**Report on Bank Management System**

**Abstract**

The bank management system is an application for maintaining a person's account in a bank. In this project we tried to show the working of a banking account system and cover the basic functionality of a bank account management system. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also, to enable the user’s workspace to have additional functionalities which are not provided under a conventional banking project. The bank account management system undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for bank account management system. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project is developed using c++ programming language. Creating and managing requirements is a challenge of it, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer including deposit, withdrawal etc. The project analyzes the system requirements and then comes up with the requirements specifications. It studies other related systems and then come up with system specifications. The system is then designed in accordance with specifications to satisfy the requirements. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system.

**Objectives**

The objective of this project is developing a bank customer management system to the best satisfaction of the customer and for profit maximization to the Banks.

The objectives are:

1. To create a banking system that is easily via internet.
2. To provide an easy way for customer to create account.
3. To make an easy withdrawal.
4. To make deposit easy.
5. To update the time this increases its performance.
6. Reduce the time wasted in going to banks to update personal details.
7. To make a menu driven system for user.

**Functional components of project**

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate; you can make appropriate assumptions and proceed. The System should maintain the following information about customer in a file Customer information which should contain the following information.

* Customer Name
* Account number
* bank balance
* Date of account creation.
* Last transaction• Account Type [Saving/ Current]

**Scope of the project**

The scope of this project is limited to some activities of the operations unit of a banking system which include opening of account, deposit of funds, withdrawal of funds and to know user account information. This application does not focus on other online services such as; purchasing items online, bill payment loan application etc.

**Menu of the program**

In this we use features of Object Oriented Programming which are:

* Classes
* Inheritance
* Composition

Menu of the program is given below:

* To create account.
* To make transactions
* To deposit
* To check balance
* To go to the main menu
* To exit

**Code**

#include<iostream.h>

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<time.h>

#include<dos.h>

class Account{

protected:

char name[32];

char account\_no[20];

float balance;

char\* createdAt;

char\* lastTransaction;

public:

///// Menus////////

//int mainMenu();

Account(){}

~Account();

void print()

{

// createAccount

cout<<"\nName : "<<name;

cout<<"\nAccount No. : "<<account\_no;

cout<<"\nBalance : "<<balance;

cout<<"\nCreate At : "<<createdAt;

cout<<"\nLast Transaction : "<<lastTransaction<<endl;

}

char\* getAccount\_no()

{

return account\_no;

}

char\* getName()

{

return name;

}

int getBalance()

{

return balance;

}

void createAccount();

//int transactionMenu();

///////// Transaction /////////

void withDrawal();

void deposit();

//void checkBalance();

};

class SavingAccount: public Account

{

float profit;

public:

SavingAccount(){}

void Apply\_Profit()

{

profit = ((2\*100)/balance);

balance = balance + profit;

}

void print()

{

Account::print();

cout<<"\n Last Profit : "<<profit;

}

};

class User{

// char name[50];

SavingAccount a2;

public:

User(){}

char\* getName()

{

return a2.getName();

}

char\* getAccount\_no()

{

return a2.getAccount\_no();

}

int getBalance()

{

return a2.getBalance();

}

void createAccount()

{

a2.createAccount();

}

void deposit()

{

a2.deposit();

}

void withDrawal()

{

a2.withDrawal();

}

void Apply\_Profit()

{

a2.Apply\_Profit();

}

void print()

{

a2.print();

}

};

class GUI{

public:

static int mainMenu();

static int transactionMenu();

};

void main(void)

{

User accounts[5];

int totalAccounts = 0;

clrscr();

int userInput;

while(1){

userInput = GUI::mainMenu();

if(userInput==3){

break;

}else{

switch(userInput){

case 1:

{

while(totalAccounts<5){

accounts[totalAccounts].createAccount();

totalAccounts++;

char ch='0';

if(totalAccounts<5){

cout<<"\n Press 1 to create another account : ";

cout<<"\n Press 0 to go back (Main Menu) : ";

do{

ch = getch();

}while(ch!='1' && ch!='0');

}

else{

cout<<"\n Press any key to go back (Main Menu) ";

}

if(ch=='0'){

break;

}

}

if(totalAccounts>4){

clrscr();

cout<<"\n\n Accounts reachs at its maximum limit!";

getch();

}

}

break;

case 2:

{

if(totalAccounts>0){

int ch = 1;

while(ch!=4)

{

ch = GUI::transactionMenu();

clrscr();

switch(ch){

case 1:

{

char acc\_no[20]; int accountfound = 0;

cout<<"\n\n====================DEPOSIT =========== ";

do{

cout<<"\n\n Enter account number : ";

gets(acc\_no);

for(int i=0;i<5;i++){

if(strcmp(acc\_no,accounts[i].getAccount\_no())==0){

accountfound = 1;

accounts[i].deposit();

break;

}

}

}while(accountfound==0);

if(accountfound==0){

cout<<"\n Account not found, Again ";

}

}

break;

case 2:

{

char acc\_no[20];

int accountfound = 0;

cout<<"\n\n==================== WITHDRAWAL =========== ";

do{

cout<<"\n\n Enter account number : ";

gets(acc\_no);

for(int i=0;i<5;i++){

if(strcmp(acc\_no,accounts[i].getAccount\_no())==0){

accountfound = 1;

accounts[i].withDrawal();

break;

}

}

if(accountfound==0){

cout<<"\n Account not found, Again ";

}

}while(accountfound==0);

}

break;

case 3:

{

char acc\_no[20];

int accountfound = 0;

cout<<"\n\n==================== CHECK BALANCE =========== ";

do{

cout<<"\n\n Enter account number : ";

gets(acc\_no);

for (int i=0;i<5;i++){

if(strcmp(acc\_no,accounts[i].getAccount\_no())==0){

accountfound = 1;

//cout<<"\n\n Customer Name : "<<accounts[i].getName();

//cout<<"\n Your account balance is : "<<accounts[i].getBalance();

accounts[i].Apply\_Profit();

accounts[i].print();

getch();

break;

}

}

if(accountfound==0){

cout<<"\n Account not found, Again ";

}

}while(accountfound==0);

}

break;

}

}

}

else{

cout<<"\n\n First create an account to make transaction!";

getch();

}

}

break;

}

}

}

}

int GUI::mainMenu(){

clrscr();

char choice;

cout<<"======================== UBL BANK SYSTEM =================";

cout<<"\n\n\t\t Press 1 to create new account";

cout<<"\n\n\t\t Press 2 to make transaction";

cout<<"\n\n\t\t Press 3 to exit";

cout<<"\n\n\t\t Enter your choice : ";

cin>>choice;

if(choice>='4'||choice<='0')

{

cout<<"Wrong Input";

getch();

}

else

return choice-48;

}

int GUI::transactionMenu(){

char choice;

clrscr();

cout<<"\n\n================ TRANSACTION MENU ================";

cout<<"\n\n\t\t Press 1 to deposit";

cout<<"\n\n\t\t Press 2 to withdrawal";

cout<<"\n\n\t\t Press 3 to check balance";

cout<<"\n\n\t\t Press 4 to go back (Main Menu)";

cout<<"\n\n\t\t Enter your choice : ";

do{

cin>>choice;

if(choice<'1' || choice>'4'){

cout<<"\n Invalid choice\n Re-enter your choice : ";

}

}while(choice<'1' || choice>'4');

return choice-48;

}

void Account::createAccount(){

clrscr();

cout<<"\n\n================ CREATE NEW ACCOUNT ============ ";

cout<<"\n\n Enter customer name : ";

gets(name);

cout<<" Enter account number : ";

gets(account\_no);

cout<<" Enter balance (at least 500) : ";

cin>>balance;

while(balance<500)

{

cout<<"\nYour minimum Deposit amount should be 500 ";

cout<<"\nPlease re-enter your deposit amount : ";

cin>>balance;

}

time\_t seconds = time(NULL);

struct tm\* tBlock = localtime(&seconds);

char\*temp = asctime(tBlock);

//cout<<"\n Created At length : "<<strlen(createdAt);

createdAt = new char[strlen(temp)];

strcpy(createdAt,temp);

lastTransaction = new char[strlen(temp)];

strcpy(lastTransaction , createdAt);

}

void Account::withDrawal(){

int bal;

cout<<"\n Customer Name : "<<name;

cout<<"\n Account No. : "<<account\_no;

cout<<"\n Enter amount to withdrawal : ";

cin>>bal;

while(bal>balance){

cout<<"\n Your entered amount is greater than your balnce";

cout<<"\n Please re-enter your amount to withdrawal : ";

cin>>bal;

}

balance = balance - bal;

cout<<"\n Your new balance is : "<<balance;

time\_t seconds = time(NULL);

struct tm\* tBlock = localtime(&seconds);

char\* temp = asctime(tBlock);

strcpy(lastTransaction,temp);

getch();

}

void Account::deposit(){

int amount;

cout<<"\n Customer Name : "<<name;

cout<<"\n Account No. : "<<account\_no;

cout<<"\n Enter amount to deposit : ";

cin>>amount;

while(amount<500)

{

cout<<"\nYour minimum Deposit amount should be 500 ";

cout<<"\nPlease re-enter your deposit amount : ";

cin>>amount;

}

balance = balance + amount;

cout<<"\n Your new balance is : "<<balance;

time\_t seconds = time(NULL);

struct tm\* tBlock = localtime(&seconds);

char\* temp = asctime(tBlock);

strcpy(lastTransaction,temp);

getch();

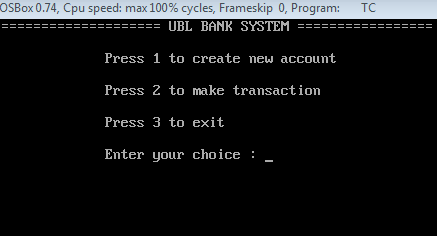
}

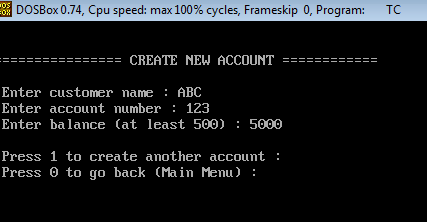
Account::~Account(){

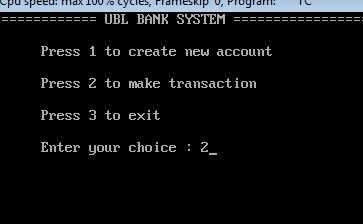
delete [] createdAt;

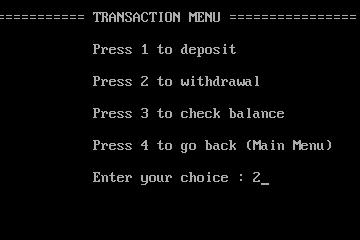
delete [] lastTransaction;

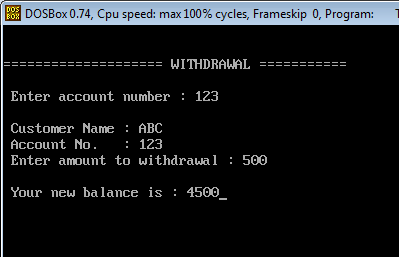
}

**Output 1**

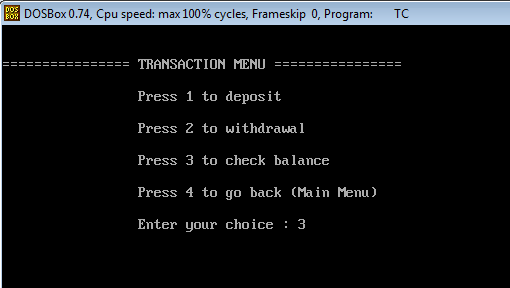
**Output 2**

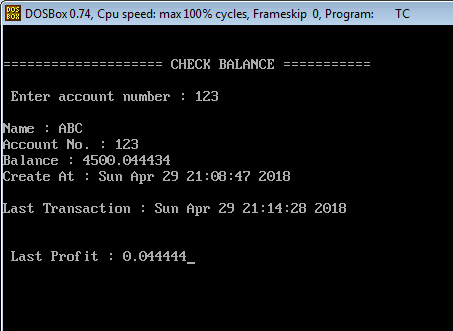
**Output 3**

**Output 4**

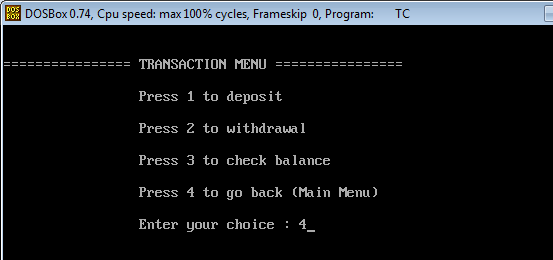
**Output 5**

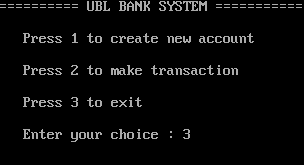
**Output 6**



****

**Output 7**

**Output 8**

**Output 9**