**BANK ACCOUNT MANAGEMENT SYSTEM**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**G.ANURADHA-24KB1A05H7**

**G.SUSHMA-24KB1A05J1**

**G.VYSHNAVI-24KB1A05J7**

**J.GOPIKA-24KB1A05L2**

**Under the Guidance of**

**Smt.B.SRUTHI**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NBKR INSTITUTE OF SCIENCE &TECHNOLOGY**

**VIDYANAGAR-524413**

**LIBRARY MANAGEMENT SYSTEM**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**G.ANURADHA-24KB1A05H7**

**G.SUSHMA-24KB1A05J1**

**G.VYSHNAVI-24KB1A05J7**

**J.GOPIKA-24KB1A05L2**

Under the Guidance of

**SHIVAN RAJ**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NBKR INSTITUTE OF SCIENCE &TECHNOLOGY**

**VIDYANAGAR-524413**

**NBKR INSTITUTE OF SCIENCE &TECHNOLOGY**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project report entitled LIBRARY MANAGEMENT SYSTEM being submitted by

**G.ANURADHA-24KB1A05H7**

**G.SUSHMA-24KB1A05J1**

**G.VYSHNAVI-24KB1A05J7**

**J.GOPIKA-24KB1A05L2**

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the **NBKR INSTITUTE OF SCIENCE & TECHNOLOGY ,**is a record of bonafied work carried out under my guidance and supervision.

|  |  |
| --- | --- |
| **Smt.B.SRUTHI** | **Dr.A.Rajasekhar Reddy**  **HOD**  **Dept.of CSE** |

**DECLARATION**

I hereby declare that the dissertation entitled **Library Management System** submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

G.Anuradha-24KB1A05H7

Place:Vidyanagar G.Sushma -24KB1A05J1

Date: 05-05-2025 G.Vyshnavi -24KB1A05j7

J.Gopika-24KB1A05L2

## ACKNOWLEDGEMENT

**Isincerely thank my faculty guide, Ma. B. Shrmthi, for her guidance and support throughout my project, "Bank account management system".I also thank Dr. A, Raja Sekhar Redldly, Head of Department, for providing the necessary resources.**

**I also extend my thanks to my college for providing the necessary resources and environment to work on this project effectively.**

**I am especially grateful to my family and friends for their unwavering support and motivation during the course of this project.**

**Finally, I would like to acknowledge all those who directly or indirectly contributed to the completion of this project. Their help and cooperation are deeply appreciated. encouragement.**

**This project enhanced my knowledge of C Programning .And it also improves my knowledge,and skills.**

**This project has been a great learning experience, helping me strengthen my knowledge in software development and real-world application design**

# ABSTRACT

**The Bank Account Management System is a software application designed to simplify and streamline the process of managing bank accounts. This system allows users to perform essential banking operations such as creating new accounts, viewing account details, depositing and withdrawing funds, and maintaining transaction history.**

**Developed with a user-friendly interface, the system ensures secure and efficient handling of customer data and transactions. It is intended to reduce the manual effort and paperwork involved in traditional banking systems by automating routine tasks.**

**The project also incorporates basic validation and error-handling mechanisms to ensure data integrity and security. It serves as a practical tool for learning core concepts of database management, object-oriented programming, and real-world financial operations in a simplified environment.**

**This system is suitable for educational purposes and can be extended for small-scale banking institutions with further enhancements.**

# **1.INTRODUCTION**

This project, titled "Bank Account Management System", aims to provide a software solution for managing bank accounts digitally. It is designed to handle common banking operations such as adding customer details, performing deposits and withdrawals, and displaying account information along with a transaction history. The systereduces the burden of manual work and ensures data consistency and security.

The project provides an interactive, menu-driven interface developed in the C programming language. It utilizes structures and linked lists to manage customer information and transaction history efficiently.

**PROBLEM STATEMENT:**

Managing bank accounts manually is slow, prone to errors, and inefficient. This project aims to develop a Bank Account Management System that automates tasks like account

creation,deposits,withdrawals, and record management, ensuring faster, more accurate, and secure banking operations.

**Objective**

The primary objective of the Bank Account Management System is to develop a reliable, secure, and efficient software application that facilitates the management of customer bank accounts. The system is designed to automate various banking operations such as creating new accounts, handling deposits and withdrawals, updating account information, and generating transaction histories. By streamlining these processes, the system aims to minimize manual workload, reduce errors, ensure data consistency, and improve overall service delivery within a banking environment.

**Scope**

The scope of the Bank Account Management System encompasses the development and implementation of a software application that enables efficient handling of essential banking operations. The system is intended for use by bank staff and administrators to manage customer accounts securely and accurately. Key functionalities include:

* Creating and managing customer bank accounts
* Performing transactions such as deposits, withdrawals, and transfers
* Viewing and generating account statements and transaction history
* Updating customer and account details
* Ensuring data integrity and security through authentication and authorization mechanisms

This project is limited to core banking functionalities and does not cover advanced features such as loan management, credit scoring, or integration with external financial systems. The system will be designed as a standalone or internal tool and is not intended for public online banking use.

**2.LITERATURE SURVEY**

The development of a Bank Account Management System draws upon various existing technologies, systems, and research in the field of banking software and database management. A review of related work and existing systems highlights the need for secure, reliable, and user-friendly platforms to manage banking operations efficiently.

**Traditional Banking Systems**  
Earlier banking systems were largely manual, relying on paper records and human labor for managing accounts and transactions. This often resulted in delays, errors, and increased operational costs. The need for automation and better record-keeping led to the evolution of computerized systems.

**Core Banking Solutions (CBS)**  
Modern banks use core banking solutions that allow centralized management of customer accounts across different branches. These systems provide real-time processing, integrated modules for different services, and support for multiple users. However, these systems are often complex and costly to implement, making simpler management systems ideal for smaller banks or educational projects.

**Database-Driven Applications**  
Several academic and industry projects have focused on building bank management systems using relational databases and programming languages such as Java, Python, or C#. These systems emphasize data integrity, transaction security, and CRUD (Create, Read, Update, Delete) operations.

**Security and Authentication**  
Literature on financial software emphasizes the importance of authentication mechanisms, role-based access control, and encryption to prevent unauthorized access and ensure data confidentiality. Various algorithms and protocols are implemented in modern systems to address these security concerns.

**User Interface Design**  
Usability studies highlight the importance of intuitive user interfaces for efficient operation by bank staff. Good interface design reduces training time, minimizes errors, and improves overall system adoption.

This literature survey demonstrates that while comprehensive banking systems are widely available, there is a significant opportunity to design simplified, focused systems tailored to specific needs such as educational use or internal bank operations. This project aims to incorporate essential functionalities of modern banking systems while maintaining simplicity, security, and ease of use.

**3.Software Requirement Analysis:**

**Functional Requirements:**

-Create, update, and delete accounts.

-Deposit and withdraw funds.

-View transaction history.

**Nonfunctional Requirements:**

-Security (password protection, encryption).

-Performance (quick response time).

-Usability (user-friendly interface)

**4.Software Design:**

Program Start: main() calls the menu() function.

Menu Options Loop: User chooses an option:

→ View Account

→ Deposit / Withdraw

→ Transfer Funds

→ View Transactions

→ Logout

Add Customer: Inputs account number and name, initializes balance and transaction history.

Deposit: Finds customer, adds amount to balance, logs transaction.

Withdraw: Finds customer, checks balance, deducts amount if sufficient, logs transaction.

View Account: Displays account info and transaction history.

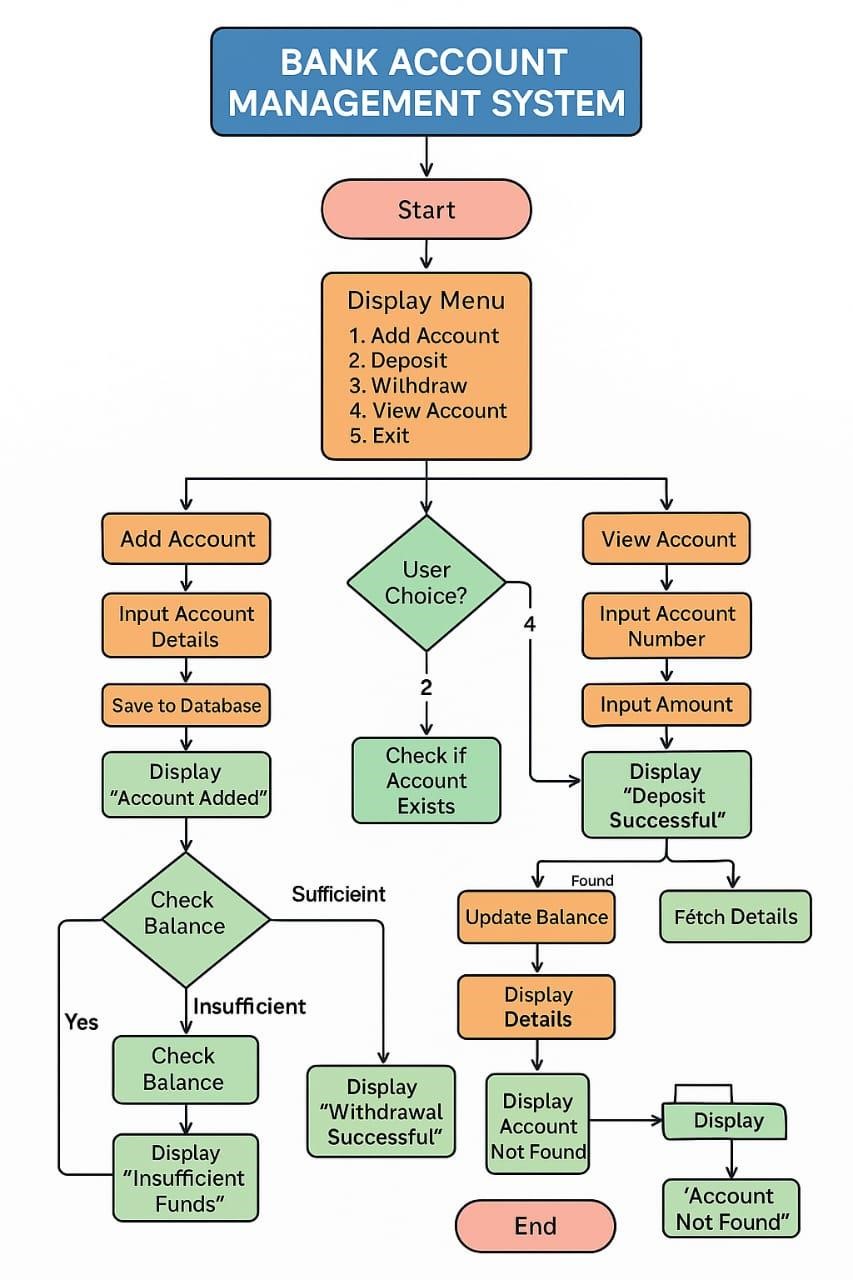
Exit: Terminates the program.

Helper Functions:

findCustomer() searches for a customer by account number.

addTransaction() and createTransaction() handle transaction records using linked lists

**FLOW CHART:**



**5.Proposed System**:

Customer Management Module

Adds new customers.

Stores customer details (account number, name, balance).

Transaction Management Module:

Records deposit and withdrawal transactions.

Maintains each customer's transaction history.

Deposit Module:

Handles money deposits.

Updates customer balance and logs the transaction.

Withdrawal Module:

Processes money withdrawals.

Checks sufficient balance and logs the transaction.

Account Viewing Module:

Displays customer information and transaction history.

User Interface Module:

Provides menu-driven interaction.

Directs user to appropriate operations

Working Flow:

Start Program → Show Menu → [1] Add Customer → [2] Deposit → [3] Withdraw → [4] View Details → [5] Exit → Perform Operation → Loop back to Menu → End.

**6.CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CUSTOMERS 100

// Structure for Transaction Node typedef struct Transaction { char type[10]; // "Deposit" or "Withdraw" float amount; struct Transaction \*next;

} Transaction;

// Structure for Customer Account typedef struct { int accountNumber; char name[50]; float balance;

Transaction \*transactions;

} Customer;

// Array of customers

Customer customers[MAX\_CUSTOMERS];

int customerCount = 0;

// Function to create a new transaction node

Transaction\* createTransaction(const char \*type, float amount) {

Transaction \*newTransaction = (Transaction \*)malloc(sizeof(Transaction)); strcpy(newTransaction->type, type); newTransaction->amount = amount; newTransaction->next = NULL; return newTransaction;

}

// Function to add a transaction to customer's linked list void addTransaction(Customer \*customer, const char \*type, float amount) { Transaction \*newTransaction = createTransaction(type, amount); newTransaction->next = customer->transactions; customer->transactions = newTransaction;

}

// Function to add a new customer void addCustomer() { if (customerCount >= MAX\_CUSTOMERS) {

printf("Customer limit reached!\n");

return;

}

Customer newCustomer; printf("Enter account number: "); scanf("%d", &newCustomer.accountNumber); printf("Enter name: "); scanf(" %[^\n]", newCustomer.name); newCustomer.balance = 0.0; newCustomer.transactions = NULL;

customers[customerCount++] = newCustomer;

printf("Customer added successfully!\n");

}

// Function to find customer by account number Customer\* findCustomer(int accountNumber) { for (int i = 0; i < customerCount; i++) { if (customers[i].accountNumber == accountNumber) return &customers[i];

}

return NULL;

}

// Function to deposit money void deposit() { int accNum; float amount; printf("Enter account number: "); scanf("%d", &accNum);

Customer \*cust = findCustomer(accNum);

if (cust == NULL) { printf("Account not found!\n"); return;

}

printf("Enter amount to deposit: "); scanf("%f", &amount);

cust->balance += amount; addTransaction(cust, "Deposit", amount);

printf("Deposit successful. New balance: %.2f\n", cust->balance);

}

// Function to withdraw money void withdraw() { int accNum; float amount; printf("Enter account number: "); scanf("%d", &accNum);

Customer \*cust = findCustomer(accNum); if (cust == NULL) { printf("Account not found!\n"); return; } printf("Enter amount to withdraw: "); scanf("%f", &amount);

if (cust->balance < amount) { printf("Insufficient balance!\n"); return;

}

cust->balance -= amount; addTransaction(cust, "Withdraw", amount);

printf("Withdrawal successful. New balance: %.2f\n", cust->balance);

}

// Function to display account and transaction history void viewAccount() { int accNum; printf("Enter account number: "); scanf("%d", &accNum);

Customer \*cust = findCustomer(accNum); if (cust == NULL) { printf("Account not found!\n"); return;

}

printf("\nAccount Number: %d\n", cust->accountNumber); printf("Name: %s\n", cust->name); printf("Balance: %.2f\n", cust->balance); printf("\nTransaction History:\n"); Transaction \*temp = cust->transactions; if (!temp) { printf("No transactions yet.\n");

} else { while (temp) { printf("%s of %.2f\n", temp->type, temp->amount); temp = temp->next;

} } printf("\n");

}

// Main Menu void menu() {

int choice; do { printf("\n--- Bank Account Management ---\n"); printf("1. Add Customer\n"); printf("2. Deposit Money\n"); printf("3. Withdraw Money\n"); printf("4. View Account Details\n"); printf("5. Exit\n"); printf("Enter your choice: "); scanf("%d", &choice);

switch (choice) { case 1: addCustomer(); break; case 2: deposit(); break; case 3: withdraw(); break; case 4: viewAccount(); break; case 5: printf("Exiting...\n"); break; default: printf("Invalid choice! Try again.\n");

}

} while (choice != 5);

}

// Main function int main() { menu(); return 0;

}

**7.OUTPUT**

-- Bank Account Management ---

1. Add Customer

2.Deposit Money

1. Withdraw Money
2. View Account Details
3. Exit

Enter your choice:1

Enter account number:112200

Enter name:anu

Customer added successfully!

--- Bank Account Management --

1. Add Customer

2.Deposit Money

3.Withdraw Money

4. View Account Details

05. Exit

Enter your choice:2

Enter account number:112200

Enter amount to deposit:10000

Deposit successful. New balance:10000.00

- -- Bank Account Management ---

1. Add Customer

2.Deposit Money

3.Withdraw Money

4. View Account Details 5.Exit

Enter your choice:4

Enter account number:112200

Account Number:112200

Vame:anu Balance：10000.00

Transaction History:

Deposit of 10000.0.

**8.Conclusion and Future Work**

**Conclusion**

The Bank Account Management System successfully addresses the need for a simplified, secure, and efficient solution to handle basic banking operations such as account creation, deposits, withdrawals, and transaction tracking. By automating these tasks, the system minimizes human error, reduces processing time, and enhances data accuracy and security. The application provides an intuitive user interface and reliable backend operations, making it suitable for use by bank employees or as a learning tool in educational environments. The project demonstrates the practical application of database management and software development concepts in solving real-world problems.

**Future Work**

While the current system meets the core requirements, several enhancements can be considered for future development:

* **Integration with online banking portals** to allow customer access and remote transactions
* **Implementation of advanced security features** such as two-factor authentication and data encryption
* **Support for additional banking services** such as loan processing, fixed deposits, and credit management
* **Mobile application development** for greater accessibility and usability
* **Comprehensive audit and logging features** to ensure accountability and traceability of operations
* **Multilingual and accessibility support** to cater to a broader user base

These future improvements would make the system more robust, scalable, and applicable to real-world banking environments with evolving user expectations.