

Visvesvaraya Technological University

Jnana Sangama, Belagavi – 590018, Karnataka



A Mini Project Report

on

“BANK DATA MANAGEMENT SYSTEM”

**Submitted in partial fulfillment of the requirement for the DBMS Laboratory
with mini project (18CSL58) of V semester**

Bachelor of Engineering

In

Computer Science and Engineering

Submitted By

KIRAN SWAMY S

(1GA18CS075)

MADHAN M

(1GA18CS079)

Under the Guidance

Mrs. Priyanka G

Assistant Professor,

Dept. of CSE



GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Raja Rajeshwari Nagar, Bengaluru – 560 098

2020-2021





GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Raja Rajeshwari Nagar, Bengaluru – 560 098



Certificate

This is to certify that V Semester Mini project entitled **“BANK DATA MANAGEMENT SYSTEM”** is a bonafide work carried out by **MADHAN M (1GA18CS079), KIRAN SWAMY S (1GA18CS075)** as a partial fulfillment for the award of Bachelor’s Degree in Computer Science and Engineering for DBMS Laboratory with Mini Project [18CSL58] as prescribed by **Visvesvaraya Technological University, Belagavi** during the year 2020-2021.

Mrs. Priyanka G
Assistant Professor,
Dept of CSE,
GAT, Bengaluru.

Dr. Srikanta Murthy K
Professor & Head,
Dept of CSE,
GAT, Bengaluru.

External Exam

Name of the Examiner

Signature with date

1. _____

2. _____

ABSTRACT

The Bank Data Management System is an application for maintaining a person's data in a bank. In this project we tried to show the working of a banking account system and cover the basic functionality of a Bank Data Management System. In this project our aim is give a look and feel of a banking system. In this user can open account, deposit, transfer money and user can also see their bank statement. A customer can Deposit money and view balance in his/her account, Transfer money to other accounts, get statement, etc.

This project maintains history of closed accounts i.e., which accounts have been closed on which dates and Bank admin or Manager can see this history.

Every customer has a customer-id and password to access his account. When a new customer registers himself with the bank, a customer-id is auto generated and given to him. Password needs to be set by the customer. It should be minimum 8 characters and can be any combination of numeric and alphabets.

A Bank admin or Manager has separate view where he can join employees, view statement of particular account, add interest based on account type. Overall aim is to feel of real-world banking system.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

We consider ourselves proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

We express our deep and sincere thanks to our Principal **Dr. N. Rana Pratap Reddy** for his support.

We are grateful to **Dr. Srikanta Murthy K**, Professor and Head, Dept. of Computer Science & Engineering who is source of inspiration and of invaluable help in channelizing my efforts in right direction.

We wish to thank our internal guide **Mrs. Priyanka G**, Assistant Professor, Dept of CSE for guiding and correcting various documents with attention and care. She has taken lot of pain to go through the document and make necessary corrections as and when needed.

We would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, we are grateful to our parents and friends for their unconditional support and help during our Project work.

MADHAN M (1GA18CS079)

KIRAN SWAMY S (1GA18CS075)

TABLE OF CONTENTS

1.0		INTRODUCTION	
	1.1	INTRODUCTION TO SQL	1
	1.2	INTRODUCTION TO FRONTEND SOFTWARE	2
	1.3	PROJECT REPORT OUTLINE	3
2.0		REQUIREMENT SPECIFICATION	
	2.1	SOFTWARE REQUIREMENTS	4
	2.2	HARDWARE REQUIREMENTS	4
3.0		OBJECTIVE OF THE PROJECT	5
4.0		IMPLEMENTATION	
	4.1	ER DIAGRAM	6
	4.2	MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM	7
	4.3	MAPPING OF THE ER SCHEMA TO RELATIONS	8
	4.4	CREATION OF TABLES	10
	4.5	CREATION OF TRIGGERS	15
5.0		FRONT END DESIGN	
	5.1	CONNECTIVITY TO DATABASE	16
	5.2	FRONT END and BACKEND CODE	17
6.0		TESTING	31

	6.1	TEST CASES FOR THE PROJECT	32
7.0		RESULTS	
	7.1	SNAPSHOTS	33
8.0		CONCLUSION	40

INTRODUCTION

1.1 INTRODUCTION TO SQL

SQL is a standard computer language for accessing and manipulating databases.

- SQL stands for **Structured Query Language**.
- SQL allows you to access a database.
- SQL is an ANSI standard computer language.
- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can insert new records in a database.
- SQL can delete records from a database.
- SQL can update records in a database.
- SQL is easy to learn.

SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc.

Unfortunately, there are many different versions of the SQL language, but to be in compliance with the ANSI standard; they must support the same major keywords in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others).

1.2 INTRODUCTION TO FRONTEND SOFTWARE

Python is a general-purpose, dynamic, object-oriented programming language. The design purpose of the Python language emphasizes programmer productivity and code readability. Python was initially developed by *Guido van Rossum*. It was first released in 1991. Python was inspired by ABC, Haskell, Java, Lisp, Icon, and Perl programming languages. Python is a high-level, multiplatform, interpreted language. Python is well suited for learning about GUI programming. GUI is a form of user interface which allows users to interact with computers through visual indicators using items such as icons, menus, windows, etc.

Tkinter is a Python binding to the Tk GUI toolkit. Tk is the original GUI library for the Tcl language. Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. There are several other popular Python GUI toolkits. Most popular are wxPython, PyQt, and PyGTK. Among all Tkinter is most widely used.

1.3 PROJECT REPORT OUTLINE

The report is arranged in the following way:

- 1: Gives the Introduction of SQL and Front-End Software
- 2: Gives the Requirement Specification of the project
- 3: Gives the Objective of the project
- 4: Gives the Implementation of the project
- 5: Gives the Front-end Design of the project
- 6: Gives the Testing information of the project
- 7: Gives the Results of the project

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Operating System: Windows XP / NT / 2000

Database: SQLITE3 database

Editor: Visual Studio code

2.2 HARDWARE REQUIREMENTS

Processor: Processor above 500MHz

RAM: 128 MB

Hard Disk: 6 GB

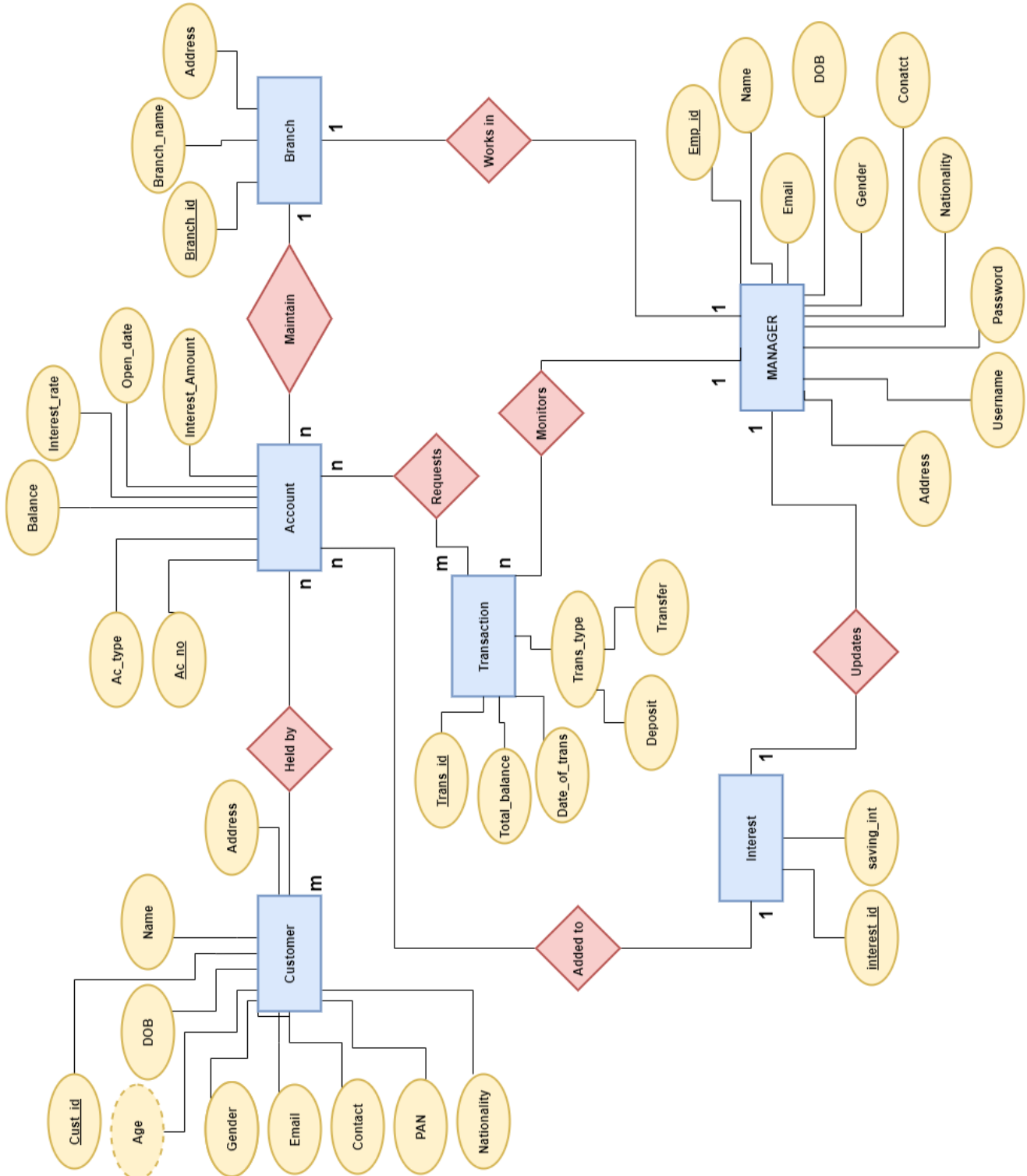
OBJECTIVE OF THE PROJECT

Bank is the place where customers feel the sense of safety for their property. In the bank, customers deposit and transfer their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers, indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank. Now a days, managing a bank is tedious job up to certain limit. So, software that reduces the work is essential. Also, today's world is a genuine computer world and is getting faster and faster day-by-day. Thus, considering above necessities, the software for bank management has become necessary which would be useful in managing the bank more efficiently.

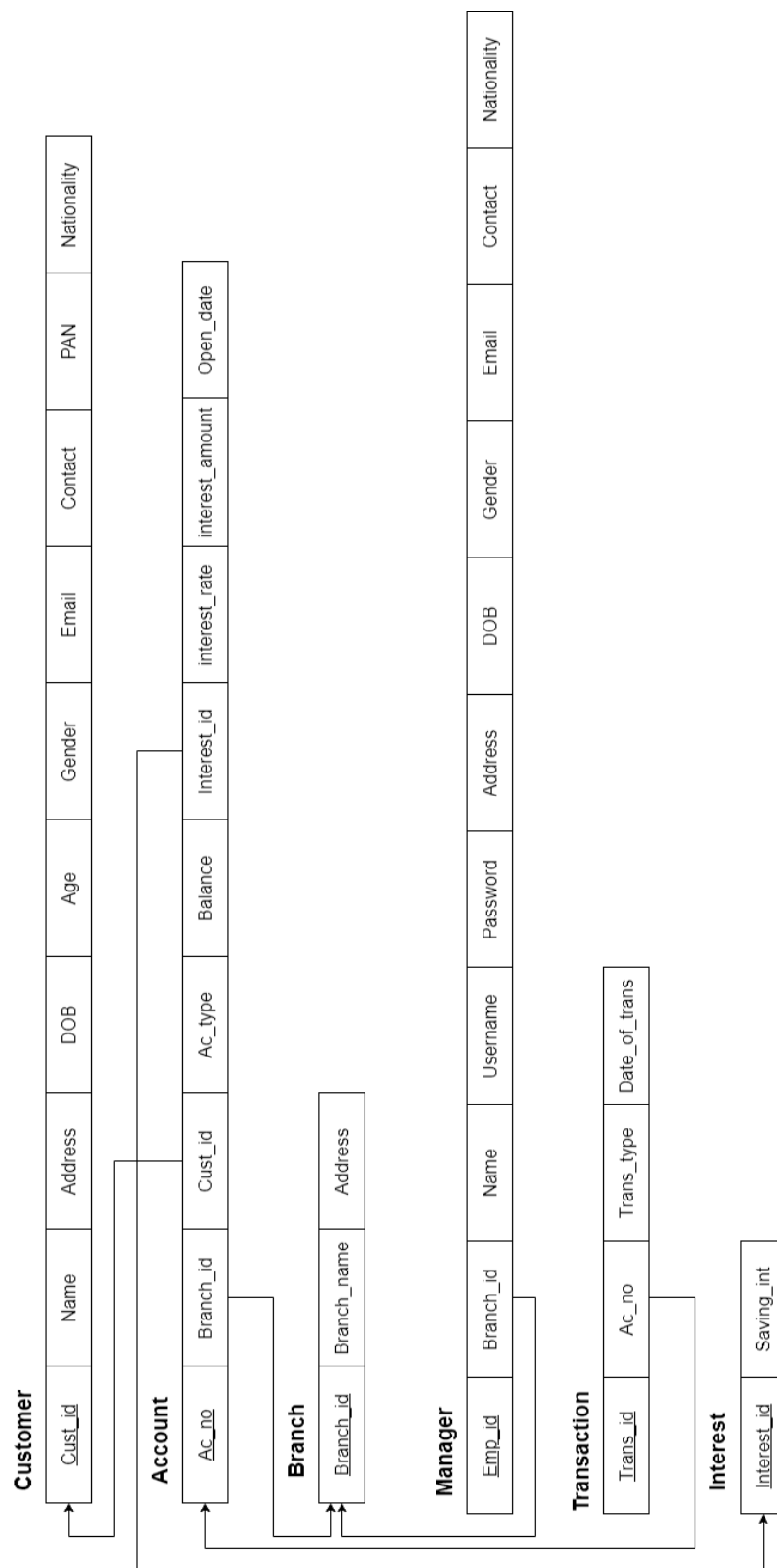
- Our software will perform and fulfill all the tasks that any customer would desire.
- Our motto is to develop a software program for managing the entire bank process related to customer accounts, and their various transaction processes efficiently.
- Hereby, our main objective is the customer's satisfaction considering today's faster world.

IMPLEMENTATION

4.1 ER DIAGRAM



4.2 MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM



Entities and their Attributes are :

- **Manager Entity:** Attributes of Manager Entity are Emp_id, Name and Address.
Emp_id is Primary Key for Manager Entity.
- **Customer Entity:** Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.
Customer_id is Primary Key for Customer Entity.
- **Branch Entity:** Attributes of Branch Entity are Branch_id, Name and Address.
Branch_id is Primary Key for Branch Entity.
- **Account Entity:** Attributes of Account Entity are Account_number, Account_Type and Balance.
Account_number is Primary Key for Account Entity.
- **Transaction Entity:** Attributes of Transaction Entity are Trans_id, Ac_no and Trans_type.
Trans_id is Primary Key for Transaction Entity.

In ER model, entities have attributes which can be of various types like single-valued, multi-valued, composite, simple, stored, derived and complex. But relationships can also have attributes associated to them. Generally, it is not recommended to give attributes to the relationships if not required because while converting the ER model into Relational model, things may get complex and we may require to create a separate table for representing the relationship.

Relationships are :

- Branch has Manager $\Rightarrow 1 : 1$

One Branch can have only one Manager and one Manager belongs to only one Branch, so the relationship between Branch and Manager is one to one relationship.

- Branch maintain Accounts $\Rightarrow 1 : N$

One Branch can have many Accounts but one Account cannot belong to many Branches, so the relationship between Branch and Account is one to many relationship.

- Account requests Transaction $\Rightarrow M : N$

One or more Account can requests many Transaction and many Transaction can belong to many account, so the relationship between Account and Transaction is many to many relationship.

- Account held by Customers $\Rightarrow M : N$

One Customer can have more than one Accounts and also One Account can be held by one or more Customers, so the relationship between Account and Customers is many to many relationship.

CREATION OF TABLES

CUSTOMER Table:

```
CREATE TABLE CUSTOMER(  
CUST_ID INT PRIMARY KEY,  
PASSWORD VARCHAR(20),  
NAME VARCHAR(50),  
DOB DATE,  
AGE INT,  
GENDER VARCHAR(6),  
EMAIL VARCHAR(50),  
CONTACT INT,  
PAN INT,  
NATIONALITY VARCHAR(15)  
);
```

CUSTOMER_Address Table:

```
CREATE TABLE CUSTOMER_Address(  
CUST_ID INT PRIMARY KEY,  
STREET VARCHAR(50),  
CITY VARCHAR(20),  
STATE VARCHAR(20),  
PIN INT,  
Foreign Key(CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON  
DELETE CASCADE  
);
```

INTEREST Table:

```
CREATE TABLE INTEREST(  
INTEREST_ID INT PRIMARY KEY,
```


SAVING_INT FLOAT

);

ACCOUNT Table:

CREATE TABLE ACCOUNT(

AC_NO INT PRIMARY KEY,

CUST_ID INT PRIMARY KEY,

Foreign Key(CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON
DELETE CASCADE,

AC_TYPE VARCHAR(10),

BALANCE INT,

Foreign Key(INTEREST_ID) REFERENCES
INTEREST(INTEREST_ID) ON DELETE CASCADE,

INTEREST_AMOUNT INT,

INTEREST_RATE INT,

OPEN_DATE DATE

);

BRANCH Table:

CREATE TABLE BRANCH(

BRANCH_ID INT PRIMARY KEY,

BRANCH_NAME VARCHAR(50)

);

BRANCH_ADDRESS Table:

CREATE TABLE BRANCH_ADDRESS(

BRANCH_ID INT PRIMARY KEY,

Foreign Key(BRANCH_ID) REFERENCES BRANCH(BRANCH_ID)
ON DELETE CASCADE,

STATE VARCHAR(20),

COUNTRY VARCHAR(20),

PIN INT);

OFFICER Table:

```
CREATE TABLE OFFICER(  
EMP_ID INT PRIMARY KEY,  
BRANCH_ID INT PRIMARY KEY,  
Foreign Key(BRANCH_ID) REFERENCES BRANCH(BRANCH_ID)  
ON DELETE CASCADE,  
NAME VARCHAR(50),  
EMAIL VARCHAR(50),  
GENDER VARCHAR(6),  
CONTACT INT,  
NATIONALITY VARCHAR(20),  
DOB DATE,  
USERNAME VARCHAR(20),  
PASSWORD VARCHAR(20)  
);
```

OFFICER_ADDRESS Table:

```
CREATE TABLE OFFICER_ADDRESS(  
EMP_ID INT PRIMARY KEY,  
Foreign Key(EMP_ID) REFERENCES OFFICER(EMP_ID) ON  
DELETE CASCADE,  
STREET VARCHAR(50),  
CITY VARCHAR(50),  
STATE VARCHAR(50),  
PIN INT  
);
```

MANAGER Table:

```
CREATE TABLE MANAGER(  

```

```
EMP_ID INT PRIMARY KEY,  
BRANCH_ID INT PRIMARY KEY,  
Foreign Key(BRANCH_ID) REFERENCES BRANCH(BRANCH_ID)  
ON DELETE CASCADE,  
NAME VARCHAR(50),  
EMAIL VARCHAR(50),  
GENDER VARCHAR(6),  
CONTACT INT,  
NATIONALITY VARCHAR(20),  
DOB DATE,  
USERNAME VARCHAR(20),  
PASSWORD VARCHAR(20)  
);
```

MANAGER_ADDRESS Table:

```
CREATE TABLE MANAGER_ADDRESS(  
EMP_ID INT PRIMARY KEY,  
Foreign Key(EMP_ID) REFERENCES OFFICER(EMP_ID) ON  
DELETE CASCADE,  
STREET VARCHAR(50),  
CITY VARCHAR(50),  
STATE VARCHAR(50),  
PIN INT  
);
```

TRANSACTION Table:

```
CREATE TABLE TRANSACTION(  
TRANS_ID INT PRIMARY KEY,  
AC_NO INT PRIMARY KEY,  
Foreign Key(AC_NO) REFERENCES ACCOUNT(AC_NO) ON  
DELETE CASCADE,  
TRANS_TYPE VARCHAR(20),
```

DATE_OF_TRANS DATE);

customerID_generator Table:

create table if not exists customerID_generator(row int primary key
,cust_id_g int)

account_NO_generator Table:

create table if not exists account_NO_generator(row int primary key
,acc_no_g int)

transactionID_generator Table:

create table if not exists transactionID_generator(row int primary key
,trans_id_g int)

employeeID_generator Table:

create table if not exists employeeID_generator(row int primary key
,employee_id_g int)

4.6 CREATION OF TRIGGERS

A trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. The trigger is made such that when a new record is inserted, it automatically changes the lower case to upper case.

TRIGGER backup_transaction_data_trigger :

```
CREATE TRIGGER backup_transaction_data_trigger BEFORE insert ON
TRANSACTIONS
    FOR EACH ROW
    BEGIN
        INSERT INTO BACKUP_TRANSACTIONS_DATA VALUES
        (
            NEW.AC_NO,
            NEW.TRANS_ID ,
            NEW.TRANS_TYPE,
            NEW.DATE_OF_TRANS
        );
    END;
```

TRIGGER backup_account_data :

```
CREATE TRIGGER backup_account_data BEFORE insert ON account
    FOR EACH ROW
    BEGIN
        INSERT INTO BACKUP_ACCOUNTS_DATA VALUES
        (
            NEW.AC_NO,NEW.INTEREST_ID,
            NEW.CUST_ID,
            NEW.AC_TYPE,NEW.BALANCE,
            NEW.INTEREST_AMOUNT,
            NEW.INTEREST_RATE,
            NEW.OPEN_DATE
        );
    END ;
```

5.0 FRONT END DESIGN

5.1 CONNECTIVITY TO DATABASE

SQLite3 can be integrated with Python using sqlite3 module. It provides an SQL interface and there is no need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use sqlite3 module, first create a connection object that represents the database and then create a cursor object, which is used for executing all the SQL statements.

There are the following steps to connect a python application to the database.

1. First, establish a connection to the SQLite database by creating a Connection object.
2. Next, create a Cursor object using the cursor method of the Connection object.
3. Then, execute a SELECT statement.
4. After that, call the fetchall() method of the cursor object to fetch the data.
5. Finally, loop the cursor and process each row individually.

CONNECTIVITY BASIC CODE USED:

Python file root.py:

```
import sqlite3 as base
# create or connect a data base
data_base = base.connect("demo1.db")
# create a cursor
cursor = data_base.cursor()
```

5.2 FRONT END and BACKEND CODE

File root_file.py:

```
from tkinter import *
import register_page
import os
import sqlite3 as base
import datetime

# create or connect a data base
data_base = base.connect("demo1.db")
# create a cursor
cursor = data_base.cursor()

login_page = Tk()
login_page.title("KM bank")
login_page.configure(bg="#93D5FF")
login_page.iconbitmap("dbmsicon.ico")
#login_page.geometry("900x700")
Label(login_page,text = "KMC BANK",font = "Algerian 35",bg = "#93D5FF").grid(sticky=E)

#####
# Only functions()
def login():
    u="admin"
    p="1234"
    user_i= username_box.get()
    passw_i=password_box.get()
    cursor.execute("SELECT PASSWORD from customer WHERE CUST_ID = :CUSTID;",{ 'CUSTID'
:user_i})
    password_real = cursor.fetchall()
    cust_id=user_i
    def manager_view():
        admin_page =Tk()
        admin_page.title("KM bank")
        admin_page.configure(bg="#93D5FF")
        admin_page.iconbitmap("dbmsicon.ico")

#functions
def employee_details_f():
    all_emp_details_page=Toplevel()
    all_emp_details_page.title("KM bank")
    all_emp_details_page.configure(bg="#93D5FF")
    all_emp_details_page.iconbitmap("dbmsicon.ico")
```

```

        cursor.execute("select * from OFFICER ")
        x =cursor.fetchall()
        i=0
        j=0
        var=''

        result=[]
        for i in range(len(x)):
            for j in range(6):
                var=var+"\t" +str(x[i][j])
                result.append(var)
                var=' '

        Label(all_emp_details_page,text ="EMP_ID  BRANCH_ID  NAME  EMAIL  GENDER
CONTACT ",font ="none 15",bg ="#93D5FF", borderwidth=2, relief="ridge",pady=10,padx=10
).grid(row=0,column=0,pady=10,padx=10,sticky = E)
        for i in range(len(result)):
            Label(all_emp_details_page,text = result[i] +"\t",font ="none 15",bg =
"#93D5FF", borderwidth=2, relief="ridge").grid(row=i+1,column=0,pady=10,padx=10,sticky
= E)

        def close_button_f():
            all_emp_details_page.destroy()
            close_button=Button(all_emp_details_page,text="Close",padx=20,pady=5,height
t = 2, width = 20,command=close_button_f)
            close_button.grid(row=30,column=0,pady=10,padx=20)
            all_emp_details_page.grab_set()

def account_details_f():
    all_ac_details_page=Toplevel()
    all_ac_details_page.title("KM bank")
    all_ac_details_page.configure(bg="#93D5FF")
    all_ac_details_page.iconbitmap("dbmsicon.ico")

    cursor.execute("select * from account ")
    x =cursor.fetchall()
    i=0
    j=0
    var=''
    result=[]
    for i in range(len(x)):
        for j in range(8):
            var=var+"\t" +str(x[i][j])
            result.append(var)
            var=' '

        Label(all_ac_details_page,text ="ACACCOUNT NO | INT_ID | CUSTOMER_ID |
AC_TYPE | BALANCE | INT_AMOUNT | INT_RATE | OPEN_DATE| ",font ="none 15",bg ="#93D

```



```

5FF", borderwidth=2, relief="ridge", pady=10, padx=10).grid(row=0, column=0, pady=10, padx=
10, sticky = E)
    for i in range(len(result)):
        Label(all_ac_details_page, text = result[i] + "\t", font = "none 15", bg = "
#93D5FF", borderwidth=2, relief="ridge").grid(row=i+1, column=0, pady=10, padx=10, sticky
= W)

    def close_button_f():
        all_ac_details_page.destroy()
        close_button=Button(all_ac_details_page, text="Close", padx=20, pady=5, height
= 2, width = 20, command=close_button_f)
        close_button.grid(row=30, column=0, pady=10, padx=20)

        all_ac_details_page.grab_set()

def account_activity_details_f():
    account_activity_details_tk = Toplevel()
    account_activity_details_tk.title("KM bank")
    account_activity_details_tk.configure(bg="#93D5FF")

    account_activity_details_tk.iconbitmap("dbmsicon.ico")
    Label(account_activity_details_tk, text = "Enter account no.", font = "none 2
5", bg = "#93D5FF").grid(row=0, column=0, pady=10, padx=20)
    acc_no_input_activity = Entry(account_activity_details_tk, font = "none 20", b
g = "#93D5FF")
    acc_no_input_activity.grid(row=1, column=0, pady=10, padx=20)
    def get_ac_no():
        if acc_no_input_activity.get() == '':
            Label(account_activity_details_tk, text="Enter Account No. or close
", font = "none 25", bg = "#93D5FF").grid(row=3, column=0, pady=10, padx=20)
        else:
            ac_no=int(acc_no_input_activity.get())
            get_acc_button.destroy()
            acc_no_input_activity.destroy()
            cursor.execute("SELECT * FROM transactions where ac_no =:ac_no",{
ac_no':ac_no})

            print(ac_no)
            x = cursor.fetchall()
            i=0
            j=0
            result =[]
            var=' '
            for i in range(len(x)):
                for j in range(4):
                    var=var+"\t" +str(x[i][j])
                result.append(var)
                var=' '
            i=0

```

```

        Label(account_activity_details_tk,text = "\tACCOUNT NO\tTRANS ID
\tACTIVITY \tACTIVITY DATETIME\t",font = "none 15",bg = "#93D5FF", borderwidth=2, relief
="ridge",pady=10,padx=10).grid(row=0,column=0,pady=10,padx=10,sticky = E)
        for i in range(len(result)):
            Label(account_activity_details_tk,text = result[i] + "\t",font
="none 15",bg = "#93D5FF", borderwidth=2, relief="ridge").grid(row=i+1,column=0,pady=10
,padx=10,sticky = E)
        def close_button_f():
            account_activity_details_tk.destroy()
            close_button=Button(account_activity_details_tk,text="Close",padx=
20,pady=5,height = 2, width = 20,command=close_button_f)
            close_button.grid(row=30,column=0,pady=10,padx=20)

get_acc_button=Button(account_activity_details_tk,text="submit",padx=20,pa
dy=5,height = 2, width = 20,command=get_ac_no)
get_acc_button.grid(row=2,column=0,pady=10,padx=20)
account_activity_details_tk.grab_set()

def add_employee_f():
    add_emp_page =Toplevel()
    add_emp_page.title("KM bank")
    add_emp_page.configure(bg="#93D5FF")

    add_emp_page.iconbitmap("dbmsicon.ico")

    #functions only
    def go_back_button():
        add_emp_page.destroy()
    def register_button():
        branch_id = branch_id_e.get()
        name = name_e.get()
        street = street_box.get()
        state = state_box.get()
        city = city_box.get()
        pin = pin_box.get()
        email = email_e.get()
        gender = gender_e.get()
        contact = contact_e.get()
        nationality= nationality_e.get()
        dob= dob_e.get()
        username = username_e.get()
        password = password_e.get()
        if branch_id == '':
            Label(add_emp_page,text = "Please enter Branch_id",font="Android 15
",padx=10,pady=10).grid(row=17,column=1)
        elif name == '':

```

```

Label(add_emp_page,text ="Please enter Name",font="Android 15",pad
x=10,pady=10)
        .grid(row=17,column=1)

        else:
            cursor.execute("INSERT INTO OFFICER VALUES (:EMP_ID ,:BRANCH_ID ,
:NAME , :EMAIL ,:GENDER ,:CONTACT ,:NATIONALITY ,:DOB ,:USERNAME ,:PASSWORD )",
                {
                    'EMP_ID':emp_id,
                    'BRANCH_ID': branch_id,
                    'NAME':name ,
                    'EMAIL':email ,
                    'GENDER':gender ,
                    'CONTACT': contact,
                    'NATIONALITY':nationality ,
                    'DOB': dob,
                    'USERNAME':username,
                    'PASSWORD':password
                })
            cursor.execute("INSERT INTO OFFICER_ADDRESS VALUES (:EMP_ID , :STR
EET ,:CITY,:STATE,:PIN)",
                {
                    'EMP_ID':emp_id,
                    'STREET':street,
                    'STATE':state ,
                    'CITY':city ,
                    'PIN':pin
                })
            add_emp_page.destroy()

            cursor.execute("SELECT employee_id_g FROM employeeID_generator where row =
1")
            x = cursor.fetchall()
            emp_id =x[0][0]
            cursor.execute("UPDATE employeeID_generator SET employee_id_g = :employee_
id_g WHERE row=1",{
                'employee_id_g':x[0][0]+1
            })

            #Only Buttons()
            register=Button(add_emp_page,text="Register Employee",padx=30,pady=5,

command=register_button)
            go_back=Button(add_emp_page,text="Go Back",padx=30,pady=5,command=go_back_
button)

            #button.grid()
            register.grid(row=15,column=1,pady=10,padx=20,sticky = W)
            go_back.grid(row=16,column=1,pady=10,padx=20,sticky = W)

```

```

def set_interest_rate_f():
    set_interest_rate = Toplevel()
    set_interest_rate.title("KM bank")
    set_interest_rate.configure(bg="#93D5FF")
    set_interest_rate.iconbitmap("dbmsicon.ico")

    cursor.execute("SELECT SAVING_INT FROM INTEREST where INTEREST_ID =1")
    x= cursor.fetchall()
    current_int_amt=x[0][0]
    Label(set_interest_rate,text ="Current Interest Rate for savings account"
,font ="none 25",bg ="#93D5FF").grid(row=0,column=0,pady=10,padx=20)
    Label(set_interest_rate,text =current_int_amt ,font ="none 25",bg ="#93D5FF").grid(row=1,column=0,pady=10,padx=20)
    Label(set_interest_rate,text ="Enter new Interest Rate" ,font ="none 25",bg ="#93D5FF").grid(row=2,column=0,pady=10,padx=20)
    new_interest_rate_e =Entry(set_interest_rate,font ="none 20",bg ="#93D5FF"
)

    new_interest_rate_e.grid(row=3,column=0,pady=10,padx=20)

def set_new_int_rate():
    new_interest_rate = new_interest_rate_e.get()
    if new_interest_rate == '':
        Label(set_interest_rate,text ="Enter any value or close",font ="none 15",bg ="#93D5FF").grid(row=31,column=0,pady=10,padx=20)
        return
    cursor.execute("UPDATE INTEREST SET SAVING_INT = :SAVING_INT WHERE INTEREST_ID =1;",{
        'SAVING_INT':new_interest_rate
    })
    cursor.execute("UPDATE ACCOUNT SET INTEREST_RATE= :INTEREST_RATE;",{
        'INTEREST_RATE':new_interest_rate
    })
    data_base.commit()
    submit_int_rate.destroy()
    Label(set_interest_rate,text ="Interest rate set to "+ str(new_interest_rate) ,font ="none 25",bg ="#93D5FF").grid(row=4,column=0,pady=10,padx=20)

    submit_int_rate=Button(set_interest_rate,text="submit",padx=20,pady=5,height = 2, width = 20,command=set_new_int_rate)
    submit_int_rate.grid(row=4,column=0,pady=10,padx=20)
def close_button_f():
    set_interest_rate.destroy()
    close_button=Button(set_interest_rate,text="Close",padx=20,pady=5,height = 2, width = 20,command=close_button_f)

```

```

close_button.grid(row=30,column=0,pady=10,padx=20)
set_interest_rate.grab_set()

def add_interest_to_accounts_f():
    add_int_acc_page = Tk()
    add_int_acc_page.title("KM bank")
    add_int_acc_page.configure(bg="#93D5FF")
    add_int_acc_page.iconbitmap("dbmsicon.ico")

def CONFIRM_f():
    cursor.execute("select AC_NO from ACCOUNT")
    x = cursor.fetchall()
    i=0
    for i in range(len(x)):
        ac_no=x[i][0]
        cursor.execute("select BALANCE from ACCOUNT where AC_NO =:AC_NO",
{'AC_NO':ac_no})

        balance_fetch= cursor.fetchall()
        balance = float(balance_fetch[0][0])
        cursor.execute("select INTEREST_RATE from ACCOUNT where AC_NO =:AC
_NO",{ 'AC_NO':ac_no})
        interest_rate_fetch= cursor.fetchall()
        int_rate = float(interest_rate_fetch[0][0])
        updated_balance = round((balance + balance*int_rate*0.001),3)
        updated_int_amt=round((balance*int_rate*0.001),3)
        cursor.execute("UPDATE account SET balance= :balance",{
'balance':updated_balance
        })
        cursor.execute("UPDATE account SET INTEREST_AMOUNT = :INTEREST_AMO

UNT ;",{

        'INTEREST_AMOUNT':updated_int_amt
        })
        cursor.execute("SELECT trans_id_g FROM transactionID_generator whe
re row =1")

        t = cursor.fetchall()
        trans_id =t[0][0]
        print("TransID = "+ str(trans_id))
        cursor.execute("UPDATE transactionID_generator SET trans_id_g = :t
rans_id_g WHERE row=1;",
        {
            'trans_id_g':t[0][0]+1
        })
        date_of_trans = datetime.datetime.now()

```

```

        cursor.execute("INSERT INTO TRANSACTIONS VALUES (:AC_NO,:TRANS_ID
, :TRANS_TYPE , :DATE_OF_TRANS )",
        {
            'AC_NO':ac_no,
            'TRANS_ID':trans_id,
            'TRANS_TYPE': "\tSavingsINT "+" str(updated_balance)+"\t",
            'DATE_OF_TRANS':date_of_trans
        })
        confirm_button.destroy()
        Label(add_int_acc_page,text ="Interest amount added to all accounts" ,
font ="none 25",bg ="#93D5FF").grid(row=1,column=0,pady=10,padx=20)
        Label(add_int_acc_page,text ="Confirm to add interest amount to all accoun
ts" ,font ="none 25",bg ="#93D5FF").grid(row=0,column=0,pady=10,padx=20)
        confirm_button=Button(add_int_acc_page,text="CONFIRM",padx=20,pady=5,height
= 2, width = 20,command=CONFIRM_f)
        confirm_button.grid(row=1,column=0,pady=10,padx=20)

def log_out_f():
    admin_page.destroy()
    #Labels
    Label(admin_page,text = "Welcome Manager",font ="Algerian 25",bg ="#93D5FF").g
rid(row=0,column=0,pady=10,padx=20)
    #Button
    employee_details=Button(admin_page,text="Employee Details",padx=30,pady=5,height
= 2, width = 20,command=employee_details_f)
    account_details=Button(admin_page,text="All Account Details",padx=30,pady=5,height
= 2, width = 20,command=account_details_f)
    account_activity_details=Button(admin_page,text="Account activity Details",pad
x=30,pady=5,height = 2, width = 20,command=account_activity_details_f)
    add_employee=Button(admin_page,text="Add Employee",padx=30,pady=5,height = 2,
width = 20,command=add_employee_f)
    set_interest_rate=Button(admin_page,text="Set Interest Rate",padx=30,pady=5,height
= 2, width = 20,command=set_interest_rate_f)
    add_interest_to_accounts=Button(admin_page,text="Add interest amount to accoun
ts",padx=30,pady=5,height = 2, width = 20,command=add_interest_to_accounts_f)
    log_out_button=Button(admin_page,text="LogOut",padx=30,pady=5,height = 2, width
= 20,command=log_out_f)
    log_out_button.grid(row=4,column=0,pady=10,padx=20)
    admin_page.grab_set()
def customer_signin_window(cust_id_):
    signin_page =Tk()
    signin_page.title("KM bank")
    signin_page.configure(bg="#93D5FF")
    signin_page.iconbitmap("dbmsicon.ico")

def check_balance_f():

```

```

        balance_page =Toplevel()
        balance_page.title("KM bank")
        balance_page.configure(bg="#93D5FF")
        balance_page.iconbitmap("dbmsicon.ico")
        cursor.execute("SELECT BALANCE from account WHERE AC_NO = :ACNO;",{ 'ACNO'
:ac_no})

        b = cursor.fetchall()
        balance =b[0][0]
        balance_l=Label(balance_page,text = "  Your Current Balance: "+ str(balanc
e)+"  ",font ="none 25",bg ="#93D5FF")
        balance_l.grid(row=0,column=0,pady=10,padx=20)
        def close_button_f():
            balance_page.destroy()
            close_button=Button(balance_page,text="Close",padx=30,pady=10,height = 2,
width = 20,command=close_button_f)
            close_button.grid(row=1,column=0,pady=10,padx=20)
            balance_page.grab_set()

        def deposit_f():
            deposit_page =Toplevel()
            deposit_page.title("KM bank")
            deposit_page.configure(bg="#93D5FF")
            deposit_page.iconbitmap("dbmsicon.ico")
            deposit_test_l=Label(deposit_page,text = "  Enter amount to d
eposit: ",font ="none 25",bg ="#93D5FF")
            deposit_amount_entry=Entry(deposit_page,font ="none 25")
            deposit_test_l.grid(row=0,column=0,pady=10,padx=20)
            deposit_amount_entry.grid(row=1,column=0,pady=10,padx=20)
            def submit_button_f():
                cursor.execute("SELECT BALANCE from account WHERE AC_NO = :ACNO;",{ 'A
CNO':ac_no})
                b = cursor.fetchall()
                balance =b[0][0]
                if deposit_amount_entry.get() == '':
                    Label(deposit_page,text = "  Enter amount to deposit or Close ",fo
nt ="none 10",bg ="#93D5FF").grid(row=1,column=1,pady=10,padx=20)
                elif deposit_amount_entry.get().isdigit():
                    cursor.execute("SELECT trans_id_g FROM transactionID_generator whe
re row =1")
                    t = cursor.fetchall()
                    trans_id =t[0][0]
                    print("TransID = "+ str(trans_id))
                    cursor.execute("UPDATE transactionID_generator SET trans_id_g = :t
rans_id_g WHERE row=1;",
                    {
                        'trans_id_g':t[0][0]+1
                    })
                    date_of_trans = datetime.datetime.now()

```



```

        b = cursor.fetchall()
        balance = b[0][0]
        if withdraw_amount_entry.get() == '':
            Label(withdraw_page, text = " Enter amount to withdraw or Close ",
font = "none 10", bg = "#93D5FF").grid(row=1, column=1, pady=10, padx=20)
        elif withdraw_amount_entry.get().isdigit():
            cursor.execute("SELECT trans_id_g FROM transactionID_generator where row =1")

            t = cursor.fetchall()
            trans_id = t[0][0]
            print("TransID = "+ str(trans_id))
            cursor.execute("UPDATE transactionID_generator SET trans_id_g = :trans_id_g WHERE row=1;",
            {
                'trans_id_g': t[0][0]+1
            })
            date_of_trans = datetime.datetime.now()

            cursor.execute("INSERT INTO TRANSACTIONS VALUES (:AC_NO, :TRANS_ID, :TRANS_TYPE, :DATE_OF_TRANS )",
            {
                'AC_NO': ac_no,
                'TRANS_ID': trans_id,
                'TRANS_TYPE': "\tWITHDREW - "
            + str(withdraw_amount_entry.get())+"\t",
                'DATE_OF_TRANS': date_of_trans
            })

            updated_balance = balance - int(withdraw_amount_entry.get())
            cursor.execute("UPDATE account SET balance = :balance WHERE AC_NO = :ACNO;", { 'balance': updated_balance, 'ACNO': ac_no })
            submit_button.destroy()
            Label(withdraw_page, text = "Rs "+str(withdraw_amount_entry.get())+" withdrawn", font = "none 25", bg = "#93D5FF").grid(row=2, column=0, pady=10, padx=20)
        else:
            Label(withdraw_page, text = " Enter only numbers ", font = "none 10", bg = "#93D5FF").grid(row=1, column=1, pady=10, padx=20)

            submit_button = Button(withdraw_page, text = "withdraw", padx=30, pady=10, height = 2, width = 20, command = submit_button_f_w)
            submit_button.grid(row=2, column=0, pady=10, padx=20)
            def close_button_f():
                withdraw_page.destroy()
            close_button = Button(withdraw_page, text = "Close", padx=30, pady=10, height = 2, width = 20, command = close_button_f)

```

```

        close_button.grid(row=3,column=0,pady=10,padx=20)
        withdraw_page.grab_set()

def mini_statement_f():
    mini_statement =Toplevel()
    mini_statement.title("KM bank")
    mini_statement.configure(bg="#93D5FF")
    mini_statement.iconbitmap("dbmsicon.ico")
    cursor.execute("SELECT * FROM transactions where ac_no =:ac_no",{ 'ac_no':a
c_no})

    print(ac_no)

    x = cursor.fetchall()
    i=0
    j=0
    result =[]
    var=' '
    for i in range(len(x)):
        for j in range(4):
            var=var+"\t" +str(x[i][j])
            result.append(var)
            var=' '
        i=0
        Label(mini_statement,text = "\tACCOUNT NO\t\tTRANSACTION ID \tACTIVITY \t
\tACTIVITY DATETIME\t",font = "none 15",bg = "#93D5FF", borderwidth=2, relief="ridge",pa
dy=10,padx=10).grid(row=0,column=0,pady=10,padx=0,sticky = E)
        for i in range(len(result)):
            Label(mini_statement,text = result[i] +"\t",font = "none 15",bg = "#93D5
FF", borderwidth=2, relief="ridge").grid(row=i+1,column=0,pady=10,padx=10,sticky = E)
        def close_button_f():
            mini_statement.destroy()
            close_button=Button(mini_statement,text="Close",padx=20,pady=5,height = 2,
width = 20,command=close_button_f)
            close_button.grid(row=30,column=0,pady=10,padx=20)
            mini_statement.grab_set()

def log_out_f():
    signin_page.destroy()
    cursor.execute("SELECT AC_NO from account WHERE CUST_ID = :CUSTID",{ 'CUSTID'
:cust_id_})

    x = cursor.fetchall()
    ac_no=x[0][0]
    #print(ac_no)
    cursor.execute("SELECT NAME from customer WHERE CUST_ID = :CUSTID",{ 'CUSTID'
:cust_id_})
    n = cursor.fetchall()
    name = n[0][0]

```

```

        #print(balance)
        #Label
        welcome_message=Label(signin_page,text = "Welcome "+name,font ="Algerian 25",bg
g ="#93D5FF")
        #Button
        check_balance=Button(signin_page,text="Check Balance",padx=30,pady=5,height =
2, width = 20,command=check_balance_f)
        deposit=Button(signin_page,text="Deposit",padx=30,pady=5,height = 2, width = 2
0,command=deposit_f)
        withdraw=Button(signin_page,text="Withdraw",padx=30,pady=5,height = 2, width =
20,command=withdraw_f)
        mini_statement=Button(signin_page,text="Mini Statement",padx=30,pady=5,height
= 2, width = 20,command=mini_statement_f)
        log_out_button=Button(signin_page,text="LogOut",padx=30,pady=5,height = 2, wid
th = 20,command=log_out_f)

        #Label.grid()
        welcome_message .grid(row=0,column=0,pady=10,padx=20)
        #Button.grid()
        check_balance .grid(row=1,column=0,pady=10,padx=20)
        deposit .grid(row=2,column=0,pady=10,padx=20)
        withdraw .grid(row=1,column=1,pady=10,padx=20)
        mini_statement .grid(row=2,column=1,pady=10,padx=20)
        log_out_button .grid(row=3,column=0,pady=10,padx=20)
        signin_page.grab_set()
    if user_i == '':
        lab = Label(login_page,text = "Please enter Cust_ID",font="Android 15",padx=10,
pady=10)
        lab.grid(row=5,column=1)
    elif passw_i == '':
        lab = Label(login_page,text = "Please enter Password",font="Android 15",padx=10
,pady=10)
        lab.grid(row=5,column=1,columnspan=2)
    elif user_i == u and passw_i == p:
        manager_view()
    elif password_real[0][0] == passw_i:
        cursor.execute("SELECT NAME from customer WHERE CUST_ID = :CUSTID;",{ 'CUSTID'
:user_i})
        n = cursor.fetchall()
        print(password_real[0][0])
        customer_signin_window(cust_id_)
        login_page.grab_set()
    else:
        lab = Label(login_page,text = "Try Again",font="Android 20",padx=10,pady=10)

        lab.grid(row=5,column=1,columnspan=2)

```

```

register_page_exists=0
def signup():
    global register_page_exists
    global register_page
    if register_page_exists == 0:
        os.system('register_page.py')

    if register_page.register_page.winfo_exists():
        register_page_exists=1
    else:
        os.system('register_page.py')
        register_page_exists=1

#####
#Only Label()
#picture = PhotoImage(file = "")
#lab = Label(login_page, image=picture)
#lab.grid(row=0,column=0,columnspan=2)
username=Label(login_page,text = "CustomerID",font = "none 15",bg = "#93D5FF")
password=Label(login_page,text="Password",font = "none 15",bg = "#93D5FF")
#####

#Only Entry()
username_box=Entry(login_page,font="none 15", w=20)
password_box=Entry(login_page,font="none 15", w=20,show="*")
#####

#Only Buttons()
login=Button(login_page,text="Login",padx=20,pady=3,command=login)
sign_up=Button(login_page,text="SignUp",padx=20,pady=3,command=signup)
#####
#GRID()
#####
#Only Label.grid()
username.grid(row=1,column=0,pady=10,padx=20)
password.grid(row=2,column=0,pady=5,padx=2)
#####
#Only Entry.grid()
username_box.grid(row=1,column=1,pady=10,padx=20)
password_box.grid(row=2,column=1,pady=10,padx=20)
#####
# Only Button.grid()
login.grid(row=3,column=1,pady=8,padx=20,sticky = W)
sign_up.grid(row=4,column=1,pady=8,padx=20,sticky = W)
mainloop()

data_base.commit()
data_base.close()

```

6.0 Testing

Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. The various types of test. Each test type addresses a specific testing requirement.

TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

TESTING OBJECTIVES

The main objectives of testing process are as follows.

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding undiscovered error.
- A successful test is one that uncovers the undiscovered error.

6.1 TEST CASES

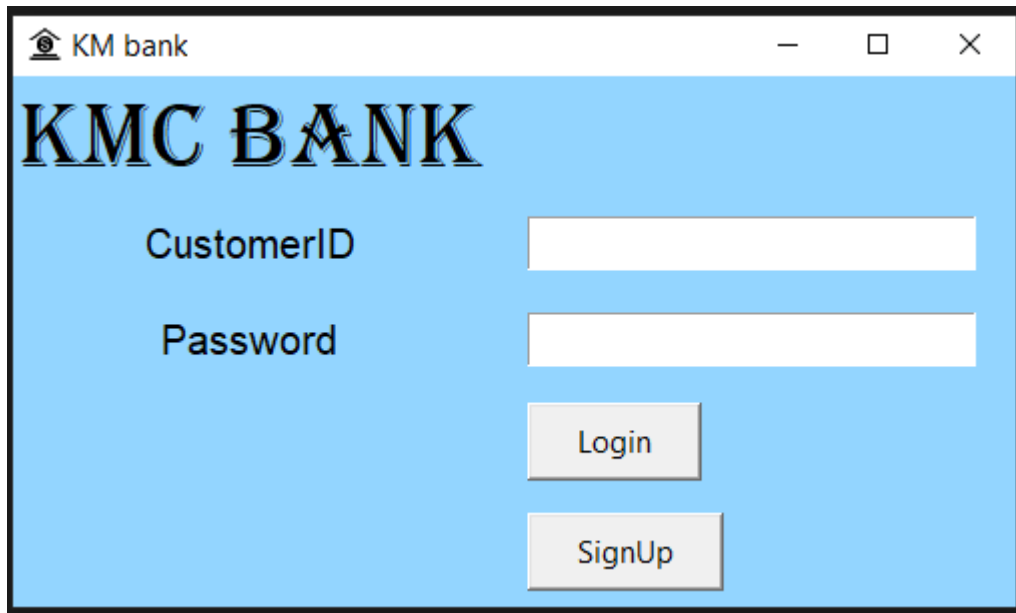
Sl no.	Test Input	Expected Results	Observed Results	Remarks
1	Insert a Record	New tuple should be inserted	Query Ok 1 row affected or inserted	PASS
2	Insert a Record	New tuple should be inserted	Query Ok 1 row affected or inserted	PASS
3	Search a Record	Display the Record	Required Record Found	TRUE
4	Search a Record	Display the Record	Required Record Found	TRUE
5	Delete a record	Delete the record	Query Ok 1 row affected or record Deleted	PASS
6	Update a record	Tuple should be updated	Query Ok 1 row updated	PASS
7	Update a record	Tuple should be updated	Query Ok 1 row updated	PASS

7.0 RESULTS

This section describes the screens of the “Project title”. The snapshots are shown below for each module.

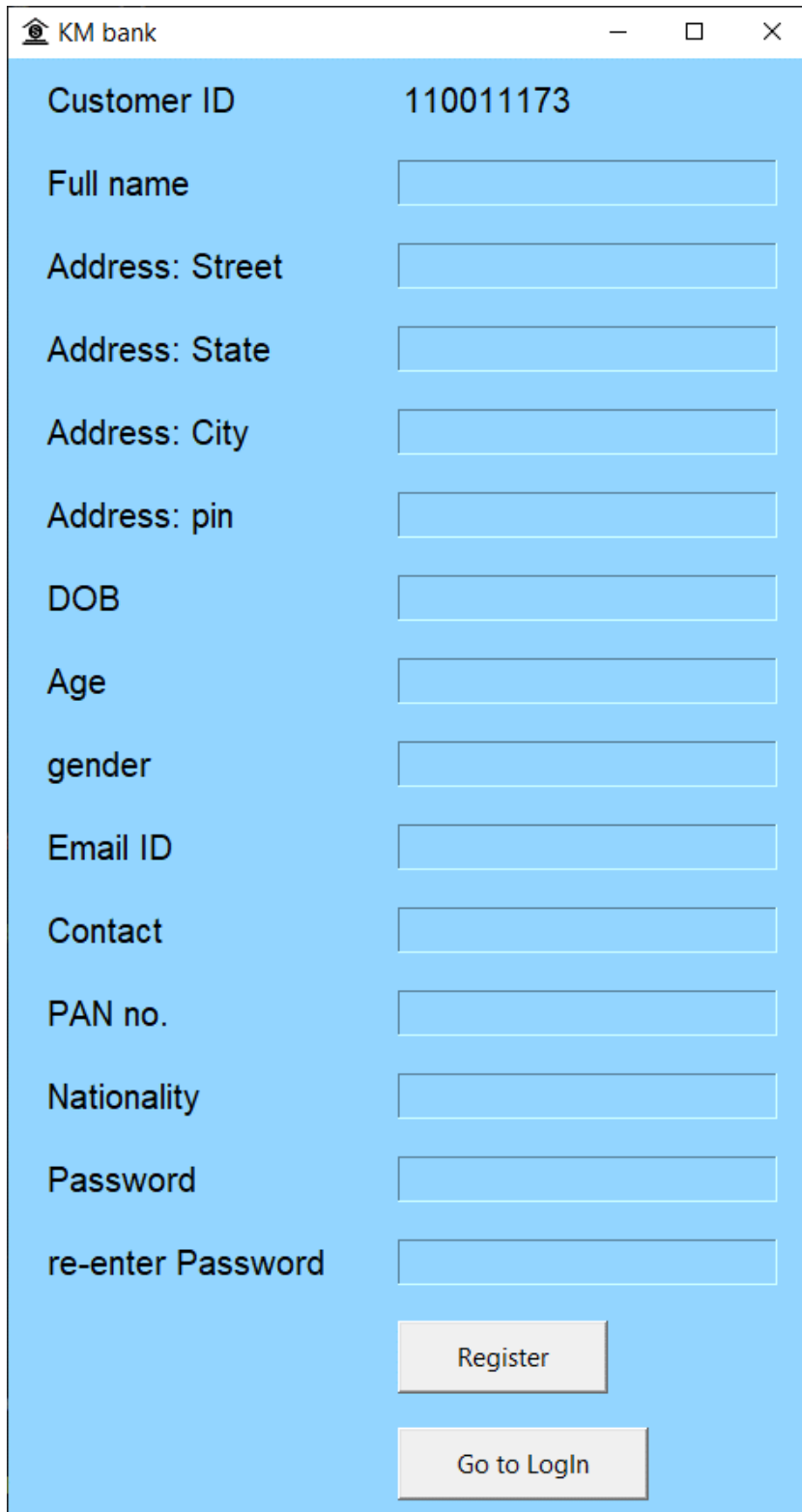
7.1 SNAPSHOTS

Login window for customer and ADMIN:



The screenshot shows a web application window titled "KM bank" with standard window controls (minimize, maximize, close). The main content area has a light blue background. At the top, the text "KMC BANK" is displayed in a large, bold, serif font. Below this, there are two input fields: "CustomerID" and "Password". To the right of each label is a white rectangular input box. Below the "Password" field, there are two buttons: "Login" and "SignUp", both with a light gray background and a thin black border.

Register Customer window:



A screenshot of a web browser window titled "KM bank". The window contains a registration form with the following fields and labels:

Customer ID	110011173
Full name	<input type="text"/>
Address: Street	<input type="text"/>
Address: State	<input type="text"/>
Address: City	<input type="text"/>
Address: pin	<input type="text"/>
DOB	<input type="text"/>
Age	<input type="text"/>
gender	<input type="text"/>
Email ID	<input type="text"/>
Contact	<input type="text"/>
PAN no.	<input type="text"/>
Nationality	<input type="text"/>
Password	<input type="text"/>
re-enter Password	<input type="text"/>

At the bottom of the form, there are two buttons:

- Register
- Go to Login

Window after registration:

The image displays two overlapping web browser windows from 'KM bank'.

Left Window (Registration Form):

Customer ID	110011174
Full name	Ambani
Address: Street	ks layout
Address: State	Karnataka
Address: City	Banaglore
Address: pin	560111
DOB	2000-02-02
Age	45
gender	Male
Email ID	ambani@ggg.com
Contact	9999999999
PAN no.	77777777
Nationality	Indian
Password	*****
re-enter Password	*****

Buttons: Register, Go to Login

Right Window (Account Confirmation):

Account No.	66556655075
<input type="checkbox"/> Saving	
<input type="checkbox"/> Current	
Submit	

ADMIN VIEW:

The image displays two overlapping windows from a web application titled "KM bank".

The top window is the login screen, featuring the "KMC BANK" logo at the top. It contains two input fields: "CustomerID" with the value "admin" and "Password" with the value "****". Below these fields are two buttons: "Login" and "SignUp".

The bottom window is the "WELCOME MANAGER" dashboard. It features a grid of buttons for administrative tasks:

- Employee Details
- Add Employee
- All Account Details
- Account activity Details
- Set Interest Rate
- Add interest amount to accounts
- LogOut

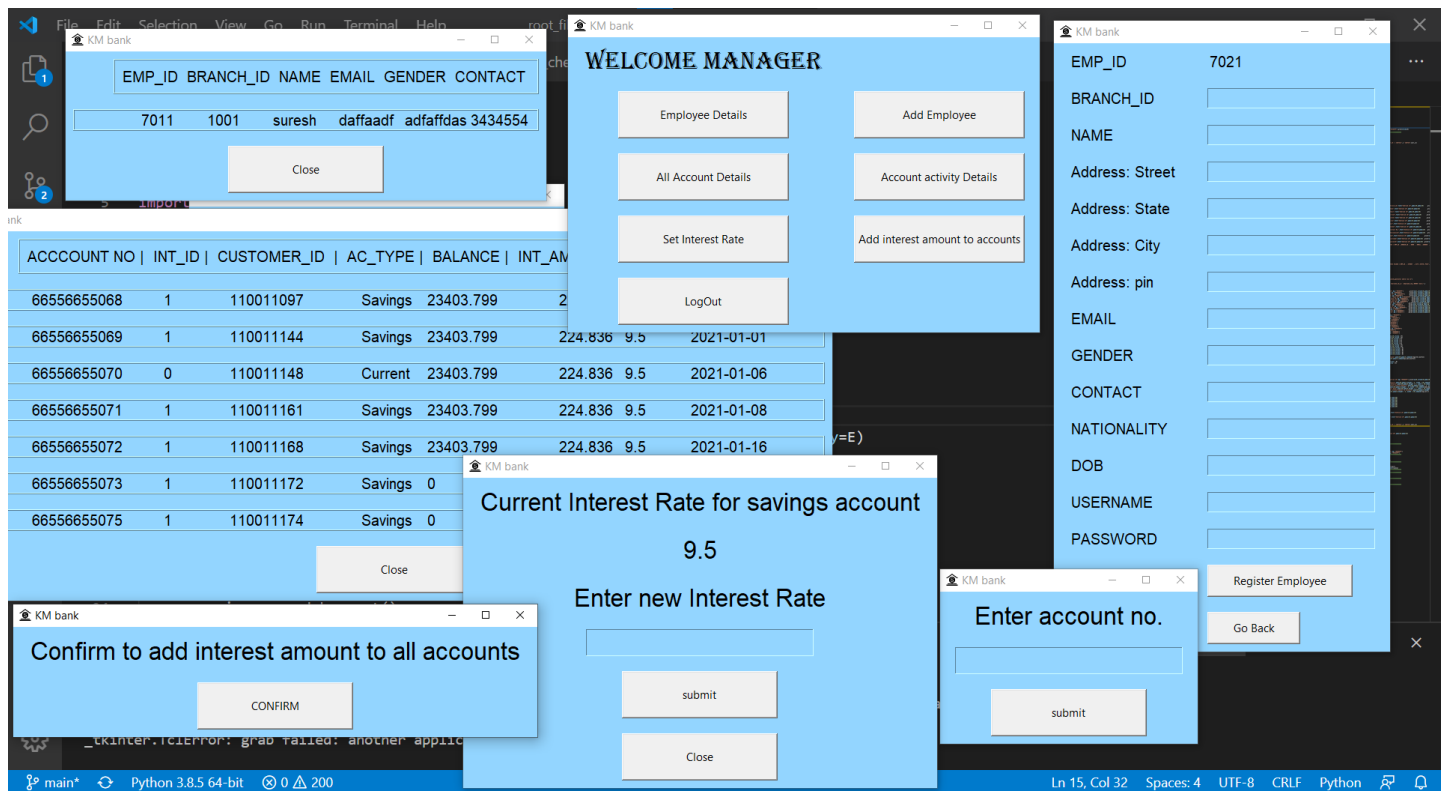
Customer/Accountholder View:

The image displays two overlapping windows from a web application titled "KM bank".

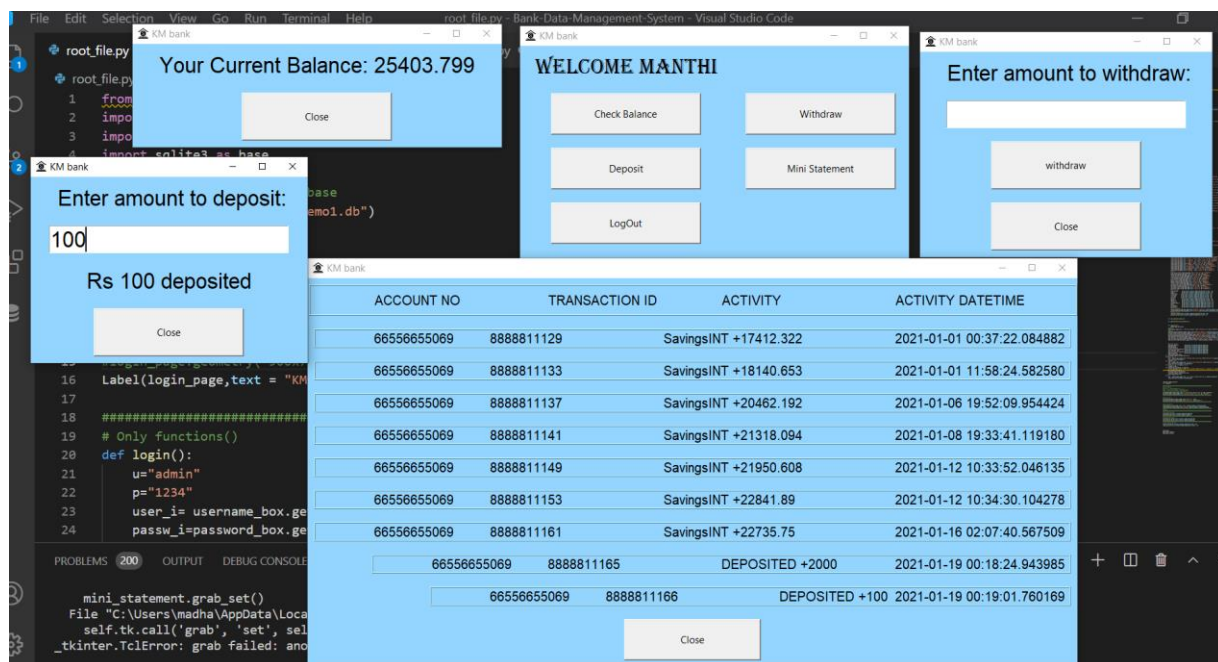
The top window shows the login interface with the "KMC BANK" logo. It contains two input fields: "CustomerID" with the value "110011097" and "Password" with masked characters "*****". Below these fields are two buttons: "Login" and "SignUp".

The bottom window shows the home interface after a successful login, displaying "WELCOME MADHAN M". It features five buttons arranged in two columns: "Check Balance", "Withdraw", "Deposit", "Mini Statement", and "LogOut".

All windows for ADMIN:



All windows for an Accountholder:



Account statement/Activity details of an Example account after few transactions:

The screenshot displays a Python application window titled "KM bank" with a light blue background. The main window shows a "WELCOME MADDD" message and five buttons: "Check Balance", "Withdraw", "Deposit", "Mini Statement", and "LogOut". Below this, a second window titled "KM bank" displays a table of account activity details. The table has four columns: "ACCOUNT NO", "TRANSACTION ID", "ACTIVITY", and "ACTIVITY DATETIME". It lists three transactions for account number 66556655073. A "Close" button is located at the bottom of the table.

ACCOUNT NO	TRANSACTION ID	ACTIVITY	ACTIVITY DATETIME
66556655073	8888811165	DEPOSITED +5000	2021-01-19 00:29:55.120489
66556655073	8888811166	WITHDREW -1000	2021-01-19 00:29:58.605871
66556655073	8888811172	SavingsINT +24769.903	2021-01-19 00:30:15.044111

Close

CONCLUSION

The current Bank data management system does insertion and updating data accurately and also performs all the functionalities better compared to a general ATM machine in terms of more functionalities.

If implemented in real world by integrating it with real-world would save customer's valuable time and can be an all-in-on software for both customer, employees and bank managers.

To improve the Bank data management system even more useful by adding Loan functionalities and automate the eligibility process for a loan and many other processes involved in getting a loan.