VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**



**Assignment Report**

**On**

**“BANKING MANAGEMENT SYSTEM”**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DATA STRUCTURES AND APPLICATIONS (BCS304) COURSE OF III SEMESTER**

Submitted by

|  |  |  |
| --- | --- | --- |
| **MOHAMMED UZAIR** |  |  |
| **[1CG23CS062]** |  |  |

**Guide: HOD:**

**Prof. Asif Ulla Khan .M. Tech. Dr. Shantala C P PhD.,**

Asst. Prof., Dept. of CSE Head, Dept. of CSE

CIT, Gubbi. CIT, Gubbi.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**Channabasaveshwara Institute of Technology**

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(**NAAC Accredited & ISO 9001:2015 Certified Institution)**

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

**2024-25**

**Rubric – B.E. Mini-Project [BCS304]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course**  **outcome** | **Rubric/Level** | **Excellent**  **(91-100%)** | **Good**  **(81-90%)** | **Average**  **(61-80%)** | **Moderate**  **(40-60%)** | **Score** |
| **CO1** | **Identification of project**  **proposal**  **(05 Marks)** |  |  |  |  |  |
| **CO2** | **Design and**  **Implementation**  **(10 Marks)** |  |  |  |  |  |
| **CO3** | **Presentation skill**  **(05 Marks)** |  |  |  |  |  |
| **CO4** | **Report**  **(05 Marks)** |  |  |  |  |  |
| **Total** | | | | | |  |

**Course outcome:**

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Understanding overall project progress and performance.**

**Student Signature Faculty signature**

**Abstract**

The Banking Management System is a software project aimed at simulating core banking operations using efficient data structures and modular programming techniques. This system provides essential functionalities such as account creation, deposits, withdrawals, balance inquiries, and transaction history management. It leverages dictionaries for account storage and lists for maintaining transaction histories, ensuring efficient data retrieval and scalability.

The system is implemented with a user-friendly, menu-driven interface that guides users through various operations. Each account is managed as an object, encapsulating data and operations for clarity and reusability. This project demonstrates practical applications of object-oriented programming and data structures, serving as a foundation for more complex financial management systems.

Through this system, users can securely and efficiently manage multiple accounts, perform transactions, and track their financial activity. The project lays the groundwork for future enhancements, such as persistent data storage, fund transfers, and authentication mechanisms. This abstract highlights the educational and practical significance of the Banking Management System, showcasing its ability to simulate real-world banking functionalities in a simplified manner.

**CHAPTER-1**

##### Introduction :Banking Management System

##### **A Banking Management System is a software program that manages bank accounts and performs operations like deposits, withdrawals, balance inquiries, and transaction history tracking. It simulates the basic functionalities of a banking system, providing an easy and efficient way to manage customer accounts and transaction**

##### **The system ensures efficient management of accounts and transactions using appropriate data structures and a modular approach. This project is particularly useful for learning data structure concepts, object-oriented programming, and implementing real-world scenarios in software development**

##### **This project is ideal for learning data structures and algorithms as it involves real-world problem-solving and uses key programming concepts like classes, functions, and collections**

##### **The main goal is to design a user-friendly banking system with:**

## **1. Account management (creation, deposit, withdrawal).**

## **2. Transaction tracking.**

## **3. Displaying account balance and history.**

## **4. An efficient backend using appropriate data structures.**

## **This program uses the following concepts:**

## **Hash Table or Database:**

## **• For larger systems, dictionaries can be replaced with hash tables or databases for persistent storage.**

## **2. Queue for Transactions:**

## **• Instead of a list, a queue can be used if transaction history is only processed sequentially.**

## **3. Tree for Sorting:**

## **• A binary search tree can store accounts to enable sorted traversal (e.g., by account balance).**

## CHAPTER-2

##### Problem Statement

**Design and implement a Banking Management System that allows users to:**

**1. Create and manage bank accounts with unique account numbers.**

**2. Perform core banking operations such as:**

**a).Deposits.**

**b).Withdrawals (with balance validation).**

**c).Balance inquiries.**

**3. Maintain transaction histories for each account in chronological order.**

**4. Provide a user-friendly interface for interaction.**

**The system should ensure:**

**1. Data integrity during transactions.**

**2. Scalability for managing multiple accounts.**

**3. Efficient retrieval and processing of account data.**

**Additional functionality (optional):**

**Support for transferring funds between accounts.**

**Persistent data storage to save account and transaction information across session**

**CHAPTER-3**   ***Implementation***

System Design:

1. Account Class: Encapsulates account-related data and operations like deposits, withdrawals, and transaction history management.

2. Dictionary: Used to store multiple accounts, identified by unique account numbers for efficient access.

3. Menu-Driven Interface: Allows users to interact with the system via a set of options.

**Code Implementation:**

class Account:

    """Class to represent a bank account."""

    def \_\_init\_\_(self, account\_number, holder\_name, initial\_balance=0):

        self.account\_number = account\_number

        self.holder\_name = holder\_name

        self.balance = initial\_balance

        self.transactions = []  # List to store transaction history

    def deposit(self, amount):

        """Deposit money into the account."""

        if amount > 0:

            self.balance += amount

            self.transactions.append(f"Deposited: {amount}")

            print(f"Successfully deposited {amount}. New balance: {self.balance}")

        else:

            print("Deposit amount must be positive.")

    def withdraw(self, amount):

        """Withdraw money from the account."""

        if amount > 0:

            if self.balance >= amount:

                self.balance -= amount

                self.transactions.append(f"Withdrew: {amount}")

                print(f"Successfully withdrew {amount}. New balance: {self.balance}")

            else:

                print("Insufficient balance.")

        else:

            print("Withdrawal amount must be positive.")

    def check\_balance(self):

        """Check the current balance."""

        print(f"Account Balance: {self.balance}")

    def show\_transactions(self):

        """Display the transaction history."""

        if self.transactions:

            print("Transaction History:")

            for transaction in self.transactions:

                print(transaction)

        else:

            print("No transactions yet.")

# Dictionary to store accounts

accounts = {}

def create\_account():

    """Create a new account."""

    account\_number = input("Enter account number: ")

    if account\_number in accounts:

        print("Account number already exists.")

        return

    holder\_name = input("Enter account holder name: ")

    initial\_balance = float(input("Enter initial balance: "))

    accounts[account\_number] = Account(account\_number, holder\_name, initial\_balance)

    print(f"Account {account\_number} created successfully.")

def deposit\_money():

    """Deposit money into an account."""

    account\_number = input("Enter account number: ")

    if account\_number in accounts:

        amount = float(input("Enter amount to deposit: "))

        accounts[account\_number].deposit(amount)

    else:

        print("Account not found.")

def withdraw\_money():

    """Withdraw money from an account."""

    account\_number = input("Enter account number: ")

    if account\_number in accounts:

        amount = float(input("Enter amount to withdraw: "))

        accounts[account\_number].withdraw(amount)

    else:

        print("Account not found.")

def check\_account\_balance():

    """Check the balance of an account."""

    account\_number = input("Enter account number: ")

    if account\_number in accounts:

        accounts[account\_number].check\_balance()

    else:

        print("Account not found.")

def view\_transactions():

    """View the transaction history of an account."""

    account\_number = input("Enter account number: ")

    if account\_number in accounts:

        accounts[account\_number].show\_transactions()

    else:

        print("Account not found.")

def main():

    """Main function to run the banking management system."""

    while True:

        print("\n--- Banking Management System ---")

        print("1. Create Account")

        print("2. Deposit Money")

        print("3. Withdraw Money")

        print("4. Check Balance")

        print("5. View Transactions")

        print("6. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            create\_account()

        elif choice == "2":

            deposit\_money()

        elif choice == "3":

            withdraw\_money()

        elif choice == "4":

            check\_account\_balance()

        elif choice == "5":

            view\_transactions()

        elif choice == "6":

            print("Exiting the system. Goodbye!")

            break

        else:

            print("Invalid choice. Please try again.")

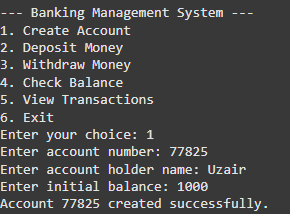
# Run the program

if \_\_name\_\_ == "\_\_main\_\_":

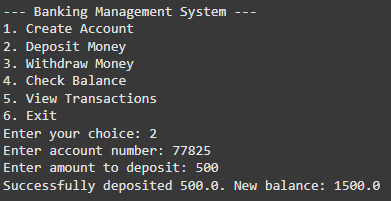
    main()

**CHAPTER-4 *Results***

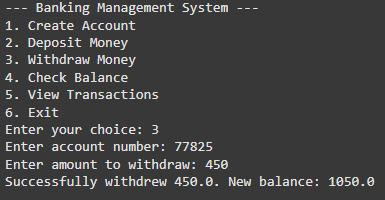
**1.Account Creation:**

****

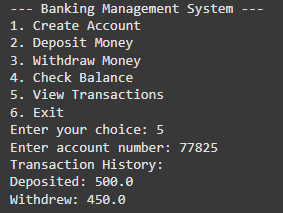
**2.Deposit Money:**

****

**3.Withdraw Money:**

****

**4.View Transaction:**

****

**CHAPTER-5** Conclusion

The Banking Management System successfully simulates essential banking operations like account creation, deposits, withdrawals, balance inquiries, and transaction tracking. Using data structures like dictionaries and lists ensures efficient management of accounts and transactions. The modular design of the system makes it extensible and easy to maintain. This project provides a solid foundation for implementing real-world banking systems and demonstrates the practical use of programming concepts like object-oriented design and data structures.

Future enhancements could include:

1. Persistent data storage for maintaining data across sessions.

2. Adding authentication for account security.

1. Expanding features to include fund transfers and admin monitoring.

**REFERENCE**

>Chatgpt

>GeeksforGeeks