C O N T E N T S

1. Introduction

This software project is developed to automate the functionalities of a bank. The purpose of the software project is to develop the Management Information System (MIS) to automate the record keeping of customer entry, account entry, transaction entry and loan entry with a view to enhance the decision making of the functionaries.

A MIS mainly consists of a computerized database, a collection of interrelated tables for a particular subject or purpose, capable of producing different reports relevant to the user. An application program is tied with the database for easy access and interface to the database. Using an application program or front-end, we can store, retrieve and manage all information in a proper way.

This software, being simple in design and working, does not require much training to users, and can be used as a powerful tool for automating a Bank System.

During coding and design of the software Project, Python IDLE, a powerful front-end tool is used for getting Command line based integrated platform and coding simplicity. As a back-end a powerful, open-source RDBMS, MySQL is used.

2. Objective & Scope of the Project

The objective of the software project is to develop a computerized MIS to automate the functions of a bank. This software project is also aimed to enhance the current record keeping system, which will help customers to retrieve the up-to-date information at the right time in the right shape.

The proposed software system is expected to do the following functionality-

- The proposed system should maintain all the records and transactions, and should generate the required reports and information when required.
- To provide command line interface to interact with a centralized database
- To identify the critical operation procedure and possibilities of simplification using modern IT tools and practices.

In its current scope, the software enables users to retrieve and update the information from a centralized database designed with MySQL. This software does not require much training time for the users due to limited functionality and simplicity.

3. Theoretical Background

3.1 Python (Front End)

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The latest version of Python as of October 2020 is Python 3.9.0.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Features of Python Programming Language:

The main features of Python programming language are:

- Open Source and Free.
- Easy to Code
- Support for GUI.
- Object-Oriented Approach.
- High-Level Language.
- Integrated by Nature.
- Highly Portable.
- Highly Dynamic.

3.2 Database (Back End)

A database is a collection of information related to a particular subject or purpose, such as tracking customer orders or maintaining a music collection. Using any RDBMS application software like MS SQL Server, MySQL, Oracle, Sybase etc, you can manage all your information from a single database file. Within the file, divide your data into separate storage containers called tables. You may retrieve the data using queries.

A table is a collection of data about a specific topic, such as products or suppliers. Using a separate table for each topic means you can store that data only once, which makes your database more efficient and reduces data-entry errors. Table organizes data into columns (called fields) and rows (called records).

A Primary key is one or more fields whose value or values uniquely identify each record in a table. In a relationship, a primary key is used to refer to specific records in one table from another table. A primary key is called foreign key when it is referred to from another table.

To find and retrieve just the data that meets conditions you specify, including data from multiple tables, create a query. A query can also update or delete multiple records at the same time and perform built-in or custom calculations on your data.

Most of the database management systems have the following capabilities:

- Creating a table, addition, deletion, modification of records.
- Retrieving data collectively or selectively.
- Various reports can be produced from the system. These may be either standardized reports or that may be specifically generated according to specific user definition.
- Mathematical functions can be performed, and the data stored in the database can be manipulated with these functions to perform the desired calculations.
- To maintain data integrity and database use.

3.3 Python-Database Connectivity:

While designing real life applications, certain situations arise pertaining to storing some important and necessary information by the user. Usually, the data inputted by the user along with generated output are displayed but are not stored, since all the program execution takes place inside the RAM, which is a temporary memory and as soon as we close the form, its contents (form input and generated output) get erased. They can't be retrieved since they are not getting saved on the hard disk (or any other secondary storage device). Thus, when the application is executed for a second time, it requires a new set of inputs from the user. This limitation can be overcome by sending the output generated and saving the input fetched from the user in a database created at the back end of the application. The input is fetched from the user using the Python interface. This is termed as the Front-End Interface of the application.

4. Problem Definition & Analysis

The hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirement. Defining and applying good, complete requirements are hard to work, and success in this endeavor has eluded many of us. Yet, we continue to make progress.

Problem definition describes the What of a system, not How. The quality of a software product is only as good as the process that creates it. Problem definition is one of the most crucial steps in this creation process. Without defining a problem, developers do not know what to build, customers do not know what to expect, and there is no way to validate that the built system satisfies the requirement.

Problem definition and Analysis is the activity that encompasses learning about the problem to be solved, understanding the needs of customers and users, trying to find out who the user really is, and understanding all the constraints on the solution. It includes all activities related to the following:

- Identification and documentation of customer's or user's needs.
- Creation of a document that describes the external behavior and the association constraints that will satisfy those needs.
- Analysis and validation of the requirements documents to ensure consistency, completeness, and feasibility
- Evolution of needs.

After the analysis of the functioning of a bank management system, the proposed System is expected to do the following: -

- To provide a user friendly, command line based integrated and centralized environment for computerized bank management systems.
- The proposed system should maintain all the records and transactions.
- To provide an efficient and secured Information storage, flow and retrieval system, ensuring the integrity and validity of records.
- To identify the critical operation procedure and possibilities of simplification using modern IT tools and practices.

5. System Implementation

5.1 The Hardware used:

While developing the system, the used hardware are:

- PC with intel i3 processor having 4GB RAM, SVGA and other required devices.

5.2 The Software used:

- Microsoft Windows® 10 as Operating System.
- Python IDLE as Front-end Development environment.
- MySQL as Back-end Server with Database.
- MS-Word 2000 for documentation.

6. System Design & Development

6.1 Database Design:

An important aspect of system design is the design of data storage structure. To begin with, a logical model of data structure is developed first. A database is a container object which contains tables, queries, reports and data validation policies, enforcement rules or constraints etc. Logical data often represented as a record are kept in different tables after reducing anomalies and redundancies. The goodness of database design lies in the table structure and its relationship.

This software project maintains a database named bms which contains the following tables.

Table Design:

The database of the bank management system contains 5 tables. The tables and their structure are given below.

Table: acc

Field	Type	Null	Key	Default	Extra
acc_no	char(10)	NO	PRI	NULL	-
dos	date	NO	-	NULL	-
balance	decimal(30,2)	YES	-	0.00	-

Table: cdet

Field	Type	Null	Key	Default	Extra
cid	char(5)	NO	PRI	NULL	-
name	varchar(25)	NO	-	NULL	-
phone	char(10)	NO	-	NULL	-
pan_card	char(10)	NO	MUL	NULL	REGEXP '[A-Z][A-Z][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][A-Z]'
address	varchar(50)	YES	-	NULL	-
email	varchar(25)	YES	-	NULL	-

account1	char(10)	NO	-	NULL	-
account2	char(10)	YES	-	NULL	-

Table: idp

Field	Type	Null	Key	Default	Extra
id	char(5)	NO	PRI	NULL	FOREIGN KEY REFERENCES CDET(CID)
password	varchar(15)	NO	-	NULL	check(length(password) > 8)

Table: loan

Field	Type	Null	Key	Default	Extra
start	date	NO	-	NULL	-
stop	date	NO	-	NULL	-
amount	decimal(30,2)	NO	-	NULL	-
acc_no	char(10)	NO	PRI	NULL	-

Table: tr

Field	Type	Null	Key	Default	Extra
acc_no	char(10)	NO	-	NULL	-
date	date	NO	-	NULL	-
time	varchar(10)	NO	-	NULL	-
withdraw	varchar(3)	YES	-	NULL	-
deposit	varchar(3)	YES	-	NULL	-
to_acc_no	char(10)	YES	-	NULL	-
amount	decimal(30,2)	YES	-	NULL	-

6.2 Package/Module used:

1) mysql.connector

2) random

3) datetime

6.3 Functions Defined

1.execute(): - This function is used to run the queries.

2.create_id():- This function is used to create account number and customer id.

3.det():- This function is used to display customer details.

4.withdraw():- This function is used to withdraw an amount from an account.

5.deposit():- This function is used to deposit amounts to an account.

6.transfer():- This function is used to transfer amounts to another account.

7.loan():- This function is used to take a loan.

8.loan_back():- This function is used to pay back a loan.

9.update():- This function is used to update customer details.

10.check_acc():- This function is used to check account status (loan , balance etc).

11.vtr():- This function is used to view transactions from all the accounts of a customer.

12.new_acc():- This function is used to create a new account.

13.del_acc():- This function is used to delete an account.

14.dashboard():- This function is used to display a dashboard.

15.options():- This function is used to display options.

7. Coding:

```
import mysql.connector as m
from random import randint as randi
from datetime import datetime as dt, timedelta as td
greeting_words = ('hello','hi','nice to meet you','hola','salut')

def execute(query,*values,state='v'):
    values = list(values)
    for i in range(values.count('')):
        a = values.index('')
        values[a] = None
    values = tuple(values)
    co = m.connect(host = 'localhost', passwd = 'python',user = 'root', database = 'bms')
    cur = co.cursor()
    cur.execute(query,values)
    if state == 'dml':
        co.commit()
        co.close()
    else:
        d = cur.fetchall()
        co.close()
        return d

def create_id(r,a='AC'):   # ac = acc no(10) and ci = cust id(5) and ma = manag id(5)
    if a == 'AC':
```

```
        data = execute('select acc_no from acc')
    else:
        data = execute('select id from idp')
    maxr = int('9'*(r-2))
    while True:
        b = str(randi(1,maxr))
        while len(b) <(r-2):
            b = '0'+b
        b = a + b
        if (b,) not in data:
            break
    return b

def det(cust):
    data = execute('select * from cdet where cid=%s',cust)
    cid,n,ph,pan,ad,em,ac1,ac2 = data[0]
    print('\t\t\t\t\t\t Name:',n)
    print('\t\t\t\t\t\t Customer id:',cid)
    print('\t\t\t\t\t\t Phone:',ph)
    print('\t\t\t\t\t\t Pan card:',pan)
    print('\t\t\t\t\t\t Email:',em)
    print('\t\t\t\t\t\t Address:',ad)
    print('\t\t\t\t\t\t Account 1:',ac1)
    print('\t\t\t\t\t\t Account 2:',ac2)

def withdraw(cust,rem=1):
```

```
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    if None not in acc:
        ch = input(f'\t\t\t\twhich account you would like to withdraw from {acc[0]} or {acc[1]}: ')
        assert (ch in acc)
    else:
        ch = acc[0]
        print(f'Withdrawing from {acc[0]}...'.center(134))
    am = float(execute('select balance from acc where acc_no=%s',ch)[0][0])
    if rem == 0:
        execute('update acc set balance=balance-balance where acc_no=%s',ch,state='dml')
        return None
    wi = float(input('\t\t\t\t\t\tEnter amount to withdraw: '))
    assert (wi>0)
    if am-wi > 5000:
        execute('insert into tr(acc_no,date,time,withdraw,amount) values(%s,date(now()),time(now()),%s,%s)',ch,'YES',wi,state='dml')
        execute('update acc set balance=balance-%s where acc_no=%s',wi,ch,state='dml')
        print('successfully withdrawn'.center(134))
    elif wi>am:
        print('Sorry but it exceeds the balance in your account'.center(134))
        raise AssertionError
    elif am-wi <= 5000:
        warni = input('\t\t\t\tWARNING: your remaining amount is less than 5000 are you sure to withdraw[y/n]: ')
        if warni == 'y':
```

```
            execute('insert into tr(acc_no,date,time,withdraw,amount)
values(%s,date(now()),time(now()),%s,%s)',ch,'YES',wi,state='dml')
            execute('update acc set balance=balance-%s where
acc_no=%s',wi,ch,state='dml')
            print('successfully withdrawn'.center(134))
        else:
            return None

def deposit(cust):
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    if None not in acc:
        ch = input(f'\t\t\t\twhich account you would like to deposit to {acc[0]} or
{acc[1]}: ')
        assert (ch in acc)
    else:
        ch = acc[0]
        print(f'Depositing to {acc[0]}...'.center(134))
    dep = float(input('\t\t\t\t\t\tEnter amount to deposit: '))
    assert (dep>0)
    execute('insert into tr(acc_no,date,time,deposit,amount)
values(%s,date(now()),time(now()),%s,%s)',ch,'YES',dep,state='dml')
    execute('update acc set balance=balance+%s where
acc_no=%s',dep,ch,state='dml')
    print('successfully deposited'.center(134))

def transfer(cust):
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    if None not in acc:
```

```
        fr = input(f'\t\t\t\twhich account you would like to transfer from {acc[0]} or {acc[1]}: ')
        assert (fr in acc)
    else:
        fr = acc[0]
        print(f'Transfering from {acc[0]}...'.center(134))
    to = input('\t\t\t\t\t\tEnter the account to tranfer: ')
    amo = float(input('\t\t\t\t\t\tEnter amount to tranfer: '))
    assert (amo>0)
    balance = float(execute('select balance from acc where acc_no=%s',fr)[0][0])
    if balance > amo:
        acc_lst = execute('select acc_no from acc')
        if (to,) in acc_lst:
            execute('insert into tr(acc_no,date,time,to_acc_no,amount) values(%s,date(now()),time(now()),%s,%s)',fr,to,amo,state='dml')
            execute('update acc set balance=balance-%s where acc_no=%s',amo,fr,state='dml')
            execute('update acc set balance=balance+%s where acc_no=%s',amo,to,state='dml')
            print(f'successfully transferred to {to}'.center(134))
        else:
            print('account doesn\'t exist...'.center(134))
            raise AssertionError
    else:
        print('Sorry not enough balance...'.center(134))


def loan(cust):
```

```
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    loan = execute('select acc_no from loan')
    if None not in acc:
        fr = input(f'\t\t\t\twhich account you would like to take loan from {acc[0]} or {acc[1]}: ')
        assert (fr in acc)
    else:
        fr = acc[0]
        print(f'Using {acc[0]}...'.center(134))
    if (fr,) not in loan:
        amo = float(input('\t\t\t\t\t\tEnter amount for loan: '))
        assert (amo>0)
        d = int(amo*(1/100))
        execute('update acc set balance=balance+%s where acc_no = %s',amo,fr,state='dml')
        stop_date = dt.now() + td(days = d)
        execute('insert into loan (start,stop,amount,acc_no) values(%s,%s,%s,%s)',dt.now().date(),stop_date.date(),amo,fr,state='dml')
        print(f'loan of {amo} granted successfully'.center(134))
    else:
        print('you can\'t take more than one loan at a time...'.center(134))


def loan_back(cust):
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    loan = execute('select acc_no from loan')
    if (acc[0],) in loan or (acc[1],) in loan:
        if None not in acc:
```

```
            fr = input(f'\t\t\t\twhich account you would like to payback from {acc[0]} or {acc[1]}: ')
            assert (fr in acc)
        else:
            fr = acc[0]
            print(f'Using {acc[0]}...'.center(134))
        assert ((fr,) in loan)
        amo,s_date = execute('select amount,stop from loan where acc_no=%s',fr)[0]
        now = dt.now().date()
        fine = 0
        if s_date < now:
            fine = (now - s_date).days
            fine *= 200
        print(f'your amount to pay back is {amo} + fine of {fine}'.center(134))
        l_ch = input('\t\t\t\t\tDo you wan\'t to payback [y/n]: ')
        if l_ch == 'y':
            balance = float(execute('select balance from acc where acc_no=%s',fr)[0][0])
            if balance >= amo+fine:
                execute('update acc set balance = balance-%s where acc_no=%s',float(amo)+fine,fr,state='dml')
                execute('delete from loan where acc_no=%s',fr,state='dml')
                print('successfully done'.center(134))
            else:
                print('your account does\'nt have enough balance....'.center(134))
                return None
```

```
        else:
            return None
    else:
        print('hurrah!!! no loan to pay back'.center(134))


def update(cust):
    print('\t\t\t\t\t\t    [1] Update Name','\t\t\t\t\t\t    [2] Update Phone number',sep='\n')
    print('\t\t\t\t\t\t    [3] Update Address','\t\t\t\t\t\t    [4] Update pan card','\t\t\t\t\t\t    [5] Update email',sep='\n')
    print('\t\t\t\t\t\t    [6] Update Password')
    ch = input('\t\t\t\tWhat to update (just type numbers one after other if many): ')
    for i in ch:
        if ch.count(i) > 1:
            ch = ch.replace(i,'') + i
    ch = tuple(ch)
    for i in ch:
        i = int(i)
        if i == 1:
            n = input('\t\t\t\t\t New name: ')
            execute('update cdet set name = %s where cid = %s',n,cust,state='dml')
        elif i == 2:
            n = input('\t\t\t\t\t New Phone number: ')
            execute('update cdet set phone = %s where cid = %s',n,cust,state='dml')
        elif i == 3:
            n = input('\t\t\t\t\t New Address: ')
```

```
            execute('update cdet set address = %s where cid = %s',n,cust,state='dml')
        elif i == 4:
            n = input('\t\t\t\t\t New pan card: ')
            execute('update cdet set pan_card = %s where cid =
%s',n,cust,state='dml')
        elif i == 5:
            n = input('\t\t\t\t\t New email: ')
            execute('update cdet set email = %s where cid = %s',n,cust,state='dml')
        elif i == 6:
            n = input('\t\t\t\t\t New password: ')
            execute('update idp set password = %s where id = %s',n,cust,state='dml')
        else:
            return None
        print('done'.center(134))


def check_acc(cust):
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    for i in acc:
        if i != None:
            det = execute('select dos,balance from acc where acc_no=%s',i)[0]
            loan = execute('select start,stop,amount from loan where acc_no=%s',i)
            print('ACCOUNT STATUS'.center(134))
            print(f'\t\t\t\t\t\t    Account number: {i}',f'\t\t\t\t\t\t    Date of issue:
{det[0]}',sep='\n')
            print(f'\t\t\t\t\t\t    Balance: {det[1]}')
            if loan != []:
                loan = loan[0]
```

```
            print('LOAN STATUS'.center(134))
            print(f'\t\t\t\t\t\t    Date of issue: {loan[0]}',f'\t\t\t\t\t\t    Date of expiry: {loan[1]}',sep='\n')
            print(f'\t\t\t\t\t\t    Amount: {loan[2]}')
        else:
            print('LOAN STATUS'.center(134))
            print('NONE'.center(134))
        print()


def vtr(cust):
    acc = execute('select account1,account2 from cdet where cid=%s',cust)[0]
    for i in acc:
        if i != None:
            tr = execute('select acc_no,date,time,withdraw,deposit,to_acc_no,amount from tr where acc_no=%s',i)
            if tr != []:
                print('\t\t  Account number','Date','Time','Withdraw','Deposit','To','Amount',sep='    |    ')
                print(('-'*110).center(134))
                for j in tr:
                    print('\t\t  {:^18}| {} |{:^12}|{:^16}|{:^15}|{:^12}|{:^14}'.format(j[0],j[1],j[2],str(j[3]),str(j[4]),str(j[5]),j[6]))
            else:
                print(f'no transactions from {i}'.center(134))
                return None
            print(('-'*110).center(134),'\n\n')
```

```
def new_acc(cust):
    d = execute('select account1,account2 from cdet where cid=%s',cust)
    if None not in d[0]:
        print('Sorry you can\'t create more than 2 accounts'.center(134))
    else:
        print('creating account...'.center(134))
        acc2 = create_id(10)
        execute('update cdet set account2 = %s where
cid=%s',acc2,cust,state='dml')
        execute('insert into acc (acc_no,dos) values
(%s,date(now()))',acc2,state='dml')
        print('account created successfully'.center(134))
        print(f'your account no is {acc2}'.center(134))


def del_acc(cust):
    d = list(execute('select account1,account2 from cdet where cid=%s',cust)[0])
    out_of_loop = 0
    if None in d:
        b = d[0]
        a = input(f'\t\t\t\t\tAre you sure you wan\'t to delete {b} [y/n]: ')
        if a == 'y':
            loan = execute('select acc_no from loan where acc_no=%s',b)
            if loan != []:
                print('your loan is pending so...'.center(134))
                return None
            print('Please withdraw your entire amount...'.center(134))
```

```
            withdraw(cust,0)
            execute('delete from cdet where cid=%s',cust,state='dml')
            out_of_loop = 1
        else:
            return None
    else:
        b = input(f'\t\t\t\t\t\tDelete {d[0]} or {d[1]}: ')
        assert (b in d)
        a = input(f'\t\t\t\t\tAre you sure you wan\'t to delete {b} [y/n]: ')
        if a == 'y':
            d.remove(b)
            loan = execute('select acc_no from loan where acc_no=%s',b)
            if loan != []:
                print('your loan is pending so...'.center(134))
                return None
            balance = execute('select balance from acc where acc_no=%s',b)
            execute('update acc set balance=balance+%s where
acc_no=%s',balance[0][0],d[0],state='dml')
            execute('update cdet set account2=null where cid =%s',cust,state='dml')
            execute('update cdet set account1=%s where cid
=%s',d[0],cust,state='dml')
        else:
            return None
    execute('delete from acc where acc_no=%s',b,state='dml')
    execute('delete from tr where acc_no=%s or to_acc_no=%s',b,b,state='dml')
    print('account deleted successfully'.center(134))
    assert (out_of_loop == 0)
```

```
def dashboard():
    print()
    print('\t\t\t\t\t\t    [1] View your details','\t\t\t\t\t\t    [2] New Account','\t\t\t\t\t\t
[3] Close Account','\t\t\t\t\t\t    [4] Payback loan',sep='\n')
    print('\t\t\t\t\t\t    [5] Withdraw','\t\t\t\t\t\t    [6] Deposit','\t\t\t\t\t\t    [7]
Transfer','\t\t\t\t\t\t    [8] View all transactions',sep='\n')
    print('\t\t\t\t\t\t    [9] Loan','\t\t\t\t\t\t    [10] Update details','\t\t\t\t\t\t    [11]
Check Account Status',sep='\n')
    print('\t\t\t\t\t\t    [12] Log out','\t\t\t\t\t\t    [13] Exit',sep='\n')


def options(cust):
    while True:
        print()
        ch = int(input('\t\t\t\t\t\t    Enter your choice: '))
        print()
        print(('```'*20).center(134))
        if ch == 1:
            det(cust)
        elif ch == 2:
            new_acc(cust)
        elif ch == 3:
            del_acc(cust)
        elif ch == 4:
            loan_back(cust)
        elif ch == 5:
            withdraw(cust)
```

```
        elif ch == 6:
            deposit(cust)
        elif ch == 7:
            transfer(cust)
        elif ch == 8:
            vtr(cust)
        elif ch == 9:
            loan(cust)
        elif ch == 10:
            update(cust)
        elif ch == 11:
            check_acc(cust)
        elif ch == 12:
            break
        elif ch == 13:
            exit()
        print(('```'*20).center(134))
        dashboard()
print('$'*134)
print('Bank Management System'.center(134))
print('$'*134)
while True:
    try:
        print()
        ch = input('\t\t\t\t\t\t    Login[l] or Register[r]: ')
        print()
```

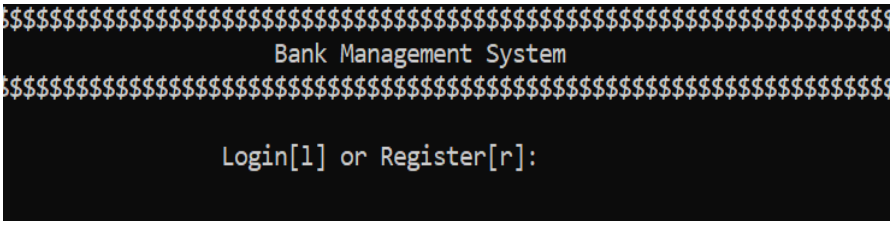
```
    if ch == 'l':
        cid = input('\t\t\t\t\t\t\tEnter id: ')
        p = input('\t\t\t\t\t\t\tEnter password: ')
        d = execute('select * from idp')
        if (cid,p) not in d:
            raise ValueError
    elif ch == 'r':
        n = input('\t\t\t\t\t\t\tName: ')
        p = input('\t\t\t\t\t\t\tPhone number: ')
        a = input('\t\t\t\t\t\t\tAddress: ')
        pan = input('\t\t\t\t\t\t\tPan card number: ')
        email = input('\t\t\t\t\t\t\tEmail id: ');print()
        cid = create_id(5,'CI')
        acc_no = create_id(10)
        print('\t\t\t\t\t\t\tYour id is',cid)
        passwd = input('\t\t\t\t\t\t\tSet Password: ')
        passwd = input('\t\t\t\t\t\t\tConfirm Password: ')
        print()
        try:
            execute('insert into cdet
(cid,name,phone,address,pan_card,account1,email) values
(%s,%s,%s,%s,%s,%s,%s)',cid,n,p,a,pan,acc_no,email,state = 'dml')
            execute('insert into idp (id,password) values (%s,%s)',cid,passwd,state
= 'dml')
            execute('insert into acc (acc_no,dos) values
(%s,date(now()))',acc_no,state='dml')
            print('Successfully created'.center(134))
```

```
            print(f'Your account number is {acc_no}'.center(134))
        except:
            raise Exception
    else:
        raise ValueError
    data = execute('select name from cdet where cid=%s',cid)
    n = data[0][0]
    print()
    print('~'*134)
    print()
    print((greeting_words[randi(0,len(greeting_words)-1)]+'
'+n.split()[0]).center(134))
    print()
    print('~'*134)
    dashboard()
    options(cid)

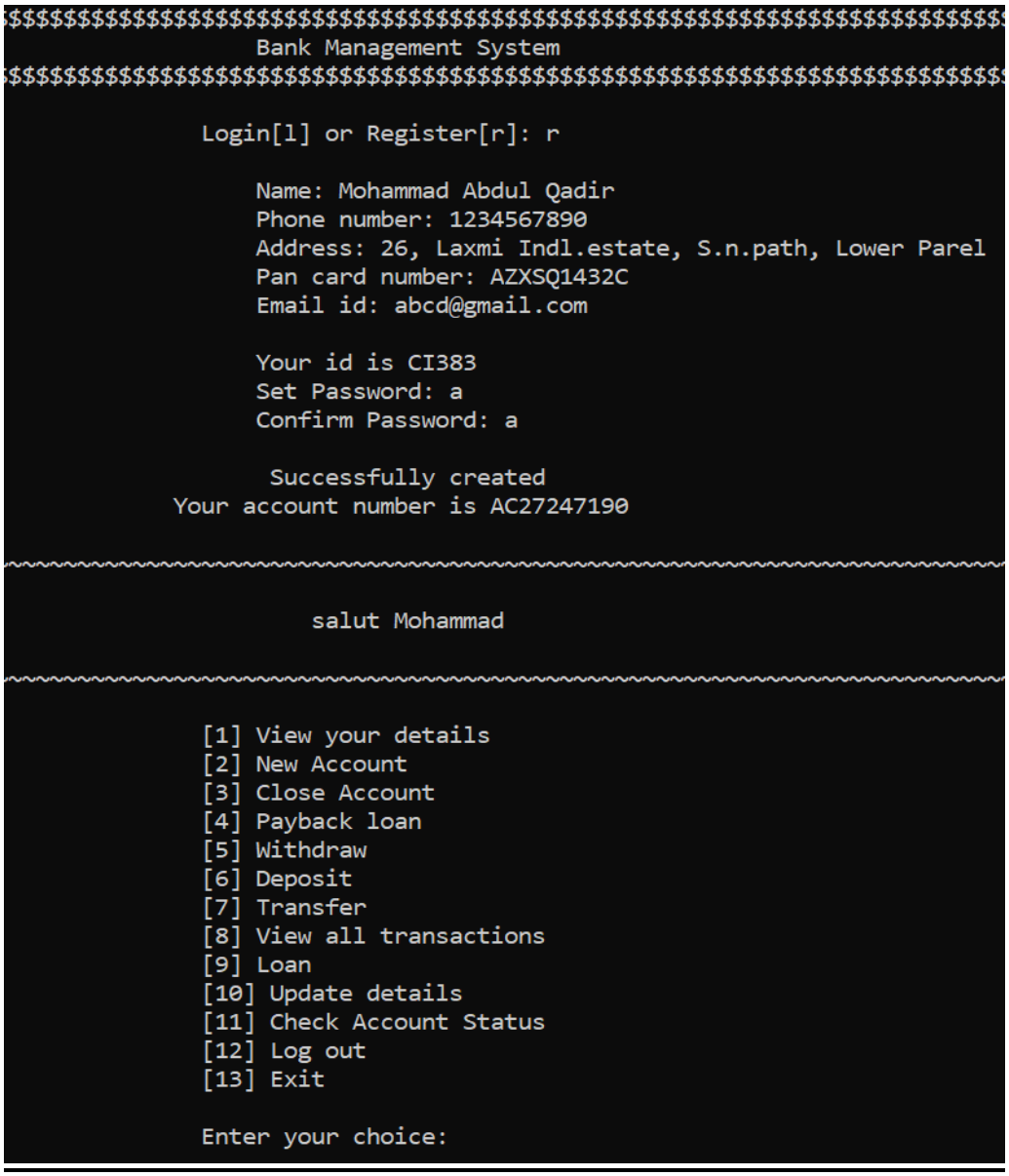
except ValueError:
    print('Please enter a valid value'.center(134))
except AssertionError:
    pass
except Exception as e:
    print (e)
    print('Oops something went wrong ...'.center(134))
    print('Please try again'.center(134))
```

8. Output Screen:

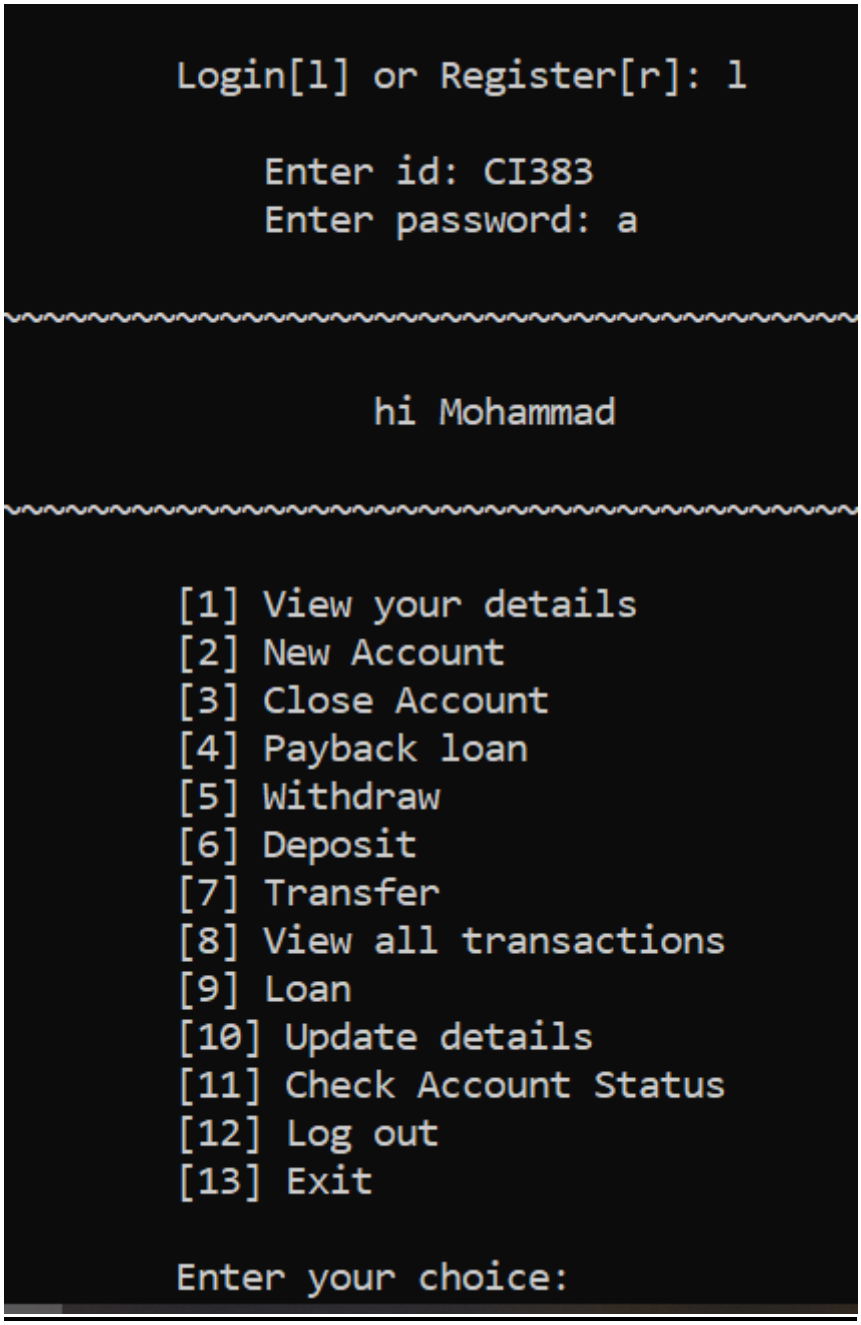
Main Screen



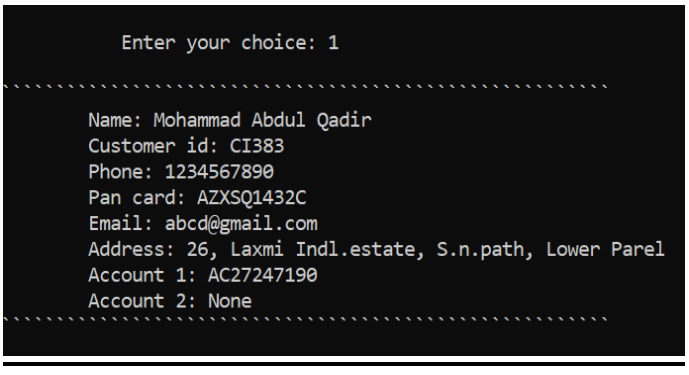
1)Registration



2)Login



3)Viewing customer details



4)Creating an account

```
Enter your choice: 2

creating account...
account created successfully
your account no is AC25424254
```

```
Enter your choice: 1

Name: Mohammad Abdul Qadir
Customer id: CI383
Phone: 1234567890
Pan card: AZXSQ1432C
Email: abcd@gmail.com
Address: 26, Laxmi Indl.estate, S.n.path, Lower Parel
Account 1: AC27247190
Account 2: AC25424254
```

5)Closing account

```
Enter your choice: 3

Delete AC27247190 or AC25424254: AC27247190
Are you sure you wan't to delete AC27247190 [y/n]: y
account deleted successfully
```

```
Enter your choice: 1

Name: Mohammad Abdul Qadir
Customer id: CI383
Phone: 1234567890
Pan card: AZXSQ1432C
Email: abcd@gmail.com
Address: 26, Laxmi Indl.estate, S.n.path, Lower Parel
Account 1: AC25424254
Account 2: None
```

6)Deposition

```
Enter your choice: 6

Depositing to AC25424254...
Enter amount to deposit: 1000
successfully deposited
```

```
Enter your choice: 11

ACCOUNT STATUS
Account number: AC25424254
Date of issue: 2021-07-08
Balance: 1000.00
LOAN STATUS
NONE
```

7)Withdrawl

```
Enter your choice: 5

Withdrawing from AC25424254...
Enter amount to withdraw: 100
WARNING: your remaining amount is less than 5000 are you sure to withdraw[y/n]: y
successfully withdrawn
```

```
Enter your choice: 11

ACCOUNT STATUS
Account number: AC25424254
Date of issue: 2021-07-08
Balance: 900.00
LOAN STATUS
NONE
```

8)Transfer to another account

```
Enter your choice: 7

Transfering from AC25424254...
Enter the account to tranfer: AC75003177
Enter amount to tranfer: 200
successfully transferred to AC75003177
```

```
Enter your choice: 11

ACCOUNT STATUS
Account number: AC25424254
Date of issue: 2021-07-08
Balance: 700.00
LOAN STATUS
NONE
```

```
ACCOUNT STATUS
Account number: AC75003177
Date of issue: 2021-07-03
Balance: 200.00
LOAN STATUS
NONE
```

9)View Transactions

```
Enter your choice: 8

Account number | Date       | Time     | Withdraw | Deposit | To        | Amount
---------------------------------------------------------------------------------
AC25424254     | 2021-07-08 | 19:16:13 | None     | YES     | None      | 1000.00
AC25424254     | 2021-07-08 | 19:18:10 | YES      | None    | None      | 100.00
AC25424254     | 2021-07-08 | 19:21:17 | None     | None    | AC75003177| 200.00
---------------------------------------------------------------------------------
```

10)Loan

```
Enter your choice: 9

Using AC25424254...
Enter amount for loan: 100
loan of 100.0 granted successfully
```

```
Enter your choice: 11

ACCOUNT STATUS
Account number: AC25424254
Date of issue: 2021-07-08
Balance: 800.00
LOAN STATUS
Date of issue: 2021-07-08
Date of expiry: 2021-07-09
Amount: 100.00
```

11)Loan deposition

```
Enter your choice: 4

Using AC25424254...
your amount to pay back is 100.00 + fine of 0
Do you wan't to payback [y/n]: y
successfully done
```

```
Enter your choice: 11

ACCOUNT STATUS
Account number: AC25424254
Date of issue: 2021-07-08
Balance: 700.00
LOAN STATUS
NONE
```

12)Update details

```
                    Enter your choice: 10

      ``````````````````````````````````````````````````````````

                    [1] Update Name
                    [2] Update Phone number
                    [3] Update Address
                    [4] Update pan card
                    [5] Update email
                    [6] Update Password
 What to update (just type numbers one after other if many): 125
          New name: Mohammad Abdul Khalique
                                  done
          New Phone number: 1234567809
                                  done
          New email: xyz@gmail.com
                                  done
      ``````````````````````````````````````````````````````````
```

```
      Enter your choice: 1

``````````````````````````````````````````````````````

   Name: Mohammad Abdul Khalique
   Customer id: CI383
   Phone: 1234567809
   Pan card: AZXSQ1432C
   Email: xyz@gmail.com
   Address: 26, Laxmi Indl.estate, S.n.path, Lower Parel
   Account 1: AC25424254
   Account 2: None
``````````````````````````````````````````````````````
```

## 9. Limitations

Despite the best effort of the developer, the following limitations and functional boundaries are visible, which limits the scope of this application software.

1. There is no actual deposition of physical currency and any account can deposit any amount that they enter.
2. There could be a security breach in the database.

So far as future scope of the project is concerned, firstly it is open to any modular expansion i.e. other modules or functions can be designed and embedded to handle the user need in future. Any part of the software and reports can be modified independently without much effort.