|  |  |  |
| --- | --- | --- |
| INDEX | | |
| S.NO | TOPIC | PAGE NUMBER. |
| 1 | Abstract | 1 |
| 2 | Existing problems | 2 |
| 3 | Proposed solutions | 3 |
| 4 | POSSIBLE CHALLENGES FOR DIGITAL BANKING AND SOLUTIONS TO IT | 4-7 |
| 5 | MINIMUM SYSTEM REQUIREMENTS | 8 |
| 6 | MINIMUM SOFTWARE REQUIREMENTS | 9-15 |
| 7 | Technology used   1. Python 2. mySQL | 17-31 |
| 8 | The code | 32-40 |
| 9 | The output | 41-51 |
| 10 | Advantages and disavantages of the program | 52 |
| 11 | DIGITAL BANKING: THE BANKING TRANSFORMATION ROADMAP | 53 |
| 12 | Conclusion | 54-55 |
| 13 | Bibliography | 56 |

ABSTRACT

The Banking System project is written in C. The project file contains a C script. This is a simple GUI based application which is very easy to understand and use. Talking about the application, the user can simply create their account and login in order to manage their bank accounts.

The user can register his/her account by providing details such as username, opening balance, and pin number. After that, the user has to provide those details in order to access their account. The user can view his/her transaction details, balance inquiry, credit and debit amount. By entering a certain amount only, the user can credit or debit amounts. From this type of application, a user can simply manage his personal accounts with fewer efforts.

This is a simple banking system application for beginners.

# 

**2. EXISTING PROBLEM**

Customer experience is an integral part of a bank’s operations. That’s why banks focus a lot on improving customer experience by removing hassles and enhancing the facilities they provide.

Opening a new account in a bank usually requires a person to visit the bank, fill out a form, and submit the necessary papers. All of these tasks take up a lot of time and dampen the overall customer experience. Moreover, many people have to take time out of their schedules to go to a bank.

.

### **3. PROPOSED SOLUTION**

You can solve this problem by creating a software solution where people can sign up and open a new account in a bank digitally. This way, the person wouldn’t have to visit the bank physically and thus, would save a lot of time and effort. The banking management system can also allow the user to make transactions, deposit and withdraw funds, and check the account balance.

Your solution would need an admin section which would look after the users’ accounts and the overall wellbeing of the database. You’ll have to connect the software to a database which will store all user information in distinct storage.

**ADVANTAGES OF DIGITAL BANKING WITH C:**

* It allows users to open new accounts
* Users can make transactions by entering the respective amounts
* Users can check the balance of their accounts
* Admin can view a list of users to see how many users there are along with their details .

**4. POSSIBLE CHALLENGES FOR DIGITAL BANKING AND SOLUTIONS TO IT**

Banking is considered by most as the business of money. However, talk to a banking veteran and he or she will categorically instruct you that banking is a business of customers. This realisation is what ensured that customer centricity is an important value adopted by banks; and it has stood us in good stead. It then follows, that if we are in the business of customer, then our business is shaped by evolving changes in customer behaviour.

I recently wrote that 2017 was considered by many to be the gloomiest year. However, the year has gone by and the year ahead has been presenting us with both challenges and opportunities.

**Challenges**

**Sustainable competitive advantage**  
One of the most important challenges that has been thrown into focus is the amount of digital banking initiatives that have incremental to no impact on business. Don’t get me wrong; I am the fiercest advocate of experimentation; however, I believe that this effort should not be focused solely on cost and features. Cost and product provided competitive advance in the previous era.

Our attempts need to be centred on experience. Poor experience leads to poor adoption and eventual failure of digital initiatives.

**Convenience and the death of loyalty**  
My earlier point above is further exemplified by the fact that we have moved into a post-loyalty world. Consumers are not tied down by the traditional bonds with their service providers. The urbanisation of everything is also forcing banks to change the way they engage customers, who are now exposed to superior digital offerings from technology giants like Google, Facebook, and

Amazon. We will need to learn from them, match up to them, partner with them, or perish in front of them.

**Understanding customer context**  
The other area where organisations have suffered is trying to superimpose digital solutions on traditional customers. We were fortunate to identify this challenge early and craft a tech centric consumer segmentation approach called DISC (Digital Native, Intelligent, Social, Connected). This allowed us to think of fresh, agile, and relevant solutions that are helping us consolidate our digital leadership.

**Automation and the future of work**  
One of the most important sociological challenges that has been thrown around is the impact of AI and Robotic Process Automation on traditional workforce. India has a proud legacy of a large service sector, with the likes of Indian Railways, India Post, and SBI being some of the largest employers in the world. Therefore, echoing the thoughts of my earlier articles; as technologist, we need to proceed with care when we adopt automation. It is incumbent on us to understand the impact on jobs and help the workforce transition to newer roles. This will not only be responsible growth, but will have an exponential positive impact in the future.

**Opportunities**

**DISC Customers**  
The benefits of the DISC customers (mentioned above) is that they are ready sponges for digital solutions. We are no longer in an era where we have to explain the basic technologies that drive our digital offerings. The post-loyalty DISC customer is also seeking to reduce human-based help in favour of digital: self-help. This once again opens up the canvas for banks across product design, service delivery, and customer support.

**Leveraging the power of Social**  
When I read that Facebook has 2.2 billion monthly active

customers, my jaws dropped. What’s more incredible is that Facebook is only the tip of the iceberg when it comes to social. The gold mine is using social technology for leveraging internal capabilities. Organisations that are able to identify this and organise efforts around this, will see huge dividends.

**The right time and the right place**  
Whilst there is so much more to be said about the challenges and opportunities for banking in the digital era, especially in India, here are some important opportunities that we should seize:

* We have the largest penetration of mobile infrastructure than ever before.
* We have a government and regulator that have been promoting the development of a strong digital economy.

**5. MINIMUM SYSTEM REQUIREMENTS**

**5.1. Software Requirements:**

* **Operating system:** Windows/macOS/Linux
* **Category:** Machine Learning, Deep Learning
* **Programming Language:** Python
* **Back End:** NA

**5.2. Hardware Requirements:**

* **CPU**: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs
* **RAM**: 32 GB (or 16 GB of 1600 MHz DDR3 RAM)
* **Storage**: 300 GB. (600 GB for air-gapped deployments.) Additional space recommended if the repository will be used to store packages built by the customer. With an empty repository, a base install requires 2 GB.

.

**5.3.Intended audience:**Education, developers, data scientists

# 6. HISTORY OF C

C, is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. By design, C provides constructs that map efficiently to typical machine instructions. It has found lasting use in applications previously coded in assembly language. Such applications include operating systems and various application software for computer architectures that range from supercomputers to PLCs and embedded systems.

A successor to the programming language B, C was originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages, with C compilers from various vendors available for the majority of existing computer architectures and operating systems. C has been standardized by ANSI since 1989 (ANSI C) and by the International Organization for Standardization (ISO).

C is an imperative procedural language. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

Since 2000, C has consistently ranked among the top two languages in the TIOBE index, a measure of the popularity of programming languages.

**7. Features of C**

Like most procedural languages in the ALGOL tradition, C has facilities for structured programming and allows lexical variable scope and recursion. Its static type system prevents unintended operations.

In C, all executable code is contained within subroutines (also called "functions", though not strictly in the sense of functional programming). Function parameters are always passed by value (except arrays). Pass-by-reference is simulated in C by explicitly passing pointer values.

C program source text is free-format, using the semicolon as a statement terminator and curly braces for grouping blocks of statements.

The C language also exhibits the following characteristics:

* The language has a small, fixed number of keywords, including a full set of control flow primitives: if/else, for, do/while, while, and switch. User-defined names are not distinguished from keywords by any kind of sigil.
* It has a large number of arithmetic, bitwise, and logic operators: +,+=,++,&,||, etc.
* More than one assignment may be performed in a single statement.
* Functions:
* Function return values can be ignored, when not needed.
* Function and data pointers permit ad hoc run-time polymorphism.
* Functions may not be defined within the lexical scope of other functions.
* Data typing is static, but weakly enforced; all data has a type, but implicit conversions are possible.
* Declaration syntax mimics usage context. C has no "define" keyword; instead, a statement beginning with the name of a type is taken as a declaration. There is no "function" keyword; instead, a function is indicated by the presence of a parenthesized argument list.
* User-defined (typedef) and compound types are possible.
* Heterogeneous aggregate data types (struct) allow related data elements to be accessed and assigned as a unit.
* Union is a structure with overlapping members; only the last member stored is valid.
* Array indexing is a secondary notation, defined in terms of pointer arithmetic. Unlike structs, arrays are not first-class objects: they cannot be assigned or compared using single built-in operators. There is no "array" keyword in use or definition; instead, square brackets indicate arrays syntactically, for example month[11].
* Enumerated types are possible with the enum keyword. They are freely interconvertible with integers.
* Strings are not a distinct data type, but are conventionally implemented as null-terminated character arrays.
* Low-level access to computer memory is possible by converting machine addresses to typed pointers.
* Procedures (subroutines not returning values) are a special case of function, with an untyped return type void.
* A preprocessor performs macro definition, source code file inclusion, and conditional compilation.
* There is a basic form of modularity: files can be compiled separately and linked together, with control over which functions and data objects are visible to other files via static and extern attributes.
* Complex functionality such as I/O, string manipulation, and mathematical functions are consistently delegated to library routines.
* While C does not include certain features found in other languages (such as object orientation and garbage collection), these can be implemented or emulated, often through the use of external libraries (e.g., the GLib Object System or the Boehm garbage collector).

8. THE CODE

**#include <stdio.h>**

**#include <string.h>**

**#include <conio.h>**

**#include <stdlib.h>**

**#include <graphics.h>**

**#include <dos.h>**

**typedef struct Account**

**{**

**char accountName[40];**

**char accountNumber[20];**

**char DateOfBirth[15];**

**char address[50];**

**char contactNum[15];**

**float accountBalance;**

**} Account;**

**void createAccount();**

**void displayAllAccount();**

**void updateAccount();**

**void deleteAccount();**

**void searchAccount();**

**void transaction();**

**int main(){**

**char option;**

**while (option != '0'){**

**system("cls");**

**printf("\t\t\t====== Welcome To A Cube Bank ======\n");**

**printf("\n\t\t\t1. Create Account");**

**printf("\n\t\t\t2. Display All Accounts Information");**

**printf("\n\t\t\t3. Update Account Information");**

**printf("\n\t\t\t4. Delete Account Information");**

**printf("\n\t\t\t5. Search Account Information");**

**printf("\n\t\t\t6. Transaction");**

**printf("\n\t\t\t7. Exit");**

**printf("\n\n\n\t\t\tEnter Your Option: ");**

**scanf(" %c", &option);**

**switch (option)**

**{**

**case '1':**

**createAccount();**

**break;**

**case '2':**

**displayAllAccount();**

**break;**

**case '3':**

**updateAccount();**

**break;**

**case '4':**

**deleteAccount();**

**break;**

**case '5':**

**searchAccount();**

**break;**

**case '6':**

**transaction();**

**break;**

**case '7':**

**printf("\n\t\t\t====== Thank You for using A Cube Bank ===========\n");**

**exit(0);**

**default:**

**printf("\n\t\t\tInvalid Option, Please Enter Right Option !\n");**

**}**

**}**

**}**

**void createAccount(){**

**FILE \*fileOne = fopen("accountInfo.bin", "ab+");**

**if (fileOne == NULL){**

**printf("\n\t\t\tError !\n");**

**}**

**Account accountInformation;**

**system("cls");**

**printf("\t\t\t====== Create Student Account ======\n");**

**printf("\n\t\t\tEnter Account's Name : ");**

**getchar();**

**gets(accountInformation.accountName);**

**printf("\t\t\tEnter Account's Number : ");**

**gets(accountInformation.accountNumber);**

**printf("\t\t\tEnter Account Holder's Date of birth : ");**

**gets(accountInformation.DateOfBirth);**

**printf("\t\t\tEnter Account Holder's Address : ");**

**gets(accountInformation.address);**

**printf("\t\t\tEnter Account Holder's Contact Number : ");**

**gets(accountInformation.contactNum);**

**printf("\t\t\tEnter Current Account Balance : $ ");**

**scanf("%f", &accountInformation.accountBalance);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, fileOne);**

**printf("\n\n\t\t\tAccount has been opened sucessfully\n");**

**printf("\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**fclose(fileOne);**

**}**

**void displayAllAccount(){**

**FILE \*fileOne = fopen("accountInfo.bin", "rb");**

**if (fileOne == NULL){**

**printf("\n\t\t\tError !\n");**

**}**

**Account accountInformation;**

**system("cls");**

**printf("\t\t\t====== Display All Account's Information ======\n\n");**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1){**

**printf("\t\t\tAccount Name : %s\n\t\t\tAccount Number : %s\n\t\t\tDate of Birth : %s\n\t\t\tAddress : %s\n\t\t\tContact Number : %s\n\t\t\tCurrent Account's Balance : %.2f\n", accountInformation.accountName, accountInformation.accountNumber, accountInformation.DateOfBirth, accountInformation.address, accountInformation.contactNum, accountInformation.accountBalance);**

**printf("\t\t\t---------------------------------------------------\n");**

**}**

**fclose(fileOne);**

**printf("\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**}**

**void updateAccount(){**

**FILE \*fileOne = fopen("accountInfo.bin", "rb");**

**FILE \*temp = fopen("temp.bin", "wb");**

**Account accountInformation, tempInformation;**

**int choice, flag = 0;**

**if (fileOne == NULL || temp == NULL)**

**{**

**printf("\n\t\t\tError !\n");**

**}**

**system("cls");**

**printf("\t\t\t\t====== Update Account Information ======\n");**

**printf("\n\t\t\tEnter Account Number : ");**

**getchar();**

**gets(tempInformation.accountNumber);**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1)**

**{**

**if (strcmp(accountInformation.accountNumber, tempInformation.accountNumber) == 0)**

**{**

**flag++;**

**printf("\n\t\t\tChoose your option :\n\t\t\t1.Update Account Name\n\t\t\t2.Update Date of Birth.\n\t\t\t3.Update Address\n\t\t\t4.Update Contact No.");**

**printf("\n\n\t\t\tEnter Your Option : ");**

**scanf("%d", &choice);**

**if (choice == 1)**

**{**

**printf("\n\t\t\tEnter Account's Name to Update: ");**

**getchar();**

**gets(tempInformation.accountName);**

**strcpy(accountInformation.accountName, tempInformation.accountName);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tUpdated successfully!\n\n");**

**}**

**else if (choice == 2)**

**{**

**printf("\n\t\t\tEnter Date of Birth to Update: ");**

**getchar();**

**gets(tempInformation.DateOfBirth);**

**strcpy(accountInformation.DateOfBirth, tempInformation.DateOfBirth);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tUpdated successfully!\n\n");**

**}**

**else if (choice == 3)**

**{**

**printf("\n\t\t\tEnter Address to Update: ");**

**getchar();**

**gets(tempInformation.address);**

**strcpy(accountInformation.address, tempInformation.address);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tUpdated successfully!\n\n");**

**}**

**else if (choice == 4)**

**{**

**printf("\n\t\t\tEnter Contact No. to Update: ");**

**getchar();**

**gets(tempInformation.contactNum);**

**strcpy(accountInformation.contactNum, tempInformation.contactNum);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tUpdated successfully!\n\n");**

**}**

**else**

**{**

**printf("\n\t\t\tInvalid Option.");**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**}**

**else**

**{**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**}**

**fclose(fileOne);**

**fclose(temp);**

**remove("accountInfo.bin");**

**rename("temp.bin", "accountInfo.bin");**

**if (flag == 0)**

**{**

**printf("\n\t\t\tAccount is not found");**

**}**

**printf("\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**}**

**void deleteAccount()**

**{**

**FILE \*fileOne = fopen("accountInfo.bin", "rb");**

**FILE \*temp = fopen("temp.bin", "wb");**

**Account accountInformation, tempInformation;**

**int choice, flag = 0;**

**if (fileOne == NULL || temp == NULL)**

**{**

**printf("\n\t\t\tError !\n");**

**}**

**system("cls");**

**printf("\t\t\t\t====== Delete Account Information ======\n");**

**printf("\n\t\t\tEnter Account Number : ");**

**getchar();**

**gets(tempInformation.accountNumber);**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1)**

**{**

**if (strcmp(accountInformation.accountNumber, tempInformation.accountNumber) == 0)**

**{**

**flag++;**

**printf("\n\t\t\tAre you sure to delete ??\n\t\t\t\t1.Yes\n\t\t\t\t2.Back\n\t\t\t\tEnter Your Option : ");**

**scanf("%d", &choice);**

**if (choice == 1)**

**{**

**printf("\n\n\t\t\tInformation has been deleted successfully!\n\n");**

**}**

**else if (choice == 2)**

**{**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**else**

**{**

**printf("\n\t\t\tInvalid Option");**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**}**

**else**

**{**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**}**

**fclose(fileOne);**

**fclose(temp);**

**remove("accountInfo.bin");**

**rename("temp.bin", "accountInfo.bin");**

**if (flag == 0)**

**{**

**printf("\n\t\t\tAccount is not found");**

**}**

**printf("\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**}**

**void searchAccount()**

**{**

**FILE \*fileOne = fopen("accountInfo.bin", "rb");**

**Account accountInformation;**

**int choice, flag = 0;**

**char searchAccountNumber[20], searchName[50];**

**if (fileOne == NULL)**

**{**

**printf("\n\t\t\tError !\n");**

**}**

**system("cls");**

**printf("\t\t\t\t====== Search Account Information ======\n");**

**printf("\n\t\t\tChoose your option :\n\t\t\t1.Search by Account Number\n\t\t\t2.Search by Account's Name");**

**printf("\n\n\t\t\tEnter Your Option : ");**

**scanf("%d", &choice);**

**if (choice == 1)**

**{**

**system("cls");**

**printf("\t\t\t\t====== Search Account Information ======\n");**

**printf("\n\n\t\t\tEnter Account Number : ");**

**getchar();**

**gets(searchAccountNumber);**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1)**

**{**

**if (strcmp(accountInformation.accountNumber, searchAccountNumber) == 0)**

**{**

**flag++;**

**printf("\n\t\t\tAccount Name : %s\n\t\t\tAccount Number : %s\n\t\t\tDate of Birth : %s\n\t\t\tAddress : %s\n\t\t\tContact Number : %s\n\t\t\tCurrent Account's Balance : %.2f\n", accountInformation.accountName, accountInformation.accountNumber, accountInformation.DateOfBirth, accountInformation.address, accountInformation.contactNum, accountInformation.accountBalance);**

**}**

**}**

**if (flag == 0)**

**{**

**printf("\n\t\t\tAccount is not found");**

**}**

**}**

**else if (choice == 2)**

**{**

**system("cls");**

**printf("\t\t\t\t====== Search Account Information ======\n");**

**printf("\n\n\t\t\tEnter Account Name : ");**

**getchar();**

**gets(searchName);**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1)**

**{**

**if (stricmp(accountInformation.accountName, searchName) == 0)**

**{**

**flag++;**

**printf("\n\t\t\tAccount Name : %s\n\t\t\tAccount Number : %s\n\t\t\tDate of Birth : %s\n\t\t\tAddress : %s\n\t\t\tContact Number : %s\n\t\t\tCurrent Account's Balance : %.2f\n", accountInformation.accountName, accountInformation.accountNumber, accountInformation.DateOfBirth, accountInformation.address, accountInformation.contactNum, accountInformation.accountBalance);**

**printf("\t\t\t---------------------------------------------------\n");**

**}**

**}**

**if (flag == 0)**

**{**

**printf("\n\t\t\tAccount is not found");**

**}**

**}**

**else**

**{**

**printf("\n\t\t\tInvalid Option");**

**}**

**fclose(fileOne);**

**printf("\n\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**}**

**void transaction()**

**{**

**FILE \*fileOne = fopen("accountInfo.bin", "rb");**

**FILE \*temp = fopen("temp.bin", "wb");**

**Account accountInformation, tempInformation;**

**int op, flag = 0;**

**if (fileOne == NULL || temp == NULL)**

**{**

**printf("\n\t\t\tError !\n");**

**}**

**system("cls");**

**printf("\t\t\t\t====== Account Transaction ======\n");**

**printf("\n\t\t\tEnter Account Number : ");**

**getchar();**

**gets(tempInformation.accountNumber);**

**while (fread(&accountInformation, sizeof(accountInformation), 1, fileOne) == 1)**

**{**

**if (strcmp(accountInformation.accountNumber, tempInformation.accountNumber) == 0)**

**{**

**flag++;**

**printf("\n\n\t\t\tDo you want to\n\t\t\t1.Deposit\n\t\t\t2.Withdraw\n\n\t\t\tEnter your choice:");**

**scanf("%d", &op);**

**if (op == 1)**

**{**

**printf("\n\t\t\tCurrent Balance:$ %.2f", accountInformation.accountBalance);**

**printf("\n\t\t\tEnter the amount you want to deposit:$ ");**

**scanf("%f", &tempInformation.accountBalance);**

**accountInformation.accountBalance += tempInformation.accountBalance;**

**printf("\n\t\t\tCurrent Balance after Deposit:$ %.2f", accountInformation.accountBalance);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tDeposited successfully!\n\n");**

**}**

**else**

**{**

**printf("\n\t\t\tCurrent Balance:$ %.2f", accountInformation.accountBalance);**

**printf("\n\t\t\tEnter the amount you want to withdraw:$ ");**

**scanf("%f", &tempInformation.accountBalance);**

**if (accountInformation.accountBalance < tempInformation.accountBalance)**

**{**

**printf("\n\n\t\t\tUnsufficient Balance!\n\n");**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**else**

**{**

**accountInformation.accountBalance -= tempInformation.accountBalance;**

**printf("\n\t\t\tCurrent Balance after Withdraw:$ %.2f", accountInformation.accountBalance);**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**printf("\n\n\t\t\tWithdraw successfully!\n\n");**

**}**

**}**

**}**

**else**

**{**

**fwrite(&accountInformation, sizeof(accountInformation), 1, temp);**

**}**

**}**

**if (flag == 0)**

**{**

**printf("\n\t\t\tAccount is not found");**

**}**

**fclose(fileOne);**

**fclose(temp);**

**remove("accountInfo.bin");**

**rename("temp.bin", "accountInfo.bin");**

**printf("\n\n\t\t\tEnter any keys to continue.......");**

**getch();**

**}**

**9. THE OUTPUT**

Main Menu

Text

Description automatically generated

* 1. CREATING AN ACCOUNT
  2. DISPLAY ALL ACCOUNTS INFORMATION
  3. UPDATING ACCOUND INFORMATION
  4. DELETING ACCOUNT INFORMATION
  5. SEARCHING ACCOUNT INFORMATION
  6. TRANSACTION
  7. EXIT

Creating an Account

Text

Description automatically generated

**Displaying the account Information**

Text

Description automatically generated

**Updating account information**

Text

Description automatically generated

Text

Description automatically generatedDeleting account information

**Searching account information**Text

Description automatically generated

Transaction

Text

Description automatically generated

Exit

Text

Description automatically generated

10 .ADVANTAGES AND DISADVANTAGES OF THE PROGRAM

PROGRAM WEAKNESSES

We know that, no program can be 100% efficient. So there are some drawbacks:

* + - * + It cannot perform all the required functions as bank required, it’s simply a record of account of customer
        + System is not sharply a graphical user interface. There is just use of some text color.
        + It’s not a multitasking program.

PROGRAM STRENGTH

* + - * + There are advantages of using this program.
        + User friendly software, and easy to use by just
        + following the instructions.
        + Needs user acc. No so that inly authorized users are allowed to access.
        + Once a record has been saved, duplicate one cannot be made, so that there is no misplace of the record

**11. DIGITAL BANKING: THE BANKING TRANSFORMATION ROADMAP**

The leaders in digital banking are more client-centric, tech-savvy, and inclusive—and are fundamentally changing to deliver the best results.

* Most banks today want to become digital banking leaders—after all, that's where the customers are. And for much of the past decade as digital banking has taken hold, most leading traditional banks have incorporated strong digital strategies.
* So what separates the digital banking leaders from the laggards? A new Kearney study on digitization, , seeks the answer and finds three main findings: the leaders understand the importance of mobile in a digital strategy, they are developing more agile operating models, and, most notably, they have tackled the need for internal culture shifts .

**12. CONCLUSION**

The mobile and wireless market has been one of the fastest growing markets in the world.  The arrival of technology and the escalating use of

 mobile and smart phone devices, has given the banking industry a new platform.  Connecting a customer anytime and anywhere to their

 money and needs is a must have service that has become an unstoppable necessity.  This worldwide communication is leading a new

generation of strong banking relationships. The banking world can achieve superior interactions with their public base if they accommodate all

 their customer needs.  They have a unique challenge to keep their customer alliances and keeping up with the new technologies, and

 competitive strategies that other banks also have to offer the public. Conveniences of services plus outside locations like ATMS are crucial to

 every banks success. Meeting all challenges including safety and security are perfect examples of good banking strategies. In order for the

 financial institutions to effectively grow they must embrace the new technologies and customize them to suit their economic success and the public’s success.

Online banking is certainly here to stay. Online banking is a necessity for the bank's that we studied and others in order for them to stay in business.

While its existence doesn't necessary give them a competitive edge because it is so common place, it is truly a cost of doing business.

As a tool of modern living and as a lifestyle aid, it is absolutely indispensable. The fact is that many services that are now being offered with online banking are almost impossible to do conveniently with regular banking.

As we venture into the future, the internet will undoubtedly continue to change the banking industry.

**13. BIBLIOGRAPHY**

**1** [**https://www.python.org/**](https://www.python.org/)

**[2 www.mysql.com](http://2 www.mysql.com)**

**3** [**https://www.upgrad.com/blog/python-banking-project/**](https://www.upgrad.com/blog/python-banking-project/)

**4** [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)

**5** [**https://blog.dime.com/personal-banking/top-5-pros-and-cons-of-digital-banking/**](https://blog.dime.com/personal-banking/top-5-pros-and-cons-of-digital-banking/)