**Table of contents**

* Abstract
* Software Development Methodology
* Introduction

1. Purpose
2. Advantages
3. Scope
4. Existing System with Limitations
5. Proposed System Features

* Specification Requirements

1. Software and Hardware requirement Specification
2. User requirements
3. Functional Requirements

* Technologies Used

1. programming language: C
2. Integrated Development Environment (IDE): Code::blocks-16.01

* Behavioral Description

1. DFD
2. Use Case Diagram
3. Activity Diagrams

* System Design

1. Sequence and Collaboration Diagram
2. Subsystem services
3. Class Diagrams

* Testing and Implementation
* Output Results
* Coding
* Conclusion
* Bibliography

**ABSTRACT**

**ABOUT THE PROJECT**

The adoption of the Electronic Banking system bycommercial enterprises has been inexistence since the mid-90s, much greaterin number due to lower operating costsassociated with it and safety. Electronic banking hasinitiallybeen in the form of automatic tellermachines and telephone transactions.More recently, it has been transferred madeby the Internet, a new delivery channel forbanking services that benefits bothcustomers and banks. Internet bankingsystem services can include: Open anaccount, Balance inquiry, Request forCheque book, Furthermore, customer’sapplication for electronic banking facilitiesis expanding as the cost savings ontransactions over the Internet aresignificant. Therefore, we came up with the idea of creating a Bank Management System in order to manage efficiently different kinds of operations related to the Bank.

**Introduction**

**INTRODUCTION**

The project that we have undertaken aims to develop a banking system that is  
clean, user-friendly and multi-functional. The development of this application  
includes a number of fields such that user feels comfortable and the system  
appears as dynamic to him. The project “Banking Management System” includes the following functionalities*:*

* User can create a new account
* User can update the information of an existing account
* User can also view and manage his/her transactions(Deposit and withdraw)
* User can check the details of an existing account
* remove the existing account and view the customers’ list.

### Advantages:

The application will be extremely beneficial for the Customers  
intending to use and operate their bank account and will get various benefits in  
the field of management of accounts on a clean and user-friendly platform.  
“Bank Management System”, is a simple application, which is especially generated  
and designed for the bank in order to efficiently manage customer's information.

The Design and development of this Bank Management system provide a more secure approach in managing bank customer’s information which strengthens the relationships between banks and their customers by providing the right solutions that use multi-level security to improve customer satisfaction.

**Scope:**

A Bank Management System will be applicable everywhere, where banking exists. It will be more efficient and provide an easier way to have a record on systems through which everyone can easily access it according to his rights as compared to the traditional banking system. Every bank will prefer the Bank Management System application instead of the traditional banking system as it contains many useful features and fastest methods for the transactions.

### Existing System with Limitations:

Internet Banking System refers to systems that enable bank customers to access accounts and general information on bank products and services through a personal computer or other intelligent devices. But most of these systems do not focus on how best to manage and keep their customer’s data more secured. The chances and threats that the internet symbolizes is no longer news to the present-day banking sector. No traditional bank would dare face investment analysts or new customers without an internet strategy. The main intention behind the commencement of electronic banking services is to provide the customers with an alternative that is more responsive and with less expensive options. With options more secured, customers have more control than ever.

### 

### Proposed System Features:

Following are the major objectives behind the new proposed system:

* It creates a user-friendly environment, where a normal user can access through all the benefits of the system.
* It provides security from unauthorized access, only authorized users are access granted to the system.
* It increases efficiency and saves time.
* No danger and obstacles from external entities.
* Easy access to save data inside the system.
* Complex Banking operations and Transaction operations are efficiently handled by the application such as calculating the interest rate of Users.
* It is cost-effective
* It is highly secured and less time consuming; hence time wastage can be avoided
* Up to date records of the customers are maintained by the system.

**SPECIFICATION**

**REQUIREMENT**

**SPECIFICATION REQUIREMENT**

In this section we divided the requirements into three parts which are Software and Hardware requirements, User requirement and functional requirement, basically, the first requirement deals with the requirement related to the software and the hardware, the second deals with what the user wants, in other words, what should the system do, and lastly the functional requirements(interaction scenarios) which allows us to see how would the system interact with the user at various level of the program. This requirement would be discussed in detail in the behavioral description.

**Software requirement Specification:**

In order to develop this project, the minimum software requirement specifications must be fulfilled

A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents, defines a requirement associated with the definition of software which allows the user to interact with the hardware, defines the software requirement as follow:

**Operating System:** Win 7, Win 8, Win 10.

**Presentation layer :** C programming language.

**Database:** File has been used to store data related to a new account, transaction, editing of account information and viewing of account information

**Documentation Tool:** Microsoft Word

**Hardware Requirement Specification:**

The collection of internal electronic circuits and external physical devices used in building a computer is called the Hardware. The minimum hardware requirement specifications for developing this project are as follows:

**Processor:** Standard processor with a speed of 1.6 GHz or more

**RAM:** 256 MB RAM or more

**Hard Disk:** 10 GB or more

**Monitor:** Standard color monitor

**User requirement Specification:**

this requirement consists of what the user wants, what the user expects moreover what the system should do according to the user specification. In our Bank Management System, the user expects to have a system that performs specific tasks, can create a new account, update the information of an existing account, view and manage transactions, check the details of an existing account, remove the existing accounts and view customers’ list.

**Functional requirements:**

1. Displays the menu or welcome screen to perform different banking activities mentioned below.
2. creates a new customer account. It asks for some personal and banking details of the customer such as name, date of birth, citizenship number, address and phone number. You can enter the amount to deposit and choose one type of deposit account – saving, current, fixed for 1 year, fixed for 2 years or fixed for 3 years.
3. View the customer’s banking information such as account number, name, address and phone number provided while creating the account.
4. change the address and phone number of a particular customer account.
5. deposit and withdraw money to and from a particular customer account.
6. deleting a customer account. shows account number, name, date of birth, citizenship number, age, address, phone number, type of account, the amount deposited and date of deposit. It also displays the amount of interest corresponding to a particular account type.

**TECHNOLOGIES**

**USED**

**TECHNOLOGIES USED**

**programming language: C**

**History of C language**

The C programming language is a computer programming language that was developed to do system programming for the operating system UNIX and is an imperative programming language. C was developed in the early 1970s by *Ken Thompson* and *Dennis Ritchie* at Bell Labs. It is a procedural language, which means that people can write their programs as a series of step-by-step instructions. C is a compiled language.

Because the ideas behind C are kept close to the design of the computer, the compiler (program builder) can generate machine code/native code for the computer. Programs built-in machine code is very fast. This makes C a good language for writing operating systems. Many operating systems, including Linux and UNIX, are programmed using this language. The language itself has very few keywords, and most things are done using libraries, which are collections of code for them to be reused.

C is available for many different types of computers. This is why C is called a "portable" language. A program that is written in C and that respects certain limitations can be compiled for many different platforms.

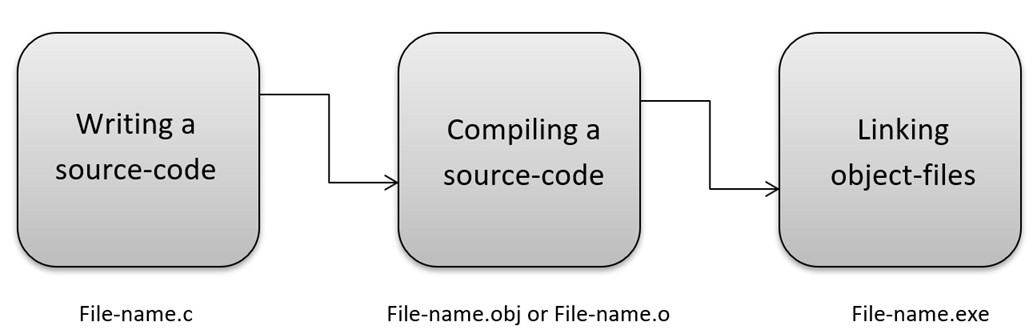
The syntax of C has also influenced many other programming languages, such as C++, C#, and Java, and many more programming languages we use nowadays.

**Where is C used? Key Applications**

1. 'C' language is widely used in embedded systems.
2. It is used for developing system applications.
3. It is widely used for developing desktop applications.
4. Most of the applications by Adobe are developed using 'C' programming language.
5. It is used for developing browsers and their extensions. Google's Chromium is built using 'C' programming language.
6. It is used to develop databases. MySQL is the most popular database software which is built using 'C'.
7. It is used in developing an operating system. Operating systems such as Apple's OS X, Microsoft's Windows, and Symbian are developed using 'C' language. It is used for developing desktop as well as mobile phone's operating system.
8. It is used for compiler production.
9. It is widely used in IoT applications.

**How 'C' Works?**

C is a compiled language. A compiler is a special tool that compiles the program and converts it into the object file which is machine-readable. After the compilation process, the linker will combine different object files and creates a single executable file to run the program. The following diagram shows the execution of a 'C' program

[](https://www.guru99.com/images/1/012419_1229_WhatisCProg4.png)

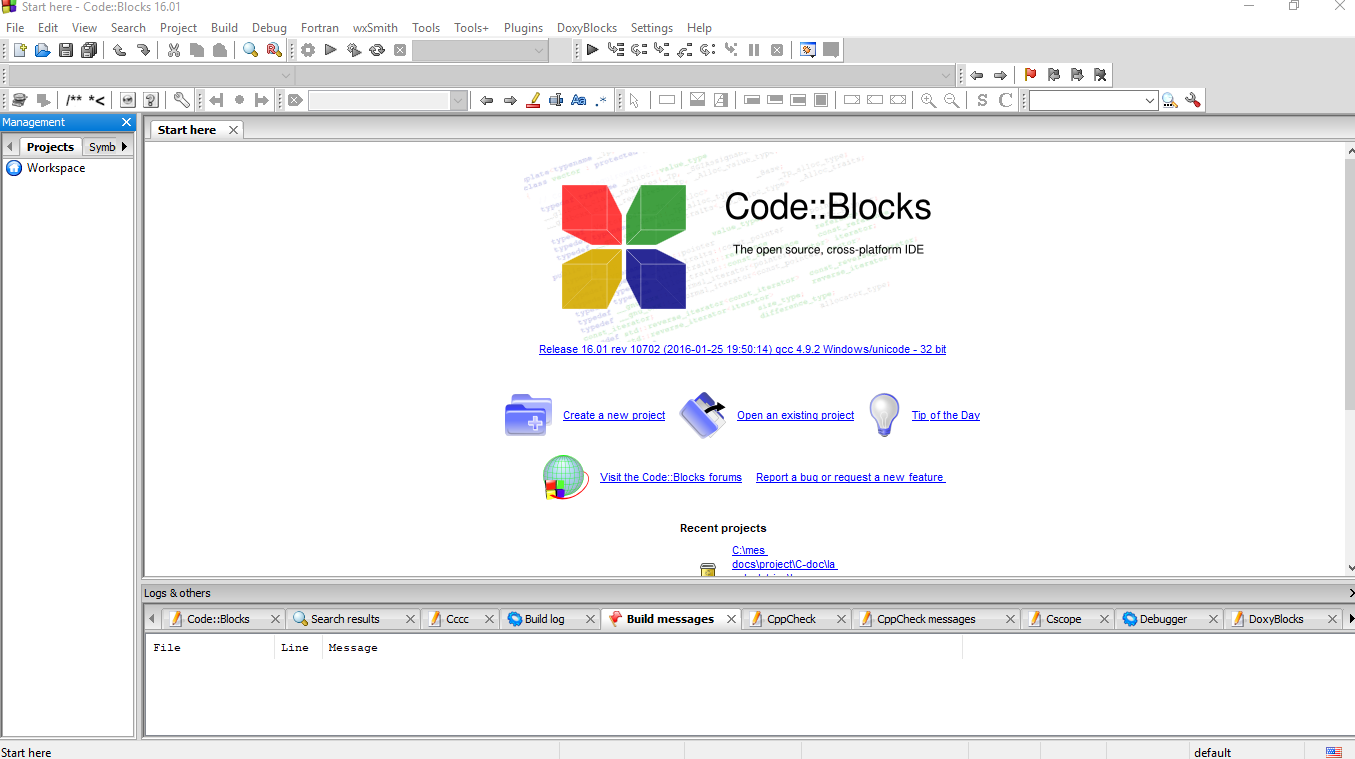
Nowadays, various compilers are available online, and you can use any of those compilers. The functionality will never differ and most of the compilers will provide the features required to execute both 'C' and 'C++' programs.

**Integrated Development Environment (IDE): code::blocks 16.01**

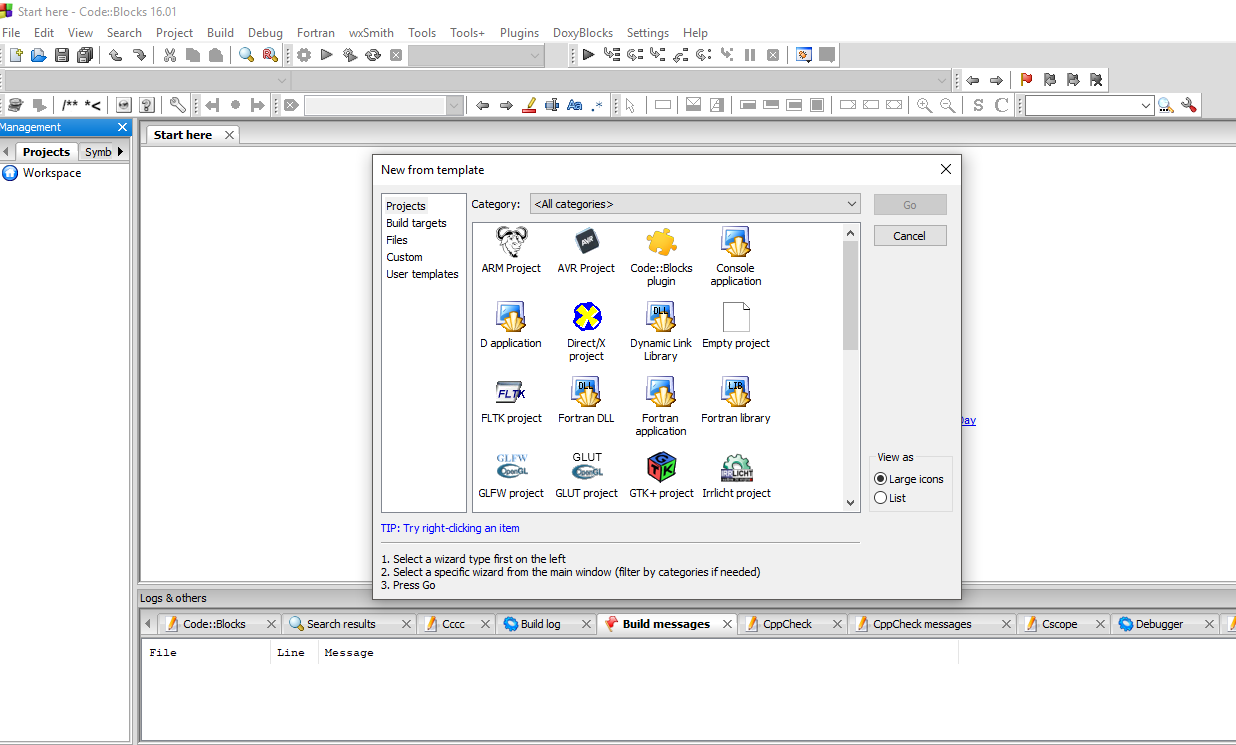
Code::Blocks is a free C, C++, and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Finally, an IDE with all the features you need, having a consistent look, feel and operation across platforms.

**code::blocks interface :**

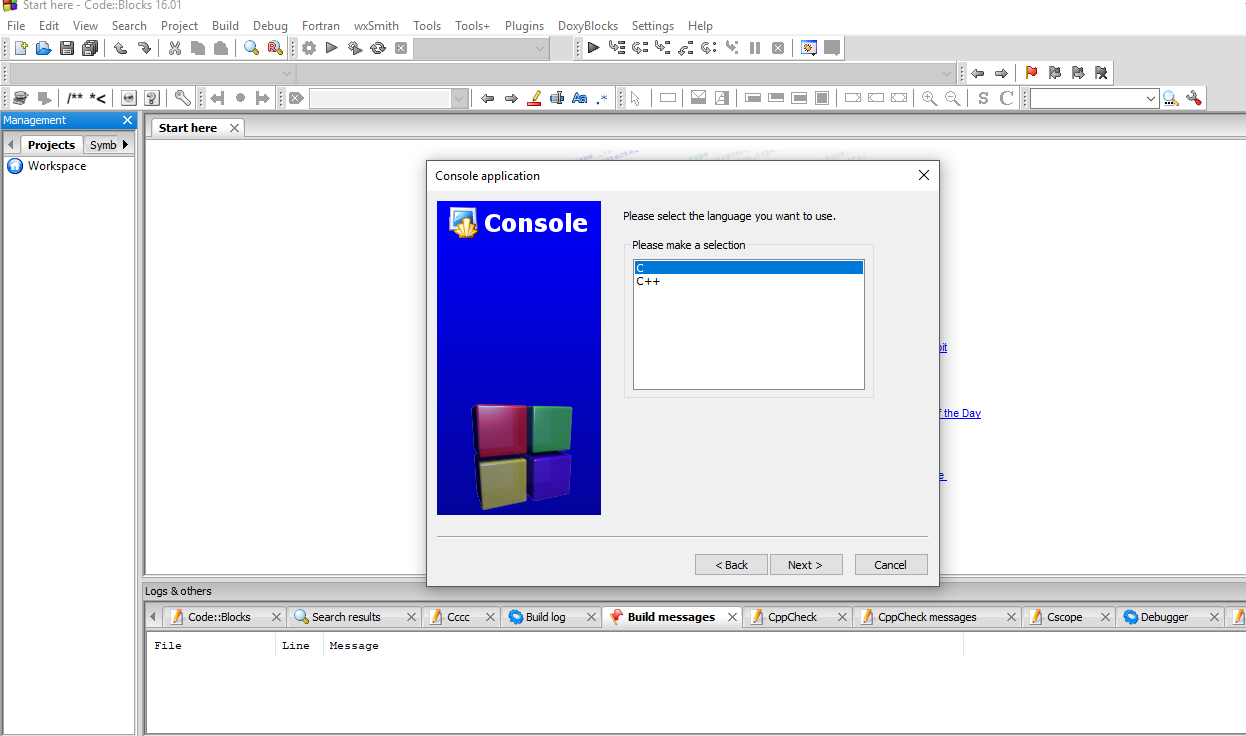
here, is how the interface of code::blocks-16.01 looks like



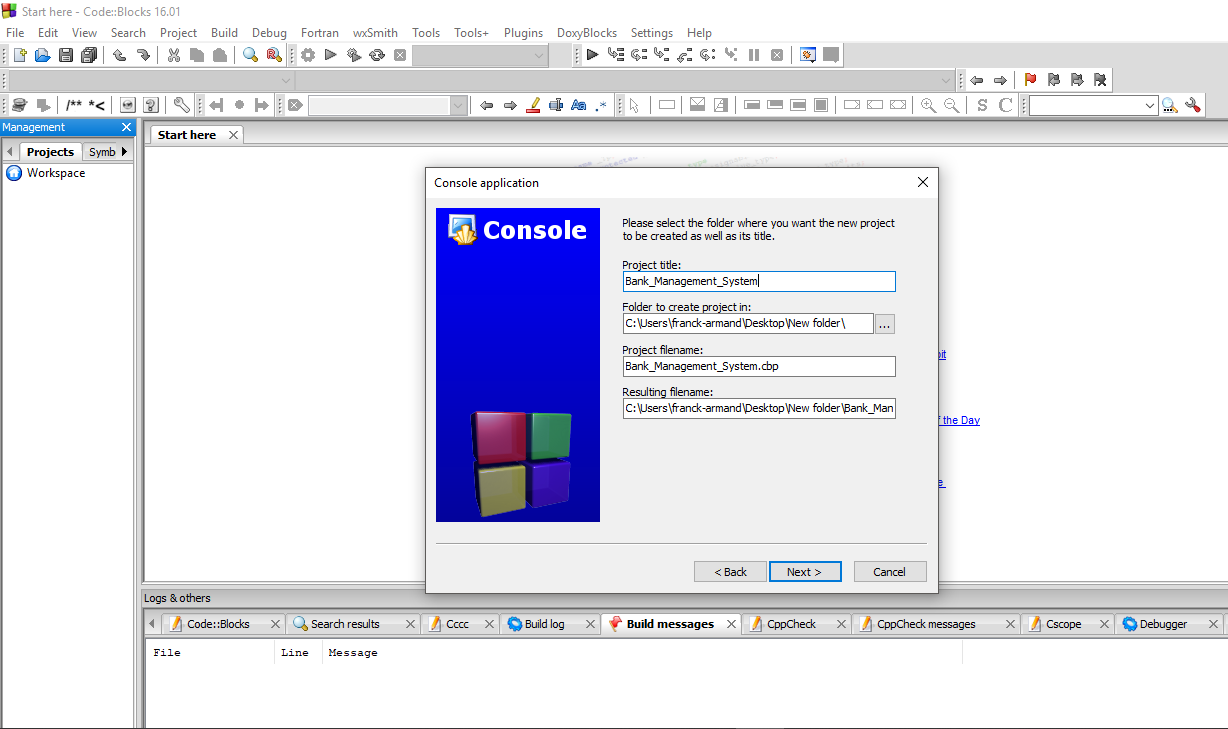
Now let’s create our project “Bank\_Management\_System.c” so from the previous picture we must click on “Create a new project”.



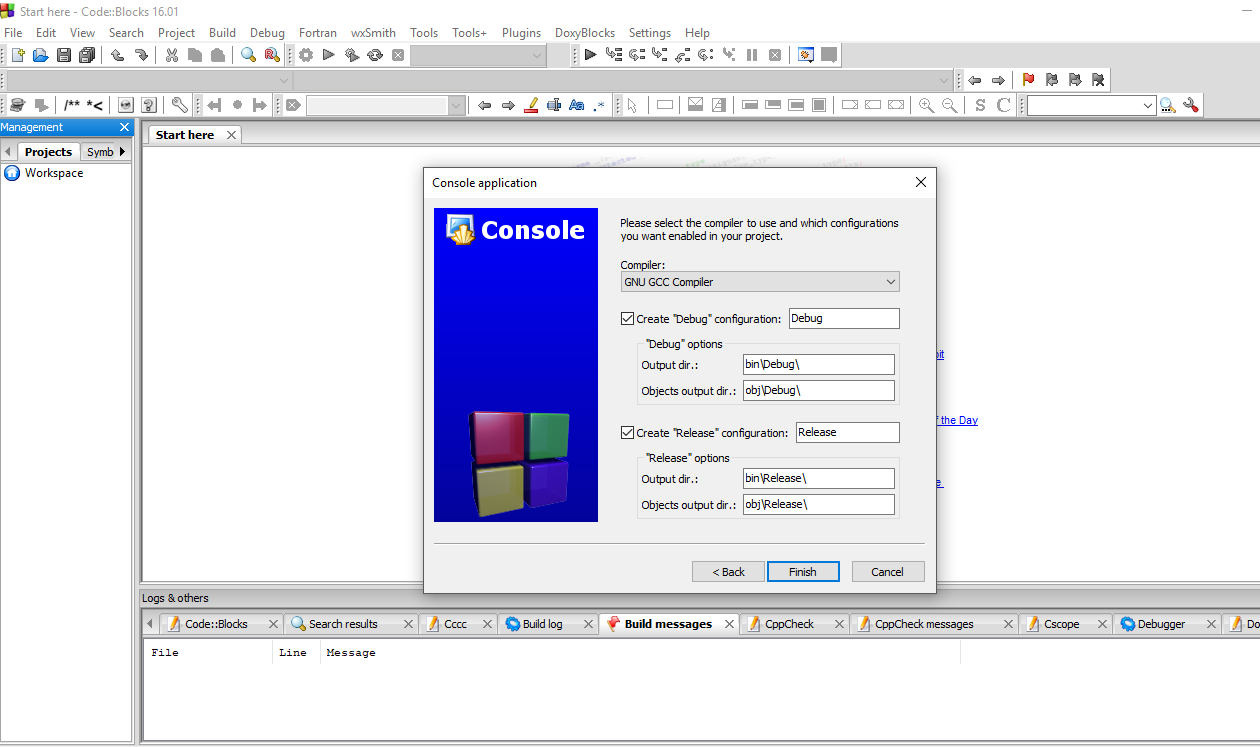
Great! now since our program rums in console mode, we must specify that to code::blocks, therefore, we click on “Console application”.



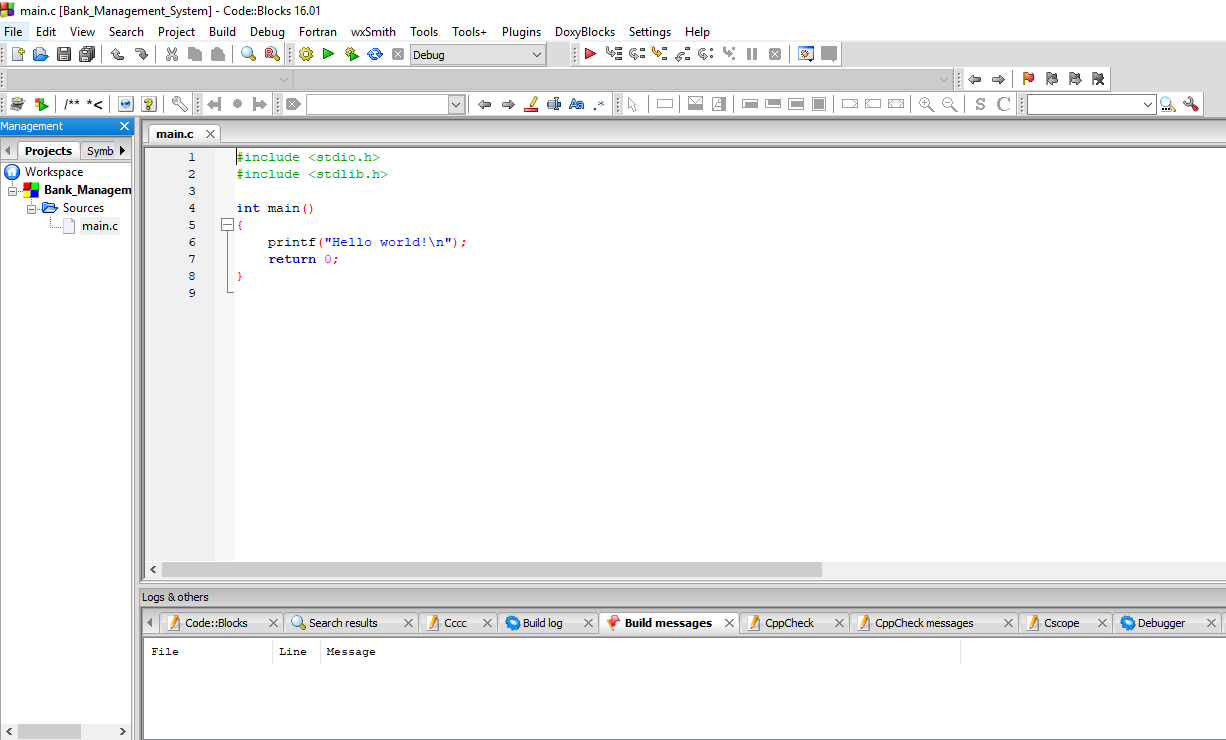
Once again, we must specify the language we want to code in, in our case it’s C and we do “Next”.



Project title: is the name we decide to give to or project as I mentioned earlier it is “Bank\_Management\_System”, also the folder we would like our project to be saved into, in my case it is saved in a new folder on my Desktop. After that, we do “Next”



As mentioned before C language is a compiled language so here is the compiler used to compile our code. We press finish!



That’s it, we are all set to start coding.

**BEHAVIORAL**

**DESCRIPTION**

**BEHAVIORAL DESCRIPTION**

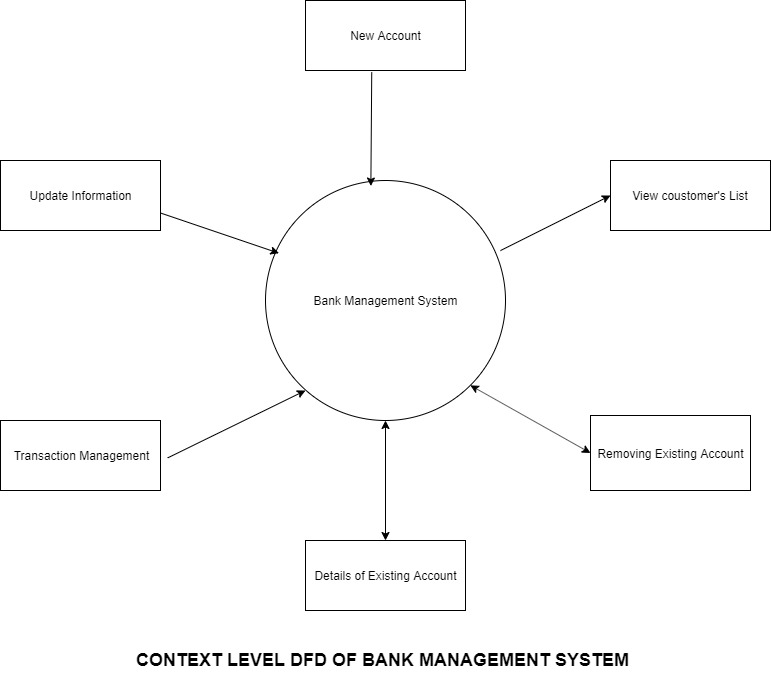
Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes the flow of data through a system to perform certain functions of a business. The physical data flow diagram describes the implementation of the logical data flow. In our project we are going to represent two types of DFD, which are mentioned below:

* Context Level DFD
* Level 2 DFD

**Context Level DFD:**

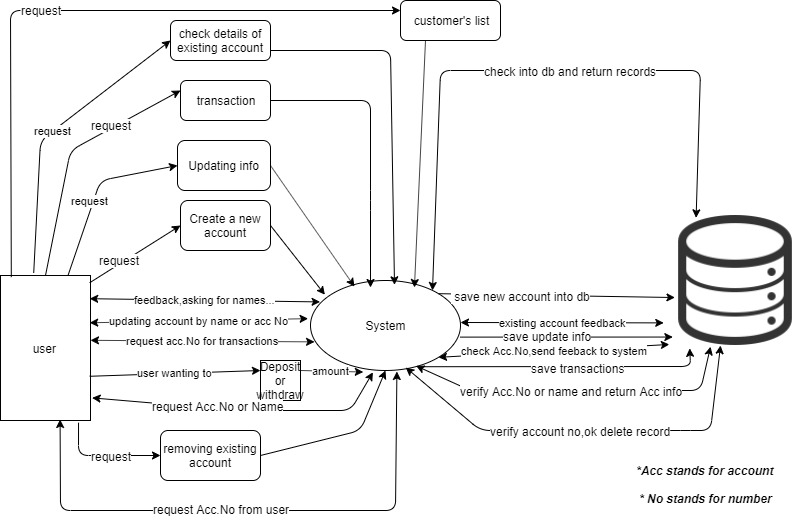
Context Level DFD, A context-level data flow diagram (DFD) provides an at-a-glance look at an information system and the ways it exchanges data with outside entities. In the Context Level, the whole system is shown as a single process.

* **Comprised of**
  + A single process representing the entire system modeled
  + External entities
  + The data flows that pass between the external entities and the system
* **Purpose is**
  + to identify and examine the interfaces between the external entities and the system



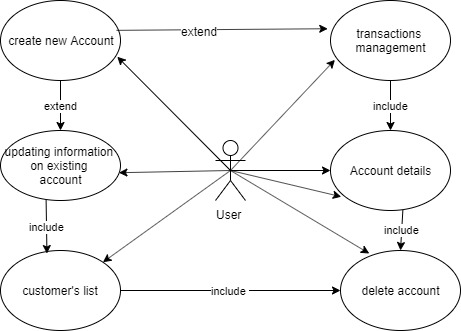
**Level 2 DFD:**

A level 2 data flow diagram (DFD) offers a more detailed look at the processes that make up an information system than a level 1 DFD does. It can be used to plan or record the specific makeup of a system.



**Use Case Documentation:**

A use case diagram is a graphic representation of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. ... The actors, usually individuals involved with the system defined according to their roles.

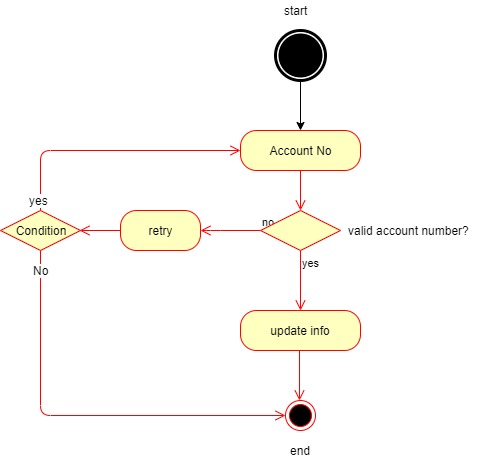


**Process Flow**

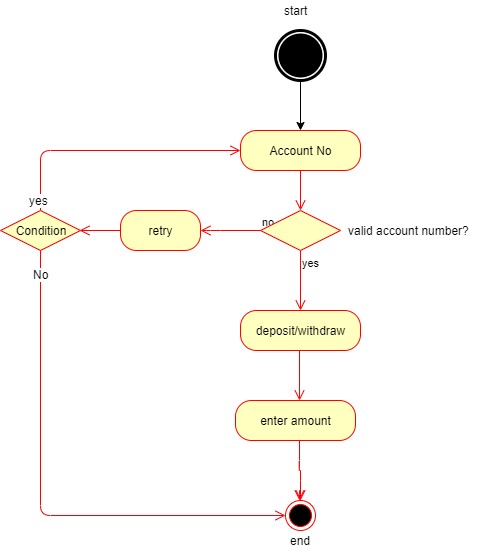
**Activity diagram**

The activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called an object-oriented flowchart. Activity diagrams consist of activities that are made up of actions that apply to behavioral modeling technology.

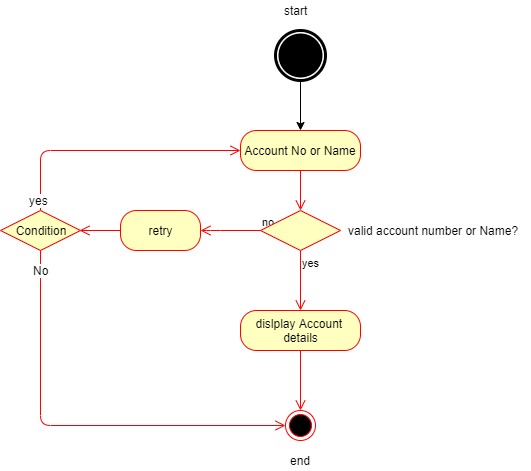
* **Update info on existing Account**



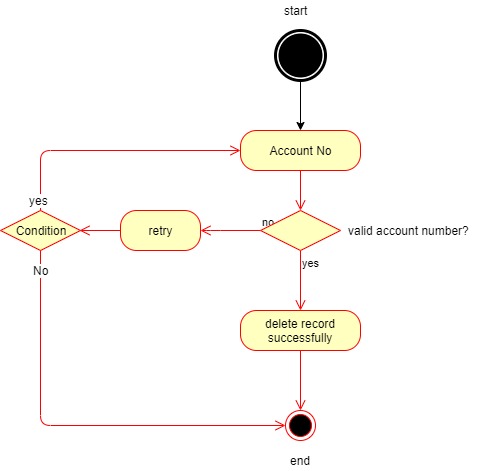
* **Transactions management**



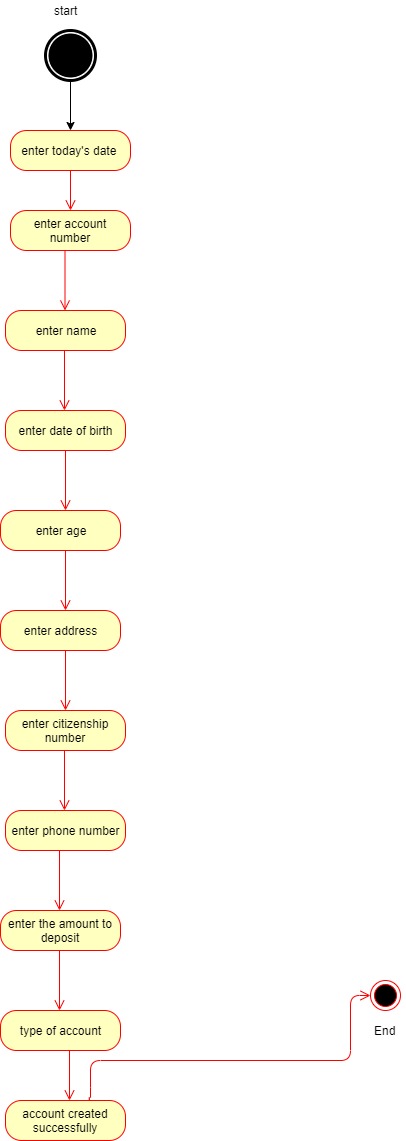
* **Check the details of the existing account**



* **Removing existing account**



* **Create a New Account**



**SYSTEM DESIGN**

**SYSTEM DESIGN**

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System Design focuses on how to accomplish the objective of the system.

System Analysis and Design (SAD) mainly focuses on

* Systems
* Processes
* Technology

**What is a System?**

The word System is derived from the Greek word “*Systema”,* which means an organized relationship between any set of components to achieve some common cause or objective.

*A system is “an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal.”*

**Constraints of a System**

A system must have three basic constraints which can be defined as follow

* A system must have some structure and behavior which is designed to achieve a predefined objective.
* Interconnectivity and interdependence must exist among the system components.
* The objectives of the organization have a higher priority than the objectives of its subsystems.

**Properties of a System**

A system has the following properties

* Organization

The organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

* Interaction

It is defined by the manner in which the components operate with each other.

For instance, in our project, the customer’s list must interact with the customer’s account in order to print the record of all the customers

* Interdependence

Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is required by another subsystem as input.

* Integration

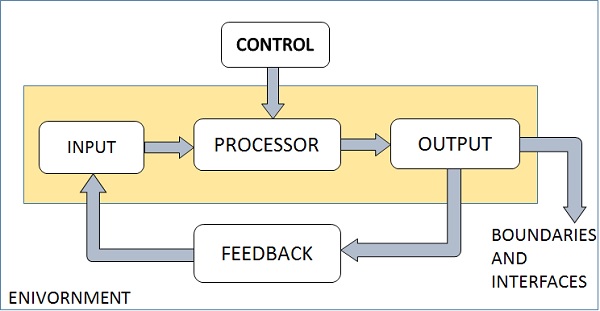
Integration is concerned with how a system component are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

* Central Objective

The objective of the system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.

**Elements of a System**

The following diagram shows the elements of a system



**Outputs and Inputs**

* The main aim of a system is to produce an output that is useful for its user.
* Inputs are the information that enters into the system for processing.
* The output is the outcome of processing.

**Processor(s)**

* The processor is the element of a system that involves the actual transformation of input into an output.
* It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.
* As the output specifications change, so does the processing. In some cases, the input is also modified to enable the processor for handling the transformation.

**Control**

* The control element guides the system.
* It is the decision–making subsystem that controls the pattern of activities governing input, processing, and output.
* The behavior of a computer system is controlled by the Operating System and software. In order to keep the system in balance, what and how much input is needed is determined by Output Specifications.

**Feedback**

* Feedback provides control in a dynamic system.
* Positive feedback is routine in nature that encourages the performance of the system.
* Negative feedback is informational in nature that provides the controller with information for action.

**Environment**

* The environment is the “supersystem” within which an organization operates.
* It is the source of external elements that strike on the system.
* It determines how a system must function. For example, vendors and competitors of an organization’s environment may provide constraints that affect the actual performance of the business.

**Boundaries and Interface**

* A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.
* Each system has boundaries that determine its sphere of influence and control.
* The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

**SUBSYSTEM SERVICES**

**USER MANAGEMENT SUBSYSTEM:**

This subsystem is responsible for managing different users of the system by taking care of the information of different users. It provides functions to Check User Record and Check User List, this subsystem uses the services of the storage subsystem to store Account information. System administrators and all users of the system communicate with this subsystem. Operations provided by this subsystem are:

* see()
* Viewlist()

All the methods of this subsystem implement the print() service, to allow the customer/user to have a backup of her operation.

**ACCOUNT MANAGEMENT SUBSYSTEM:**   
This subsystem is responsible for managing user accounts. It provides functions for Opening an account, updating an account and Closing account. This subsystem uses the storage subsystem for storing the account’s information. Operations provided by this subsystem are:

* new\_acc()
* edit()
* erase()

All the methods of this subsystem implement the print() service, to  
allow the customer to have a backup of her operation.

**TRANSACTION MANAGEMENT SUBSYSTEM:**  
This subsystem is responsible for managing the transactions of accounts. This subsystem provides all functions for managing a variety of transactions like deposit and withdrawal, the customer is the actors who communicate with this subsystem. This subsystem also communicates with the storage subsystem. Operations provided by this subsystem are:

* transact()

All the methods of this subsystem implement the print() service, to  
allow the customer to have a backup of her operation.

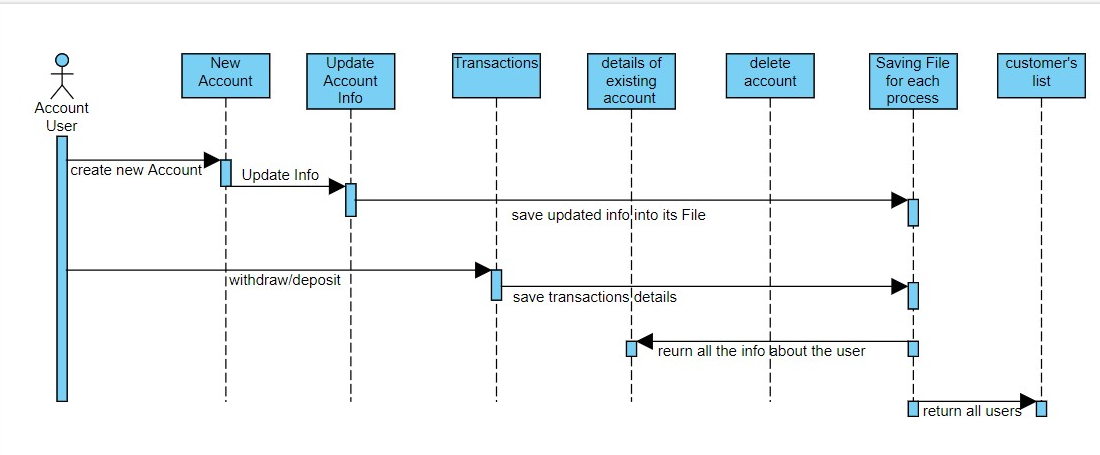
**STORAGE SUBSYSTEM:**  
This subsystem is responsible for specifying a common interface for the above subsystems for managing data. This subsystem is responsible for getting system-related data from different subsystems and issuing specific files for information storage and retrieval.

**SEQUENCE AND COLLABORATION DIAGRAMS**

**Sequence Diagram**

Sequence Diagram is an interaction diagram that details how operations are carried out, what messages are sent and when. Sequence diagrams are organized according to time. The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

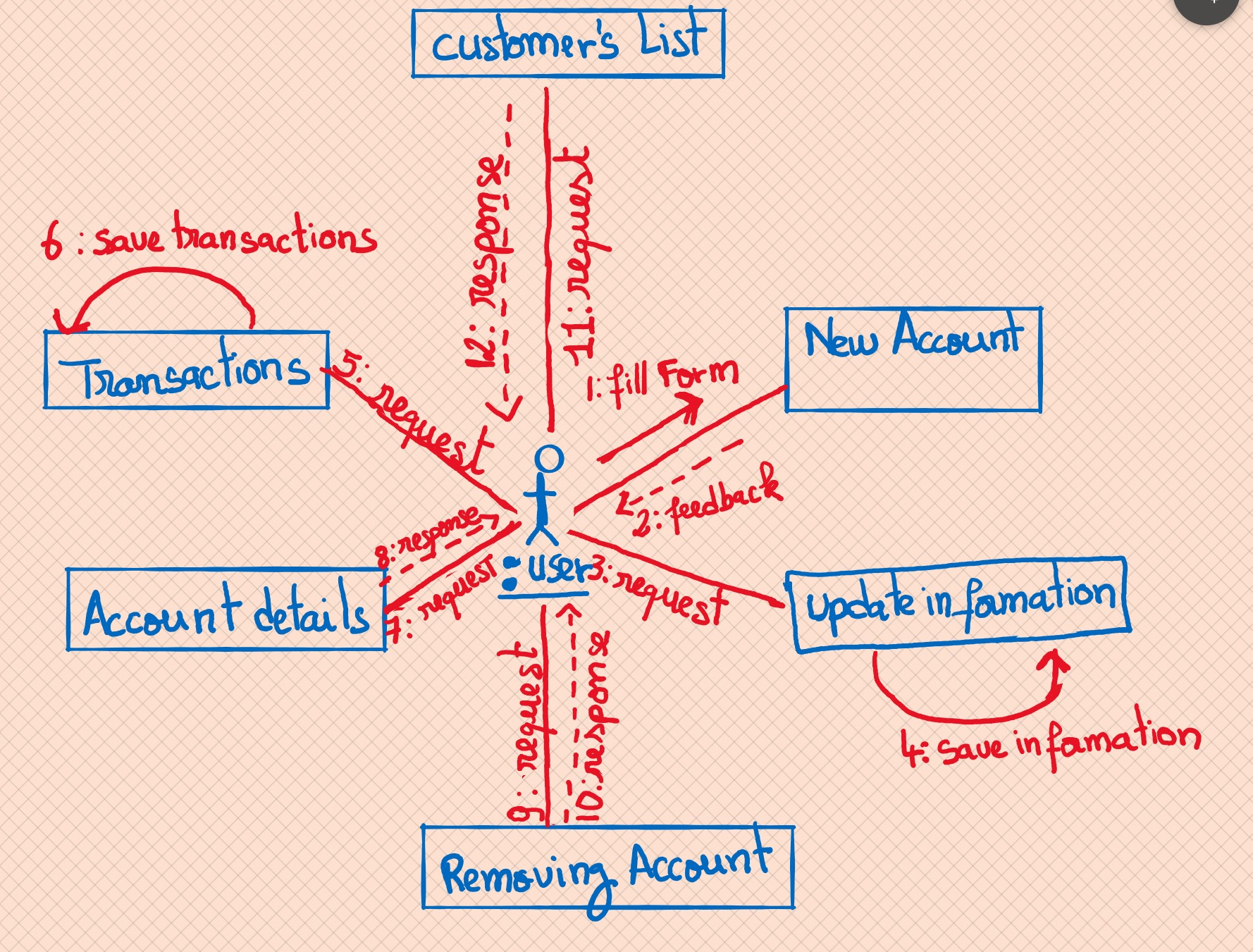
Below is the sequence diagram of our bank management system:



**Collaboration Diagram**

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case.  They are the primary source of information used in determining class responsibilities and interfaces. Collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that’s bigger than the sum of all its parts.

Below is the collaboration diagram of our bank management system:

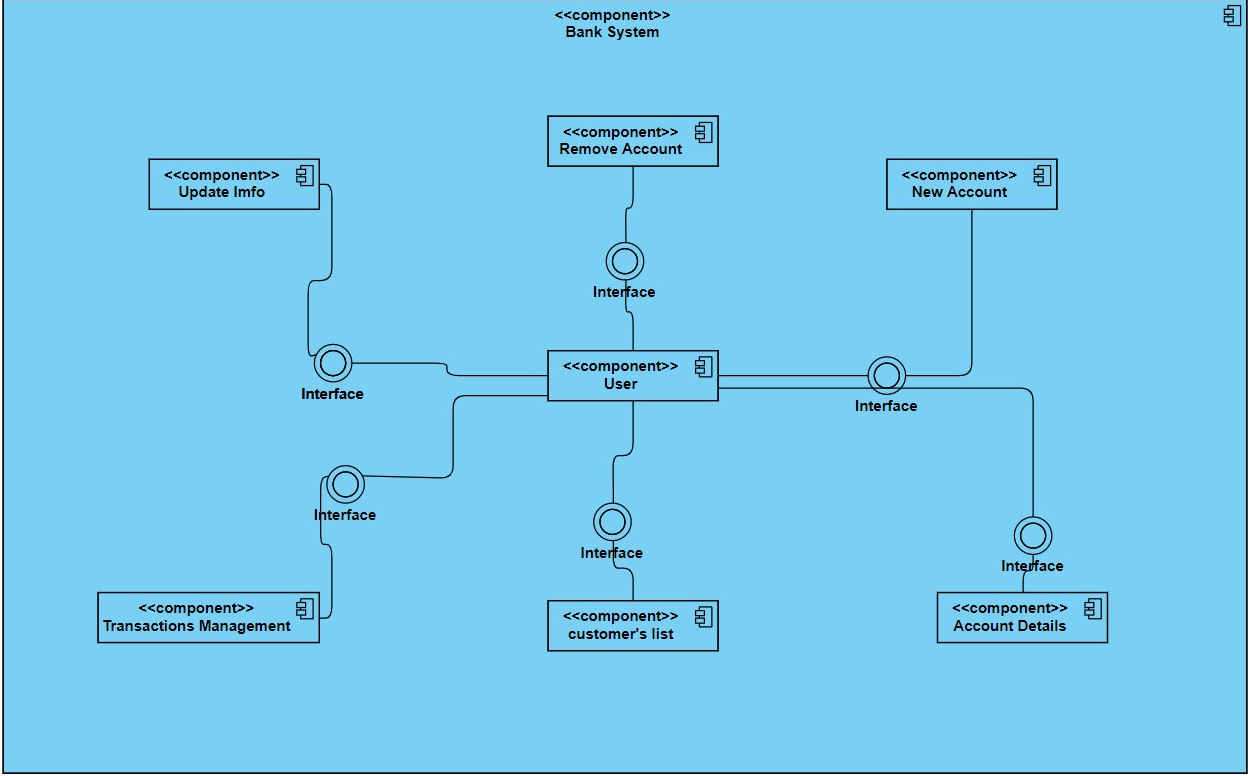


**Component Diagram:**

A component diagram provides the set of required interfaces that a component realizes or implements. These are the static diagrams of the unified modeling language. Component diagrams are used to represent the working and behavior of various components of a system.

In other words,

* A component is a replaceable and executable piece of a system.
* A component provides the set of required interfaces that a component realizes or implements.
* These are the static diagrams of the unified modeling language.
* It is a modular part of a system that encapsulates its contents
* Component diagrams are used to represent the working and behavior of various components of a system.
* Various components together make a single system.



**Deployment Diagram:**

A deployment diagram in the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) models the physical deployment of [artifacts](https://en.wikipedia.org/wiki/Artifact_(UML)) on [nodes](https://en.wikipedia.org/wiki/Node_(UML)). a deployment diagram would show what hardware components ("nodes") exist (e.g., a [web server](https://en.wikipedia.org/wiki/Web_server), an [application server](https://en.wikipedia.org/wiki/Application_server), and a [database server](https://en.wikipedia.org/wiki/Database_server)), what software components ("artifacts") run on each node (e.g., [web application](https://en.wikipedia.org/wiki/Web_application), [database](https://en.wikipedia.org/wiki/Database)), and how the different pieces are connected

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

1. Device Node
2. Execution Environment Node

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.

**TESTING AND IMPLEMENTATION**

|  |  |
| --- | --- |
|  | **Test Objective** |
| 1. | To check whether the program runs or not. |
| 2. | To check if the program menu displays all menu options or not. |
| 3. | To check if all options are working perfectly. |
| 4. | To check the account was created or not. |
| 5. | To check if the program adds amount after depositing. |
| 6. | To check if the program subtracts the amount after withdrawing. |
| 7. | To check if the account details option shows the info or not. |
| 8. | To check if Updating Account info works or not. |
| 9. | To check if removing account works or not |
| 10. | To check if the exit option works or not. |
| 11. | To check whether menu options return back or not. |
| 12. | To check whether account info is displayed after entering the correct account number or name. |
| 13. | To check if the color text is displayed or not. |

**Tests:**

|  |  |
| --- | --- |
| Test Case | 1 |
| Test Objective | To check whether the program runs or not. |
| Test Data | Running the program. |
| Expected Result | The main screen should display successfully. |
| Test Result | The main screen appears correctly. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 2 |
| Test Objective | To check if the program menu displays all menu options. |
| Test Data | Open the main menu. |
| Expected Result | All the options should be displayed correctly with serial order. |
| Test Result | Menu options are displayed in the right order |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 3 |
| Test Objective | To check if all options are working perfectly. |
| Test Data | Open menu. |
| Expected Result | All options are working. |
| Test Result | All options are working and displayed correctly. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 4 |
| Test Objective | To check if the account was created. |
| Test Data | Enter the account number. |
| Expected Result | The bank account information must be expected. |
| Test Result | The account was successfully created. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 5 |
| Test Objective | To check if the program adds amount after depositing. |
| Test Data | Deposit amount. |
| Expected Result | The amount must be added to the bank balance. |
| Test Result | The bank balance was updated successfully. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 6 |
| Test Objective | To check if the program subtracts the amount after withdrawing. |
| Test Data | Withdraw amount. |
| Expected Result | The amount must be subtracted from the bank balance. |
| Test Result | The bank balance was updated successfully. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 7 |
| Test Objective | To check if the account details option shows the info. |
| Test Data | View Account detail. |
| Expected Result | Details must be shown. |
| Test Result | Details appear correctly. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 8 |
| Test Objective | To check if Updating Account info works correctly. |
| Test Data | Updating Account Info. |
| Expected Result | The account address and phone number must be updated. |
| Test Result | Address and phone number updated successfully. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 9 |
| Test Objective | To check if removing account works. |
| Test Data | Delete Account. |
| Expected Result | Account must be deleted. |
| Test Result | Account deleted successfully. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 10 |
| Test Objective | To check if the exit option works or not. |
| Test Data | Run program. |
| Expected Result | Exit program. |
| Test Result | Program exits. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 11 |
| Test Objective | To check whether menu options return back or not. |
| Test Data | Run program. |
| Expected Result | Menu must be returned. |
| Test Result | Menu returns. |
| Conclusion | The expected result matches the actual result. |

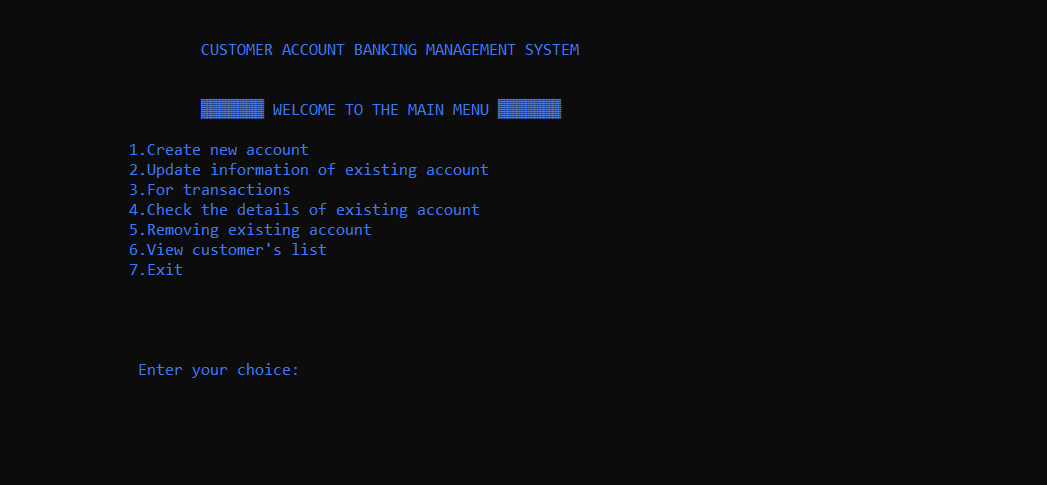
|  |  |
| --- | --- |
| Test Case | 12 |
| Test Objective | To check whether account info is displayed after entering the correct account number or name. |
| Test Data | Enter account number or name. |
| Expected Result | The information must be displayed. |
| Test Result | information displays. |
| Conclusion | The expected result matches the actual result. |

|  |  |
| --- | --- |
| Test Case | 13 |
| Test Objective | To check if the colored text is displayed or not. |
| Test Data | Run program. |
| Expected Result | The colored text should be seen. |
| Test Result | The colored text is seen. |
| Conclusion | The expected result matches the actual result. |

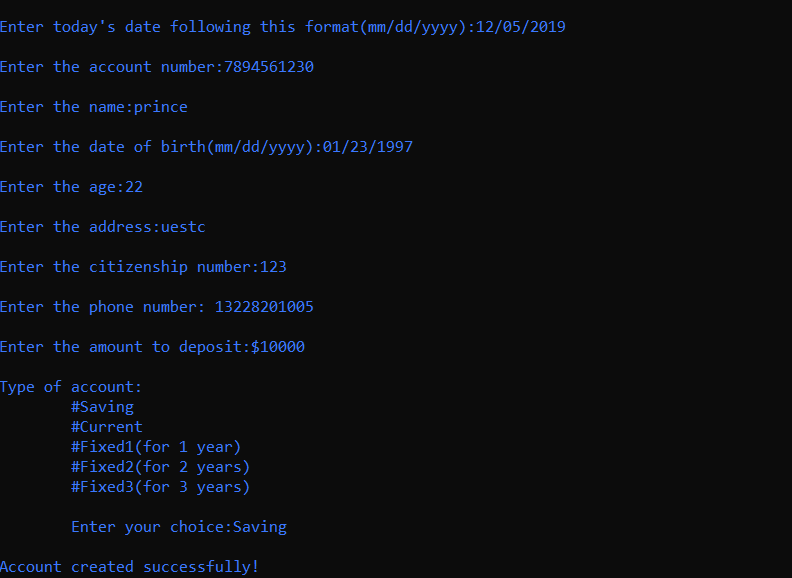
**GUI**

**OUTPUT RESULTS**

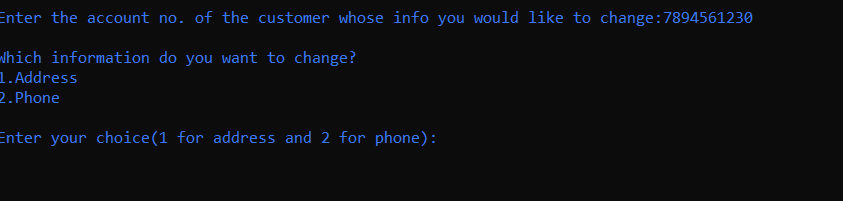
* Welcome screen

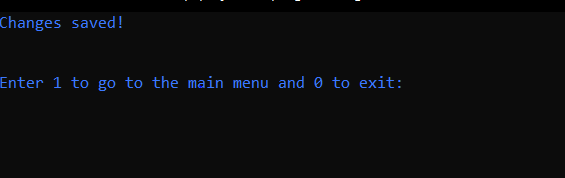


1. **Create a new account**

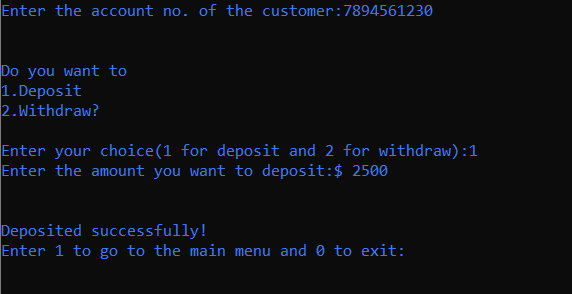


1. **Update information of existing account**

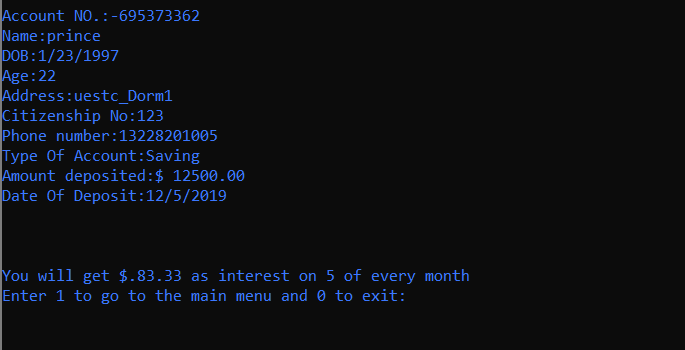




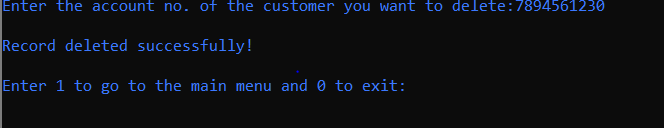
1. **For transactions**



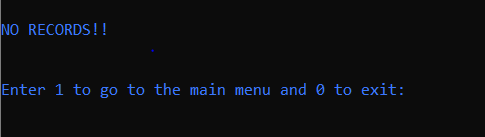
1. **Check details of the existing account**



1. **Removing existing account**



1. **View customer’s list**



1. **Exit**

which would just close the program

**Coding**

#include<stdio.h>

#include<stdlib.h>

#include<windows.h>

int i,j;

int main\_exit;

void menu();

struct date{ // structure for the date

int month,day,year;

};

struct { // structure for variables and arrays

char name[60];

int acc\_no,age;

char address[60];

char citizenship[15];

double phone;

char acc\_type[10];

float amt;

struct date dob;

struct date deposit;

struct date withdraw;

}add,upd,check,rem,transaction; // variables that would be use to call each entities inside the struct

float interest(float t,float amount,int rate) // we define the function that calculates the interest rate of the user

{

float SI;

SI=(rate\*t\*amount)/100.0;

return (SI);

}

void fordelay(int j) //delay function is used to suspend execution of a program for a particular time.

{ int i,k;

for(i=0;i<j;i++)

k=i;

}

void new\_acc()

{

int choice;

FILE \*p;

p=fopen("record.dat","a+"); // here we create a file in both reading and writing mode, that file would save the details about each accounts

account\_no:

system("cls"); // clear screen

printf("\t\t\t\xB2\xB2\xB2\ ADD RECORD \xB2\xB2\xB2\xB2");

printf("\n\n\nEnter today's date following this format(mm/dd/yyyy):");

scanf("%d/%d/%d",&add.deposit.month,&add.deposit.day,&add.deposit.year);

printf("\nEnter the account number:");

scanf("%d",&check.acc\_no);

while(fscanf(p,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if (check.acc\_no==add.acc\_no) // this function cheecks if the acc we are trying to create has not been created already

{printf("Account no. already in use!");

fordelay(1000000000);

goto account\_no;

}

}

add.acc\_no=check.acc\_no;

printf("\nEnter the name:");

scanf("%s",add.name);

printf("\nEnter the date of birth(mm/dd/yyyy):");

scanf("%d/%d/%d",&add.dob.month,&add.dob.day,&add.dob.year);

printf("\nEnter the age:");

scanf("%d",&add.age);

printf("\nEnter the address:");

scanf("%s",add.address);

printf("\nEnter the citizenship number:");

scanf("%s",add.citizenship);

printf("\nEnter the phone number: ");

scanf("%lf",&add.phone);

printf("\nEnter the amount to deposit:$");

scanf("%f",&add.amt);

printf("\nType of account:\n\t#Saving\n\t#Current\n\t#Fixed1(for 1 year)\n\t#Fixed2(for 2 years)\n\t#Fixed3(for 3 years)\n\n\tEnter your choice:");

scanf("%s",add.acc\_type);

fprintf(p,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

fclose(p);

printf("\nAccount created successfully!");

add\_invalid:

printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else if(main\_exit==0)

close();

else

{

printf("\nInvalid!\a");

goto add\_invalid;

}

}

void view\_list() // this function checks the registered accounts

{

FILE \*view;

view=fopen("record.dat","r");

int test=0;

system("cls");

printf("\nACC. NO.\tNAME\t\t\tADDRESS\t\t\tPHONE\n");

while(fscanf(view,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

printf("\n%6d\t %10s\t\t\t%10s\t\t%.0lf",add.acc\_no,add.name,add.address,add.phone);

test++;

}

fclose(view);

if (test==0)

{ system("cls");

printf("\nNO RECORDS!!\n");}

view\_list\_invalid:

printf("\n\nEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else if(main\_exit==0)

close();

else

{

printf("\nInvalid!\a");

goto view\_list\_invalid;

}

}

void edit(void) // this function edits the customer's info

{

int choice,test=0;

FILE \*old,\*newrec;

old=fopen("record.dat","r");

newrec=fopen("new.dat","w");

printf("\nEnter the account no. of the customer whose info you would like to change:");

scanf("%d",&upd.acc\_no);

while(fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if (add.acc\_no==upd.acc\_no)

{ test=1;

printf("\nWhich information do you want to change?\n1.Address\n2.Phone\n\nEnter your choice(1 for address and 2 for phone):");

scanf("%d",&choice);

system("cls");

if(choice==1)

{printf("Enter the new address:");

scanf("%s",upd.address);

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,upd.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

system("cls");

printf("Changes saved!");

}

else if(choice==2)

{

printf("Enter the new phone number:");

scanf("%lf",&upd.phone);

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,upd.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

system("cls");

printf("Changes saved!");

}

}

else

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

}

fclose(old);

fclose(newrec);

remove("record.dat");

rename("new.dat","record.dat");

if(test!=1)

{ system("cls");

printf("\nRecord not found!!\a\a\a");

edit\_invalid:

printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else if (main\_exit==2)

close();

else if(main\_exit==0)

edit();

else

{printf("\nInvalid!\a");

goto edit\_invalid;}

}

else

{printf("\n\n\nEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else

close();

}

}

void transact(void) //function for deposit and withdraw

{ int choice,test=0;

FILE \*old,\*newrec;

old=fopen("record.dat","r");

newrec=fopen("new.dat","w");

printf("Enter the account no. of the customer:");

scanf("%d",&transaction.acc\_no);

while (fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if(add.acc\_no==transaction.acc\_no)

{ test=1;

if(strcmpi(add.acc\_type,"fixed1")==0||strcmpi(add.acc\_type,"fixed2")==0||strcmpi(add.acc\_type,"fixed3")==0)

{

printf("\a\a\a\n\nYOU CANNOT DEPOSIT OR WITHDRAW CASH IN FIXED ACCOUNTS!!!!!");

fordelay(1000000000);

system("cls");

menu();

}

printf("\n\nDo you want to\n1.Deposit\n2.Withdraw?\n\nEnter your choice(1 for deposit and 2 for withdraw):");

scanf("%d",&choice);

if (choice==1)

{

printf("Enter the amount you want to deposit:$ ");

scanf("%f",&transaction.amt);

add.amt+=transaction.amt;

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

printf("\n\nDeposited successfully!");

}

else

{

printf("Enter the amount you want to withdraw:$ ");

scanf("%f",&transaction.amt);

add.amt-=transaction.amt;

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

printf("\n\nWithdrawn successfully!");

}

}

else

{

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

}

}

fclose(old);

fclose(newrec);

remove("record.dat");

rename("new.dat","record.dat");

if(test!=1)

{

printf("\n\nRecord not found!!");

transact\_invalid:

printf("\n\n\nEnter 0 to try again,1 to return to main menu and 2 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==0)

transact();

else if (main\_exit==1)

menu();

else if (main\_exit==2)

close();

else

{

printf("\nInvalid!");

goto transact\_invalid;

}

}

else

{

printf("\nEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else

close();

}

}

void erase(void) // function that deletes users info

{

FILE \*old,\*newrec;

int test=0;

old=fopen("record.dat","r");

newrec=fopen("new.dat","w");

printf("Enter the account no. of the customer you want to delete:");

scanf("%d",&rem.acc\_no);

while (fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if(add.acc\_no!=rem.acc\_no)

fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

else

{test++;

printf("\nRecord deleted successfully!\n");

}

}

fclose(old);

fclose(newrec);

remove("record.dat");

rename("new.dat","record.dat");

if(test==0)

{

printf("\nRecord not found!!\a\a\a");

erase\_invalid:

printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");

scanf("%d",&main\_exit);

if (main\_exit==1)

menu();

else if (main\_exit==2)

close();

else if(main\_exit==0)

erase();

else

{printf("\nInvalid!\a");

goto erase\_invalid;}

}

else

{printf("\nEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else

close();

}

}

void see(void) // function that returns all the info about the customer's account including the interest rate

{

FILE \*p;

int test=0,rate;

int choice;

float time;

float intrst;

p=fopen("record.dat","r");

printf("Do you want to check by\n1.Account no\n2.Name\nEnter your choice:");

scanf("%d",&choice);

if (choice==1)

{ printf("Enter the account number:");

scanf("%d",&check.acc\_no);

while (fscanf(p,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if(add.acc\_no==check.acc\_no)

{ system("cls");

test=1;

printf("\nAccount NO.:%d\nName:%s \nDOB:%d/%d/%d \nAge:%d \nAddress:%s \nCitizenship No:%s \nPhone number:%.0lf \nType Of Account:%s \nAmount deposited:$ %.2f \nDate Of Deposit:%d/%d/%d\n\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,

add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

if(strcmpi(add.acc\_type,"fixed1")==0)

{

time=1.0;

rate=9;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+1);

}

else if(strcmpi(add.acc\_type,"fixed2")==0)

{

time=2.0;

rate=11;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+2);

}

else if(strcmpi(add.acc\_type,"fixed3")==0)

{

time=3.0;

rate=13;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+3);

}

else if(strcmpi(add.acc\_type,"saving")==0)

{

time=(1.0/12.0);

rate=8;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d of every month",intrst,add.deposit.day);

}

else if(strcmpi(add.acc\_type,"current")==0)

{

printf("\n\nYou will get no interest\a\a");

}

}

}

}

else if (choice==2)

{ printf("Enter the name:");

scanf("%s",&check.name);

while (fscanf(p,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d",&add.acc\_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&add.age,add.address,add.citizenship,&add.phone,add.acc\_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)

{

if(strcmpi(add.name,check.name)==0)

{ system("cls");

test=1;

printf("\nAccount No.:%d\nName:%s \nDOB:%d/%d/%d \nAge:%d \nAddress:%s \nCitizenship No:%s \nPhone number:%.0lf \nType Of Account:%s \nAmount deposited:$%.2f \nDate Of Deposit:%d/%d/%d\n\n",add.acc\_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,

add.acc\_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);

if(strcmpi(add.acc\_type,"fixed1")==0)

{

time=1.0;

rate=9;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+1);

}

else if(strcmpi(add.acc\_type,"fixed2")==0)

{

time=2.0;

rate=11;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+2);

}

else if(strcmpi(add.acc\_type,"fixed3")==0)

{

time=3.0;

rate=13;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+3);

}

else if(strcmpi(add.acc\_type,"saving")==0)

{

time=(1.0/12.0);

rate=8;

intrst=interest(time,add.amt,rate);

printf("\n\nYou will get $.%.2f as interest on %d of every month",intrst,add.deposit.day);

}

else if(strcmpi(add.acc\_type,"current")==0)

{

printf("\n\nYou will get no interest\a\a");

}

}

}

}

fclose(p);

if(test!=1)

{ system("cls");

printf("\nRecord not found!!\a\a\a");

see\_invalid:

printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");

scanf("%d",&main\_exit);

system("cls");

if (main\_exit==1)

menu();

else if (main\_exit==2)

close();

else if(main\_exit==0)

see();

else

{

system("cls");

printf("\nInvalid!\a");

goto see\_invalid;}

}

else

{printf("\nEnter 1 to go to the main menu and 0 to exit:");

scanf("%d",&main\_exit);}

if (main\_exit==1)

{

system("cls");

menu();

}

else

{

system("cls");

close();

}

}

void close(void)

{

printf("\n\n\n\nThis Bank Management System is developed by Franck Armand!");

}

void menu(void) // main menu function

{ int choice;

system("cls");

system("color 9");

printf("\n\n\t\t\tCUSTOMER ACCOUNT BANKING MANAGEMENT SYSTEM");

printf("\n\n\n\t\t\t\xB2\xB2\xB2\xB2\xB2\xB2\xB2 WELCOME TO THE MAIN MENU \xB2\xB2\xB2\xB2\xB2\xB2\xB2");

printf("\n\n\t\t1.Create new account\n\t\t2.Update information of existing account\n\t\t3.For transactions\n\t\t4.Check the details of existing account\n\t\t5.Removing existing account\n\t\t6.View customer's list\n\t\t7.Exit\n\n\n\n\n\t\t Enter your choice:");

scanf("%d",&choice);

system("cls");

switch(choice)

{

case 1:new\_acc(); // if user selects 1 go to new\_acc

break;

case 2:edit(); // if user selects 2 go to edit

break;

case 3:transact(); // if user select 3 go to transaction

break;

case 4:see(); // if user selects 4 go to See, to check user info

break;

case 5:erase(); // if user selects 5 go to delete user record

break;

case 6:view\_list(); // if user selects 6 go to view customer's list

break;

case 7:close(); // if user selects 7 , exit

break;

}

}

int main()

{

menu();

}

**CONCLUSION**

*Program Weakness:*  
 As we know, programs can be 100% reliable and efficient. So, there are also some drawbacks from my system which are as follows:

It cannot perform all the required functions as bank required, it’s simply a record of customer’s account alongside simple bank operations such as deposit, withdraw and calculate the interest rate.  
 It’s not a multitasking program. It can’t perform the various task at a single time.

*Program Strength:*  
There are many advantages of using this program as it contains various features like,  
 It is actually a user-friendly software, as it is just easy to use by just following the instruction which is appeared on the screen.  
 This program needs the user account number to access user information so that only authorized users are only allowed to access through the internal main system.  
 Once a record has been saved, duplicate records can’t be made. All the record has a different account number so that there will not be any misplaced of the records entered.

**BIBLIOGRAPHY**

**BIBLIOGRAPHY**

* Reference websites:

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

* Books:
  + *The C Programming Language (2nd Ed.)*
  + *The C Programming Language-Pearson Education-Dennis M. Ritchie, Brian W. Kernighan-2015-EDN*