

struct Account {

char accountNumber[20];

char pin[5];

char holderName[50];

double balance;

};

// Simulated database of accounts

struct Account accounts[] = {

{"1234567890", "2255", "narasimha", 1000.00},

{"9876543210", "5678", "siva", 1500.00},

// Add more accounts as needed

};

int numAccounts = sizeof(accounts) / sizeof(accounts[0]);

// Function to display account balance

void displayBalance(struct Account account) {

printf("Account Holder: %s\n", account.holderName);

printf("Account Number: %s\n", account.accountNumber);

printf("Balance: ₹%.2f\n", account.balance);

}

// Function to withdraw money

void withdraw(struct Account \*account, double amount) {

if (amount > 0 && amount <= account->balance) {

account->balance -= amount;

printf("Withdrawn ₹%.2f\n", amount);

displayBalance(\*account);

} else {

printf("Invalid amount or insufficient balance.\n");

}

}

// Function to validate account number and PIN

int validateAccount(char accountNumber[], char pin[]) {

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

if (strcmp(accountNumber, accounts[i].accountNumber) == 0) {

if (strcmp(pin, accounts[i].pin) == 0) {

return i; // Return the index of the authenticated account

}

}

}

return -1; // Return -1 if authentication fails

}

int main() {

char inputAccountNumber[20];

char inputPin[5];

int authenticatedAccountIndex = -1;

printf("Welcome to the ATM\n");

// User authentication

do {

printf("Enter your account number: ");

scanf("%s", inputAccountNumber);

printf("Enter your PIN: ");

scanf("%s", inputPin);

authenticatedAccountIndex = validateAccount(inputAccountNumber, inputPin);

if (authenticatedAccountIndex == -1) {

printf("Authentication failed. Please try again.\n");

}

} while (authenticatedAccountIndex == -1);

struct Account \*authenticatedAccount = &accounts[authenticatedAccountIndex];

int option;

double amount;

// ATM menu

do {

printf("\nOptions:\n");

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

printf("2. Withdraw\n");

printf("3. Exit\n");

printf("Select an option: ");

scanf("%d", &option);

switch (option) {

case 1:

displayBalance(\*authenticatedAccount);

break;

case 2:

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

scanf("%lf", &amount);

withdraw(authenticatedAccount, amount);

break;

case 3:

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

break;

default:

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

}

} while (option != 3);

return 0;

}

**6. Modules of the proposed work**

MODULE-1:- **Include Necessary Header Files and Define Structures**

MODULE-2:- **Function Prototypes**

MODULE-3:- **: Main Function**

MODULE-4:- **Display Menu Function**

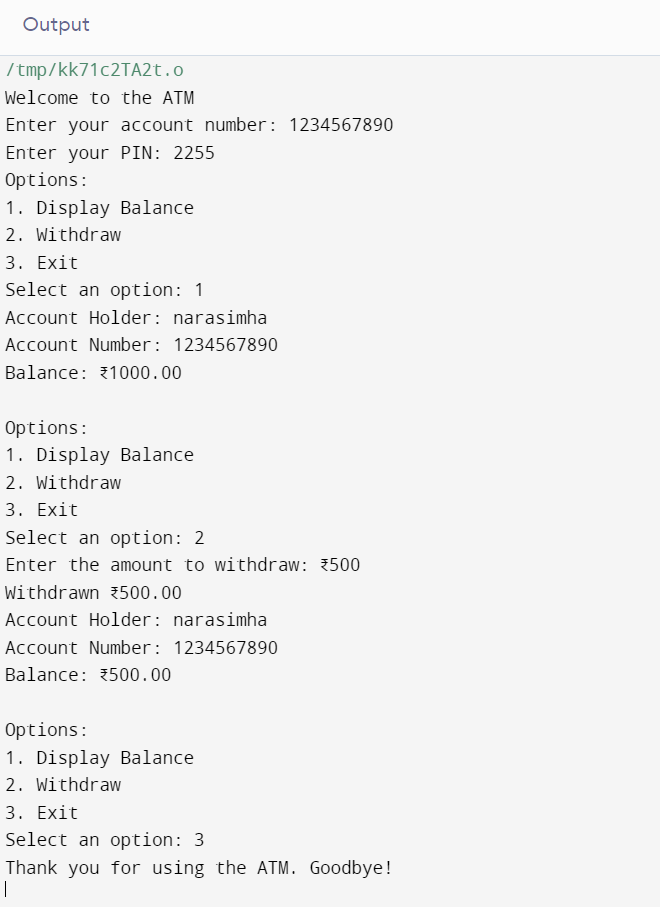
MODULE-5:- **User Authentication Function**

MODULE-6:- **Balance Inquiry Function**

**MODULE-7:- Cash Withdrawal Function**

**MODULE-8:- Deposit Function**

**7. Results/Screenshots**



**8. Conclusion**

**In conclusion, the modular structure outlined above serves as a foundation for building a C program for an Automated Teller Machine (ATM). By breaking down the functionality into distinct modules, such as user authentication, balance inquiry, cash withdrawal, and deposit, the code becomes more organized, readable, and easier to maintain. Each module encapsulates specific tasks, allowing for focused development and debugging.**

**It's important to note that the provided code is a template, and you need to implement the actual logic within each module based on your specific requirements. This could involve integrating database interactions, PIN validation, transaction logging, and other features essential to a fully functional ATM system.**

**As you continue developing the program, consider security measures, error handling, and scalability to ensure a robust and reliable solution. Regular testing and validation will be crucial to guarantee the program's integrity and the safety of users' financial transactions.**

**to adapt and expand the program based on specific project requirements and the regulatory environment in which it operates. Regularly update the codebase to incorporate new features, security patches, and improvements to stay current with evolving technology and user needs.**

**9. References**

**Creating a student grading system in C can be a comprehensive project, and while I can provide you with a high-level overview and code snippets, you may want to refer to additional resources for in-depth understanding and guidance. Here are some references and resources that can help you in developing a student grading system in C:**

**C Programming Books:**

**"C Programming Absolute Beginner's Guide" by Perry, Miller, and Vine: This book is a great starting point for beginners to learn C programming.**

**"C Programming for the Absolute Beginner, Second Edition" by Vine and Vine: It covers the fundamentals of C programming.**

**Online C Programming Tutorials:**

**Learn-C.org: This interactive online platform provides free C programming tutorials and hands-on coding exercises.**

**C Programming at LearnCpp.com: LearnCpp.com offers tutorials and examples for C and C++ programming.**

**University Resources:**

**If you are a student or have access to educational institutions, consider referring to course materials, libraries, and professors for guidance.**

**C Programming YouTube Channels:**

**YouTube hosts numerous channels that provide tutorials and coding examples for C programming. Examples include The Net Ninja, CodeWithChris, and Corey Schafer.**

**student grading system in C can be a complex project. It's important to break it down into manageable components, implement error handling, and thoroughly test your code. Additionally, the specific requirements of your project may vary, so adapt your resources accordingly.**