Bonnie Foi Co - Ed. Sr. Sec. School, Bhopal, M.P.

(Affiliated to C.B.S.E.)

(Session 2021 - 2022)

A

Project file

On

Banking Management and System

Submitted to:

Mr. Vivek Kumar Sharma

PGT (Computer Science)

Vikram Sarkar

Class XII

INDEX:

PREFACE

This project has been prepared keeping in view the requirements of central board of secondary education, New Delhi.

The project deals on the topic of Banking management system through Python and MySQL. Tkinter has been used to make the project more interactive, simpler and as easier as possible.

The project is aimed at providing a thorough base and understanding on Python and MySQL. It also helped in understanding the various latest trends and techniques in MySQL and Python.

Vikram Sarkar
Bonnie Foi Co. Ed. Sr. Sec. School

INTRODUCTION

During the past several decades personnel function has been transformed from a relatively obscure record keeping staff to central and top-level management function. There are many factors that have influenced this transformation like technological advances, professionalism, and general recognition of human beings as most important resources.

A computer-based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation. This project intends to introduce more user friendliness in the various activities such as record updating, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updating can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The entire information has maintained in the database or Files and whoever wants to retrieve can't retrieve, only authorization user can retrieve the necessary information which can be easily be accessible from the file.

OBJECTIVE OF THE PROJECT

A computer based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation.
This project intends to introduce more user friendliness in the various activities such as record updating, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updating can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.
The main objective of our project is providing the different typed of customers facility, the main objective of this system is to find out the actual customer service. Etc.

- It should fulfill almost all the process requirements of any Bank.
- It should increase the productivity of bank by utilizing the working hours more and more, with minimum manpower.

This project includes the entire upgraded feature required for the computerization banking system. This system is very easy to use, so that any user can use without getting pre-knowledge about this. Its very much user friendly and meet almost all daily working process requirements. This system is completely GUI based and can be use by mouse and as well as keyboard. This system is melded in such a way that has got all features to upgrade without making much change in existing components.

SYSTEM ANALYSIS

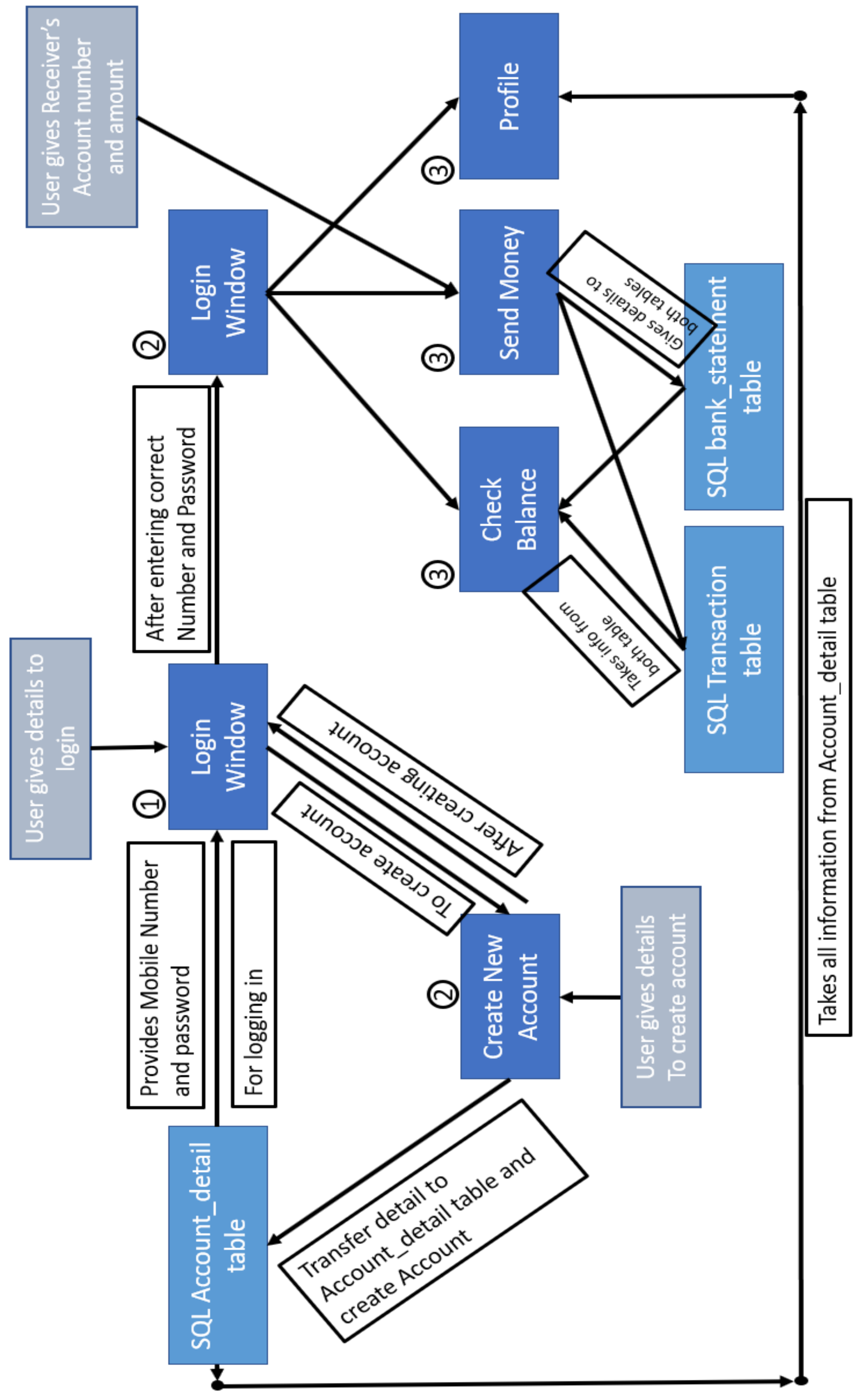
- **Current System:**
  - Requires a 64-bit processor and operating system
  - OS: Windows 11
  - Processor: Intel Core i3-10100
  - Memory: 8 GB DDR4 RAM
  - Graphics: NVIDIA GeForce GT 710
  - DirectX: Version 11

- **Minimum System Requirement:**
  - Requires a 64-bit processor and operating system
  - OS: Windows 7
  - Processor: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD
  - Memory: 2 GB DDR3 RAM
  - Graphics: Nvidia or ATI with support of OpenGL 1.5 or higher
  - DirectX: Version 11

- **Requirement Analysis:**
  - Requires Python version 2.0.1 or above:
    Modules of python required:
    1) tkinter
    2) Pillow
    3) mysql-connector-python
  - Requires mysql 7.0 or above

Project Design

Data Flow Diagram:


User gives details to login
Login Window
①
Provides Mobile Number and password
For logging in
SQL Account_detail table
After entering correct Number and Password
Login Window
②
User gives Receiver's Account number and amount
To create account
After creating account
Create New Account
②
User gives details To create account
Transfer detail to Account_detail table and create Account
Check Balance
③
Send Money
③
Profile
③
Gives details to both tables
Takes info from both table
SQL bank_statement table
SQL Transaction table
Takes all information from Account_detail table

Front-end and Back-end

About Python

Python is a high level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

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

This makes python one of the most popular programming language.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and Modula 3. It was first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

- **Advantages of python are as follows:**
  1. Easy to use
  2. Expressive language
  3. Interpreted language
  4. Free and open source
- **Disadvantages of python are as follows:**
  1. Not the fastest language
  2. Lesser library than C, Java, Perl

3. Not easily convertible
4. Not strong on type-binding

About IDE used (Visual Studio)

Microsoft visual studio IDE (integrated development environment) is being used in this project of Banking management system.

It is an IDE developed by Microsoft. It is used to develop computer programs as well as websites, web applications etc. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.

Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, Typescript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins. Java were supported in the past.

The most basic edition of Visual Studio, the Community edition, is available free of charge. The slogan for Visual Studio Community edition is "Free, fully-featured IDE for students, open-source and individual developers".

- ### Features of Visual Studio:
  1. Code Editor: Visual Studio (like any other IDE includes a code editor that supports syntax highlighting and code completion using IntelliSense for variables, functions, methods, loops, and LINQ, queries
  2. Debugger: Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio.
  3. Designer: Visual Studio includes a host of visual designers to aid in the development of applications.

Visual studio has a lot more features.

Python Modules used

- ### Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

1. Import the *Tkinter* module.
2. Create the GUI application main window.
3. Add one or more of the above-mentioned widgets to the GUI application.
    - **a.** Enter the main event loop to take action against each event triggered by the user.

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter which are as follows:

Button, Canvas, Radiobutton, Checkbutton, Entry, Frame, Label, Listbox,

Menubutton, Menu, Message, Scale, Scrollbar, Text, Toplevel, Spinbox.

- ### Pillow:

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

Capability of Pillow Module:

Pillow offers several standard procedures for image manipulation. These include:

1. per-pixel manipulations,

2. masking and transparency handling,
3. image filtering, such as blurring, contouring, smoothing, or edge finding,
4. image enhancing, such as sharpening, adjusting brightness, contrast or color,
5. adding text to images and much more.

- **MySQL-connector-python**

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with Python Database.

MySQL Connector/Python includes support for:

- Almost all features provided by MySQL Server up to and including MySQL Server version 8.0.
- Converting parameter values back and forth between Python and MySQL data types, for example Python datetime and MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.
- All MySQL extensions to standard SQL syntax.
- Protocol compression, which enables compressing the data stream between the client and server.
- Connections using TCP/IP sockets and on Unix using Unix sockets.
- Secure TCP/IP connections using SSL.
- Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library.

About SQL

SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc. SQL is an **ANSI** (American National Standards Institute) standard language, but there are many different versions of the SQL language.

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as −

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

About MySQL

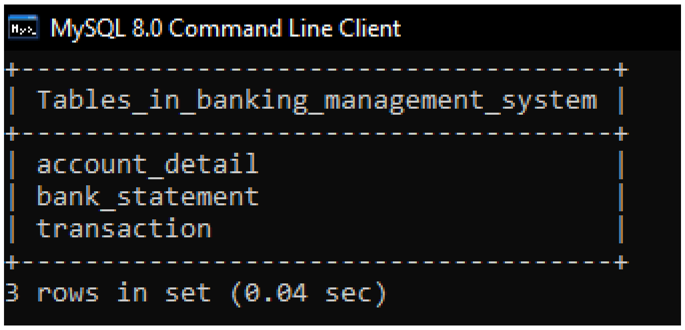
My SQL is an open-source relational database management system (RDMS).

Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation. In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

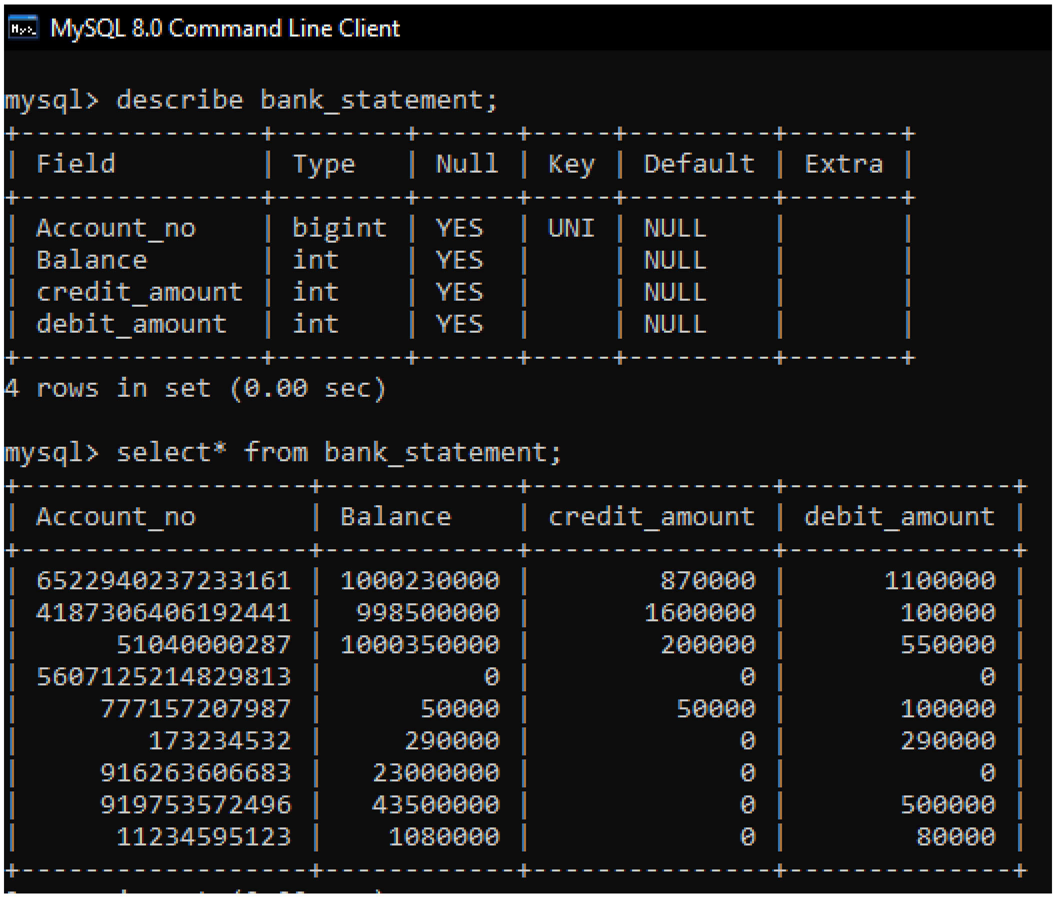
Data Table

Tables present in database


MySQL 8.0 Command Line Client

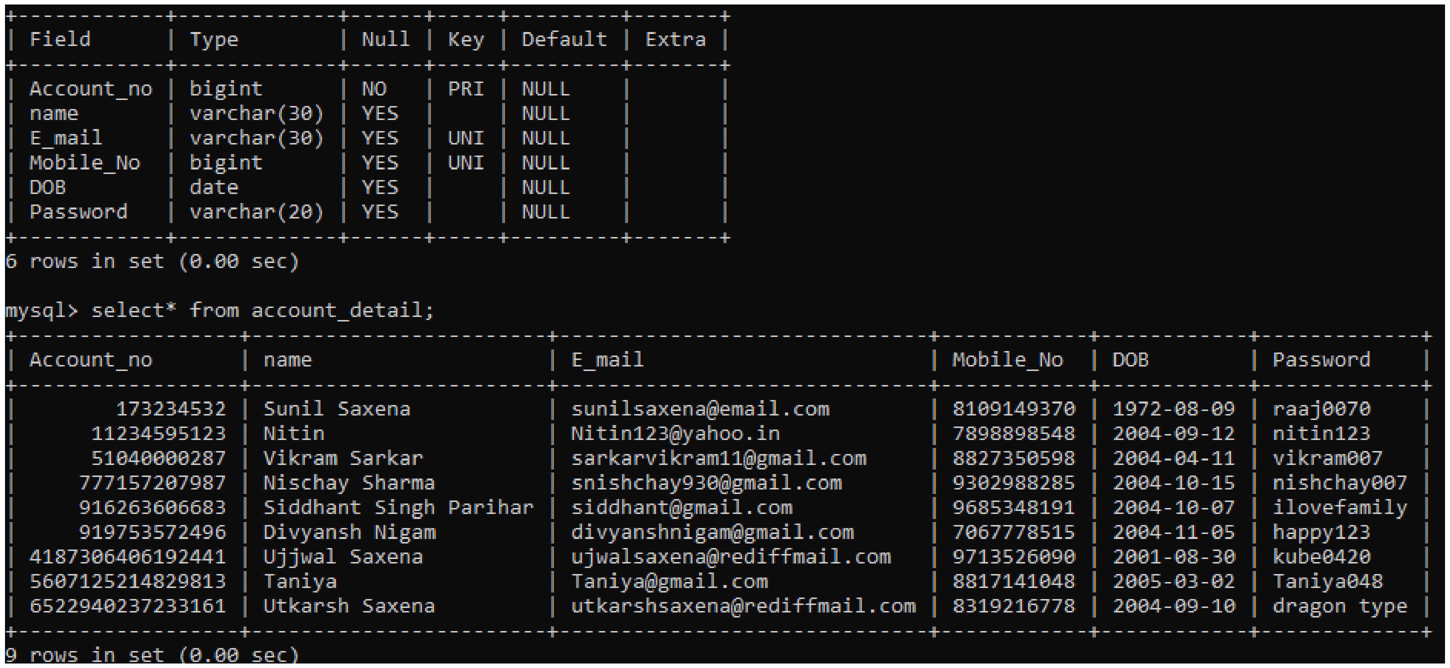
```
+------------------------------------+
| Tables_in_banking_management_system |
+------------------------------------+
| account_detail                     |
| bank_statement                     |
| transaction                        |
+------------------------------------+
3 rows in set (0.04 sec)
```

- **Bank_Statement**


MySQL 8.0 Command Line Client

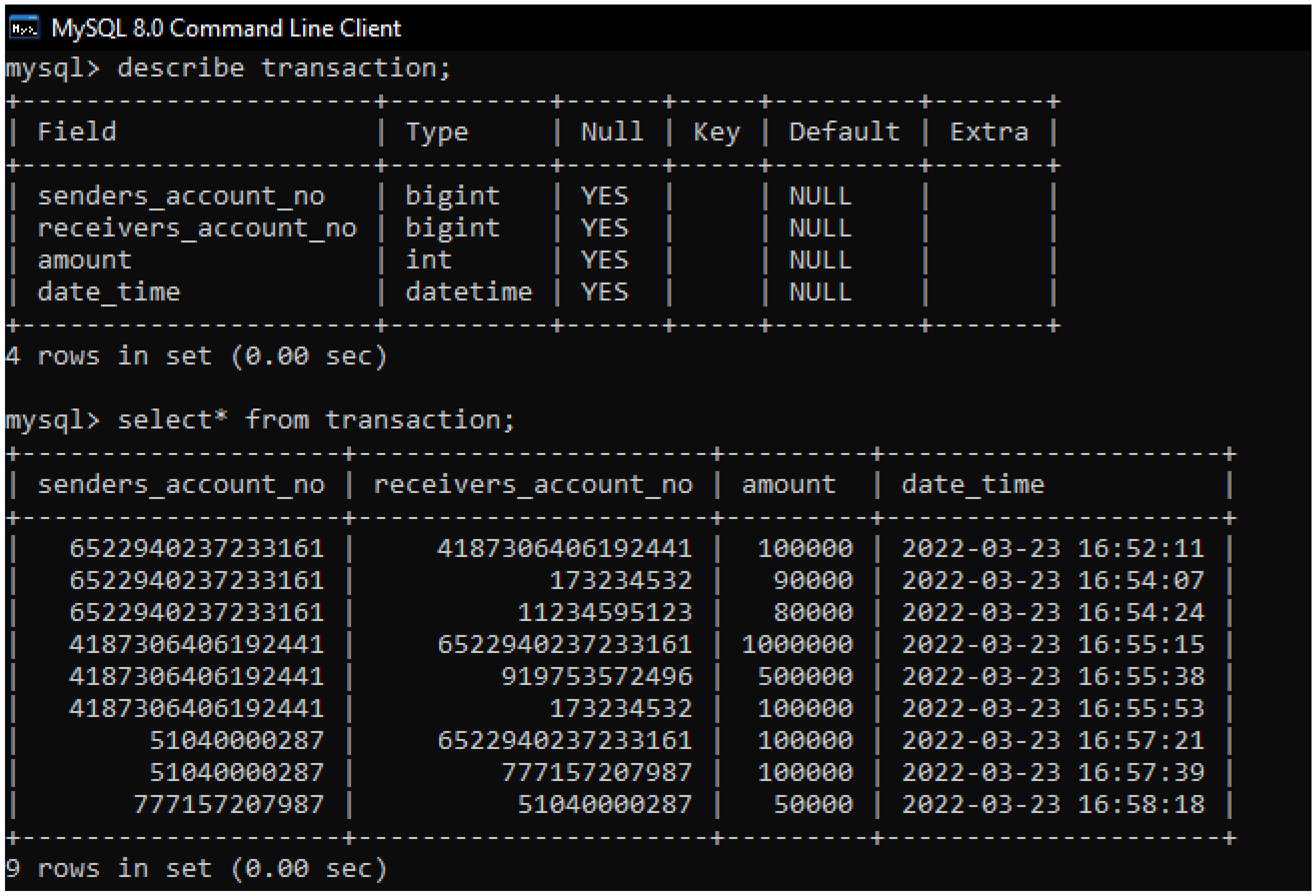
```
mysql> describe bank_statement;
+---------------+--------+------+-----+---------+-------+
| Field         | Type   | Null | Key | Default | Extra |
+---------------+--------+------+-----+---------+-------+
| Account_no    | bigint | YES  | UNI | NULL    |       |
| Balance       | int    | YES  |     | NULL    |       |
| credit_amount | int    | YES  |     | NULL    |       |
| debit_amount  | int    | YES  |     | NULL    |       |
+---------------+--------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> select* from bank_statement;
+------------------+------------+---------------+--------------+
| Account_no       | Balance    | credit_amount | debit_amount |
+------------------+------------+---------------+--------------+
| 6522940237233161 | 1000230000 |        870000 |      1100000 |
| 4187306406192441 |  998500000 |       1600000 |       100000 |
|      51040000287 | 1000350000 |        200000 |       550000 |
| 5607125214829813 |          0 |             0 |            0 |
|     777157207987 |      50000 |         50000 |       100000 |
|        173234532 |     290000 |             0 |       290000 |
|     916263606683 |   23000000 |             0 |            0 |
|     919753572496 |   43500000 |             0 |       500000 |
|      11234595123 |    1080000 |             0 |        80000 |
+------------------+------------+---------------+--------------+
```

- **Account_detail:**



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Account_no | bigint      | NO   | PRI | NULL    |       |
| name       | varchar(30) | YES  |     | NULL    |       |
| E_mail     | varchar(30) | YES  | UNI | NULL    |       |
| Mobile_No  | bigint      | YES  | UNI | NULL    |       |
| DOB        | date        | YES  |     | NULL    |       |
| Password   | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> select* from account_detail;
+------------------+------------------------+------------------------------+------------+------------+--------------+
| Account_no       | name                   | E_mail                       | Mobile_No  | DOB        | Password     |
+------------------+------------------------+------------------------------+------------+------------+--------------+
|        173234532 | Sunil Saxena           | sunilsaxena@email.com        | 8109149370 | 1972-08-09 | raaj0070     |
|      11234595123 | Nitin                  | Nitin123@yahoo.in            | 7898898548 | 2004-09-12 | nitin123     |
|      51040000287 | Vikram Sarkar          | sarkarvikram11@gmail.com     | 8827350598 | 2004-04-11 | vikram007    |
|     777157207987 | Nischay Sharma         | snishchay930@gmail.com       | 9302988285 | 2004-10-15 | nishchay007  |
|     916263606683 | Siddhant Singh Parihar | siddhant@gmail.com           | 9685348191 | 2004-10-07 | ilovefamily  |
|     919753572496 | Divyansh Nigam         | divyanshnigam@gmail.com      | 7067778515 | 2004-11-05 | happy123     |
| 4187306406192441 | Ujjwal Saxena          | ujwalsaxena@rediffmail.com   | 9713526090 | 2001-08-30 | kube0420     |
| 5607125214829813 | Taniya                 | Taniya@gmail.com             | 8817141048 | 2005-03-02 | Taniya048    |
| 6522940237233161 | Utkarsh Saxena         | utkarshsaxena@rediffmail.com | 8319216778 | 2004-09-10 | dragon type  |
+------------------+------------------------+------------------------------+------------+------------+--------------+
9 rows in set (0.00 sec)
```

- **Transaction:**



MySQL 8.0 Command Line Client

```
mysql> describe transaction;
+----------------------+----------+------+-----+---------+-------+
| Field                | Type     | Null | Key | Default | Extra |
+----------------------+----------+------+-----+---------+-------+
| senders_account_no   | bigint   | YES  |     | NULL    |       |
| receivers_account_no | bigint   | YES  |     | NULL    |       |
| amount               | int      | YES  |     | NULL    |       |
| date_time            | datetime | YES  |     | NULL    |       |
+----------------------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> select* from transaction;
+--------------------+----------------------+---------+---------------------+
| senders_account_no | receivers_account_no | amount  | date_time           |
+--------------------+----------------------+---------+---------------------+
|   6522940237233161 |     4187306406192441 |  100000 | 2022-03-23 16:52:11 |
|   6522940237233161 |            173234532 |   90000 | 2022-03-23 16:54:07 |
|   6522940237233161 |          11234595123 |   80000 | 2022-03-23 16:54:24 |
|   4187306406192441 |     6522940237233161 | 1000000 | 2022-03-23 16:55:15 |
|   4187306406192441 |         919753572496 |  500000 | 2022-03-23 16:55:38 |
|   4187306406192441 |            173234532 |  100000 | 2022-03-23 16:55:53 |
|        51040000287 |     6522940237233161 |  100000 | 2022-03-23 16:57:21 |
|        51040000287 |         777157207987 |  100000 | 2022-03-23 16:57:39 |
|       777157207987 |          51040000287 |   50000 | 2022-03-23 16:58:18 |
+--------------------+----------------------+---------+---------------------+
9 rows in set (0.00 sec)
```

USER MANUAL

On opening this program, a login page will appear which will require mobile number and password to login to the main window if you have already registered you can enter your detail and login, but if you haven't registered you can create your account from create new account button.

- **Create new account:** On clicking this button a new window will open in which you have to fill some details to create your account
  Details will include:
  i. **Account Number:** It should not be longer than 16digits and it has to be unique
  ii. **Name:** Enter you desired name, and it should not be longer than 30 characters
  iii. **Date Of Birth:** Enter it in YYYYMMDD format.
  iv. **E-mail:** It should be no longer than 30 characters. And same email cannot be entered twice.
  v. **Mobile Number:** It should be unique.
  vi. **Password and Confirm Password:** They should be same and not longer than 20 characters.

  Then after entering the details your account will be created successfully and a message box will appear.
  And after closing message box you will be directed to previous window and now you can enter you details and login.

After logging new window will open i.e., home window it contains options to perform other functions. On the left corner of this page is and info button which tells you about who has created this program. And in the center, there are three options to choose from:

1. Check Bank Balance.
2. Fund Transfer/Transfer Money
3. Profile

- **Check bank balance:** On selecting this option and clicking continue button a new window will open which will tell you your balance. It will also tell you about your total credited amount, total debited amount and it will tell you about your last five transactions with sender's account number, receiver's account number, amount and date and time.

- **Fund Transfer/Transfer Money:** On selecting this option and clicking continue button a new window will open from which you can transfer money to others. Your account number will already be there you just have to enter receiver's account number and amount and press transfer button. Your transaction will be successfully completed and a message box will appear if you have balance.
- **Profile:** On selecting this option and clicking continue a new window will open which will show you your information
  i.e., User Name, Account Number, Date of Birth, Mobile Number, E-Mail ID and Password.

This is all about the functions available in the program. And all windows have a back button on their upper left corner to go back to previous window.

CODING:

```
####************************************
#### HEADER FILE USED IN PROJECT
####************************************

import tkinter
from PIL import ImageTk,Image
import tkinter.messagebox
import mysql.connector
from functools import partial

####************************************

### Defining functions ###

##Button functions (functions which are executed after clicking buttons)

###************************************************************
***********
### 1. transaction_fnc executed during the execution of Check_balance_fnc
###    it gives recent transaction details.
###************************************************************
***********
def transaction_fnc(counter,amount=0,acc_no_receiver=None):
    global transaction    ## DECLARING GLOBAL
    if counter==1:
        #Sending data to sql table

        add="INSERT INTO Transaction Values ("
+str(userprofile[0])+","+str(acc_no_receiver)+","+str(amount)+",NOW() )"
        cursor3.execute(add)
        connect1.commit()
        cursor3.execute("select* from transaction order by date_time desc")
        transaction=cursor3.fetchall()
    elif counter==2:
        check=0
        a=cursor3.rowcount
        ycord=300
        for x in range(a):
            if check<5:
                tup=transaction[x]
                if tup[0]==userprofile[0] or tup[1]==userprofile[0]:
                    sender_acc=tup[0]
                    receiver_acc=tup[1]
                    amount1=str(tup[2])+' ₹'
                    time=tup[3]
                    lb=tkinter.Label(CBB, text=sender_acc , fg='Black', font=("Times New
Roman", 12,"bold")).place(x=140, y=ycord)
                    lb=tkinter.Label(CBB, text=receiver_acc , fg='Black', font=("Times New
Roman", 12,"bold")).place(x=300, y=ycord)
                    lb=tkinter.Label(CBB, text=amount1 , fg='Black', font=("Times New
Roman", 12,"bold")).place(x=470, y=ycord)
                    lb=tkinter.Label(CBB, text=time , fg='Black', font=("Times New Roman",
12,"bold")).place(x=550, y=ycord)
                    ycord=ycord+20
                    check=check+1


###**************************************************************
***********
### 2. Check_bank_balance_fnc executed when check balance option is selected
###    in home page it tells balance,credit and debit amount and exectues
###    transaction_fnc.
###**************************************************************
***********
def check_bank_balance_fnc():
    global userstatement   #DECLARING GLOBAL
    a=cursor2.rowcount
    x=0
    for x in range(a):
        tup=data_bank_statement[x]
```

```
      if userprofile[0]==tup[0]:
          userstatement=tup
  balance=userstatement[1]
  credit=userstatement[2]
  debit=userstatement[3]
  transaction_fnc(counter=2)
  return balance,credit,debit


###*****************************************************************
***********
### 3. amount_fnc executed while sending money from one person to another
###    it tells wheter you have sufficient amount to transfer money or not
###    and also transfers the money.
###*****************************************************************
***********
def anmount_fnc(amount,acc_no_receiver):
  global data_bank_statement    #DECLARING GLOBAL
  blank="   "*100
  a=int(0)
  a=cursor2.rowcount
  #function to apper msg box after successfully transfering money
  def msg():
      a=tkinter.messagebox.showinfo("Laxmi Cheat Fund",'Money
Transfered',parent=FT)
  for x in range(a):
      tup=data_bank_statement[x]
      if userprofile[0]==tup[0]:

          if tup[1]<amount:
              lbl=tkinter.Label(FT, text="Insufficient Balance To transfer Money" ,
fg='red', font=("Times New Roman", 9)).place(x=260,y=290)
          elif tup[1]>=amount:
              lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman",
9)).place(x=260,y=290)
              #Sending data to sql table
```

```
            minus="UPDATE bank_statement SET balance = balance - 
"+str(amount)+",credit_amount=credit_amount + " +str(amount) + " WHERE 
Account_no=" + str(userprofile[0])
            cursor2.execute(minus)
            plus="UPDATE bank_statement SET balance = balance + 
"+str(amount)+",debit_amount=debit_amount + " +str(amount) + " WHERE 
Account_no=" + str(acc_no_receiver)
            cursor2.execute(plus)
            connect1.commit()
            cursor2.execute("select* from bank_statement")
            data_bank_statement=cursor2.fetchall()
            msg()


###*************************************************************
***********
### 4. transfer_fnc executed while sending money from one person to another
###    it tells wheter you have entered valid account number or not , and also
###    calls the amount_fnc for execution.
###*************************************************************
***********
def transfer_fnc(account_entry,amount_entry):
    blank="    "*100
    data1=data_account_detail
    a=int(0)
    x=int(0)
    check=int(0)
    a=cursor1.rowcount
    for x in range (a):
        try:
            acc_no_receiver=int(account_entry.get())
            lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman", 
9)).place(x=260,y=290)
        except ValueError:
         lbl=tkinter.Label(FT, text="Enter Numbers only In Receivers Account
Number" , fg='red',
                              font=("Times New Roman", 9)).place(x=260,y=290)
         break

      try:
         amount=int(amount_entry.get())
         lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman",
9)).place(x=260,y=290)
      except ValueError:
         lbl=tkinter.Label(FT,text="Enter Numbers only In
Amount",fg='red',font=("Times New Roman",9)).place(x=260,y=290)
         break

      if acc_no_receiver==userprofile[0]:
         lbl=tkinter.Label(FT, text="Don't Enter Your Account Number" , fg='red',
font=("Times New Roman", 9)).place(x=260,y=290)
         break
      lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman",
9)).place(x=260,y=290)
      if amount<0:
         lbl=tkinter.Label(FT, text="Enter valid amount" , fg='red', font=("Times New
Roman", 9)).place(x=260,y=290)
         break
      lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman",
9)).place(x=260,y=290)
      tup=data1[x]
      if acc_no_receiver==tup[0]:
         lbl=tkinter.Label(FT, text=blank , fg='red', font=("Times New Roman",
9)).place(x=260,y=290)
         anmount_fnc(amount,acc_no_receiver)
         check=check+1
         transaction_fnc(1,amount,acc_no_receiver)
      elif x==(a-1) and check==0:
         lbl=tkinter.Label(FT, text="Account Number Dosen't exist" , fg='red',
font=("Times New Roman", 9)).place(x=260,y=290)
```

```
###*****************************************************************
***********
### 5. continue_fnc executed after clicking continue button on home page
###    it decides which window to open next .
###*****************************************************************
***********
def continue_fnc():
   a=var.get()
   if a==1:
      Check_bank_balance()
   elif a==2:
      Fund_Transfer()
   elif a==3:
      Profile()


###*****************************************************************
***********
### 6. login_fnc executes after clicking login button on login page
###    it takes user input of mobile no and password and check wheter they are
###    valid or not, and open account if they are valid.
###*****************************************************************
***********
def login_fnc(mob_entry,pass_entry):
   global userprofile     #DECLARING GLOBAL
   data1=data_account_detail
   check=0
   blank="   "*100
   a=int(0)
   x=int(0)
   a=cursor1.rowcount
   b=0
   for x in range (a):
      try:
          tup=data1[x]
          mobile_no=int(mob_entry.get())
          password=str(pass_entry.get())

       except ValueError:
          lbl=tkinter.Label(mainwindow, text="Enter Number in mobile number" ,
fg='red', font=("Times New Roman", 6)).place(x=300,y=150)
          break
       lbl=tkinter.Label(mainwindow, text=blank, fg='red', font=("Times New
Roman", 6)).place(x=300,y=150)
       if tup[3]==mobile_no and tup[5]==password :
          userprofile=tup
          lbl1=tkinter.Label(mainwindow, text=blank, fg='red', font=("Times New
Roman", 6)).place(x=300,y=195)
          check=1
          login_window()
       elif x==a-1 and check==0:
          lbl1=tkinter.Label(mainwindow, text="Invalid password or mobile number" ,
fg='red',
                       font=("Times New Roman", 6)).place(x=300,y=195)


###*************************************************************
***************
### 7. create_acc_fnc executes after clicking create button on create new account
###    window. it takes input from user and send them in sql table to create
###    new account.
###*************************************************************
***************
def
create_acc_fnc(acc_no_entry,name_entry,dob_entry,email_entry,mob_no_entry,
pass_entry,confirm_pass_entry):
   global data_account_detail    #DECLARING GLOBAL
   global data_bank_statement    #DECLARING GLOBAL
   blank="   "*100
   check=0
   cond=True
   a=cursor1.rowcount
   lbl=tkinter.Label(create_new_acc, text=blank , fg='red', font=("Times New
Roman", 6)).place(x=300,y=313)
   ##function to appear msg box after succesfully creating account

   def msg():
      a=tkinter.messagebox.showinfo("Laxmi Cheat Fund",'Account Created
Sucessfully',parent=create_new_acc)

   while True:
      try:
         name=name_entry.get()
         email=email_entry.get()
         password=pass_entry.get()
         password_confirm=confirm_pass_entry.get()
         dob=int(dob_entry.get())
         acc_no=int(acc_no_entry.get())
         mob_no=int(mob_no_entry.get())
         lbl=tkinter.Label(create_new_acc, text=blank , fg='red', font=("Times New
Roman", 6)).place(x=350,y=313)
      except ValueError:
         lbl=tkinter.Label(create_new_acc, text="Enter Valid details" , fg='red',
font=("Times New Roman", 6)).place(x=350,y=313)
         break

      for x in range(a):
         tup=data_account_detail[x]
         if tup[0]==acc_no:
            lbl=tkinter.Label(create_new_acc, text="account number already
registered." , fg='red',
                           font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
         elif tup[2]==email:
            lbl=tkinter.Label(create_new_acc, text=" E-mail already registered." ,
fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif tup[3]==mob_no:
            lbl=tkinter.Label(create_new_acc, text="Mobile number already
registered." , fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif len(email)>30:
            lbl=tkinter.Label(create_new_acc, text="E-mail id too long." , fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif len(str(acc_no))>16:
            lbl=tkinter.Label(create_new_acc, text="Max 16 digit account number is
allowed." , fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif len(str(mob_no))>10:
            lbl=tkinter.Label(create_new_acc, text="Mobile Number invalid" ,
fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif len(name)>30:
            lbl=tkinter.Label(create_new_acc, text="Name is too long" , fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        elif len(password)>20:
            lbl=tkinter.Label(create_new_acc, text="Password too long" , fg='red',
                    font=("Times New Roman", 6)).place(x=300,y=313)
            check=0
            break
        else:
            check=1

    dob1=str(dob)

    if check==1:

        if dob<10000000 or dob>99999999 or dob1[4:6]<'01' or dob1[4:6]>'12' or
dob1[6:8]<'01' or dob1[6:8]>'30' :
            lbl=tkinter.Label(create_new_acc, text="Enter Valid DOB." , fg='red',
font=("Times New Roman", 6)).place(x=300,y=313)
            check=1
        else:

            check=2
    if check==2:
        if password==password_confirm:
            check=3
        else:
            lbl=tkinter.Label(create_new_acc, text="Enter same password." , fg='red',
font=("Times New Roman", 6)).place(x=300,y=313)

    if check==3:
        #Sending data to sql table

        lbl=tkinter.Label(create_new_acc, text=blank , fg='red', font=("Times New
Roman", 6)).place(x=300,y=313)
        add="INSERT INTO
account_detail(Account_no,name,E_mail,Mobile_No,DOB,Password) VALUES (" +
str(acc_no) + "," +"'"+str(name)+"'" + "," +"'"+ email +"'" + "," + str(mob_no) + ","+
str(dob)+"," +"'"+ password+"'"+" )"
        cursor1.execute(add)
        connect1.commit()
```

```
          cursor1.execute("select* from account_detail")
          data_account_detail=cursor1.fetchall()

          add2="INSERT INTO
bank_statement(Account_no,Balance,credit_amount,debit_amount) VALUES (" +
str(acc_no)+", 0,0,0)"
          cursor2.execute(add2)
          connect1.commit()
          cursor2.execute("select* from bank_statement")
          data_bank_statement=cursor2.fetchall()
          msg()
          create_new_acc.destroy()
          break
       break


#Window function (mains windows)

###**************************************************************
***********
### 1. Check_bank_balance execute after select check bank balance in home page
###    open check bank balance window
###**************************************************************
***********

# main windows
def Check_bank_balance():
   global CBB      #DECLARING GLOBAL

   CBB=tkinter.Toplevel(mainwindow)
   CBB.geometry("800x450")
   CBB.title("Laxmi Cheat Fund")
   balance,credit,debit=check_bank_balance_fnc() ##calling function

   balance1=str(balance)+" ₹"
   debit1="Debit Amount: " +str(debit) +" ₹"
```

```
    credit1="Credit Amount: " +str(credit) +" ₹"

    # label widget
    lbl=tkinter.Label(CBB, text="Bank Balance" , fg='Black', font=("Times New
Roman", 32,"bold")).place(x=250, y=25)
    lb2=tkinter.Label(CBB, text=balance1 , fg='dimgray', font=("Times New Roman",
24,"bold")).place(x=300, y=80)
    lb3=tkinter.Label(CBB, text="Accumulative:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=140)
    lb4=tkinter.Label(CBB ,text=debit1, fg='Black', font=("Times New Roman",
12,"bold")).place(x=160, y=170)
    lb5=tkinter.Label(CBB, text=credit1,fg='Black', font=("Times New Roman",
12,"bold")).place(x=160, y=200)
    lb6=tkinter.Label(CBB, text="Recent Transactions:" , fg='Black', font=("Times
New Roman", 16,"bold")).place(x=140, y=250)
    lb7=tkinter.Label(CBB, text="Sender's Account No     Receiver's Account No
Amount     Date and Time" , fg='Black',
                    font=("Times New Roman", 12,"bold")).place(x=140, y=280)



    # insterting picture
    canvas=tkinter.Canvas(CBB,width=0,height=0)
    canvas.place(x=0,y=0)
    img=ImageTk.PhotoImage(Image.open("arrow3.png"))
    canvas.create_image(00,0,image=img)
    label = tkinter.Label(CBB,image=img)
    label.image =img # keep a reference

    canvas2=tkinter.Canvas(CBB,width=800,height=5)
    canvas2.place(x=100,y=120)
    img2=ImageTk.PhotoImage(Image.open("line1.png"))
    label2=tkinter.Label(CBB,image=img2)
    label2.image=img2
    canvas2.create_image(200,10,image=img2)

    canvas3=tkinter.Canvas(CBB,width=800,height=5)
    canvas3.place(x=100,y=230)
    img3=ImageTk.PhotoImage(Image.open("line1.png"))
    label3=tkinter.Label(CBB,image=img3)
    label3.image=img3
    canvas3.create_image(200,10,image=img3)



    #button

b1=tkinter.Button(CBB,image=img,text="pic",font=("arial",14),bd=0,command=CB
B.destroy).place(x=30,y=20)


###*************************************************************
***********
### 2. Fund_Transfer executes after select transfer money in home page
###    open window to send money to other
###*************************************************************
***********
def Fund_Transfer():
    global FT       #DECLARING GLOBAL
    FT=tkinter.Toplevel(mainwindow)
    FT.geometry("800x450")
    FT.title("Laxmi Cheat Fund")

    acc_no=userprofile[0]
    # label widget
    lbl=tkinter.Label(FT, text="Transfer Money" , fg='Black', font=("Times New
Roman", 32,"bold")).place(x=250, y=25)
    lb2=tkinter.Label(FT, text="Your Account Number:" , fg='Black', font=("Times
New Roman", 14,"bold")).place(x=140, y=100)
    lb3=tkinter.Label(FT, text=acc_no , fg="dimgray", font=("Times New Roman",
16,"bold")).place(x=140, y=130)
    lb4=tkinter.Label(FT ,text="Receivers Account Number:", fg='Black',
font=("Times New Roman", 16,"bold")).place(x=140, y=160)
```

```
   lb5=tkinter.Label(FT, text="Amount:",fg='Black', font=("Times New Roman",
16,"bold")).place(x=140, y=220)

   #entry
   account_entry=tkinter.Entry(FT,width=30,fg='Grey',font=("Times New
Roman",14))
   account_entry.place(x=140,y=190)
   amount_entry=tkinter.Entry(FT,width=12,fg='Grey',font=("Times New
Roman",14))
   amount_entry.place(x=230,y=220)

   # insterting picture
   canvas=tkinter.Canvas(FT,width=0,height=0)
   canvas.place(x=0,y=0)
   img=ImageTk.PhotoImage(Image.open("arrow3.png"))
   canvas.create_image(00,0,image=img)
   label = tkinter.Label(FT,image=img)
   label.image =img # keep a reference

   canvas2=tkinter.Canvas(FT,width=800,height=5)
   canvas2.place(x=100,y=80)
   img2=ImageTk.PhotoImage(Image.open("line1.png"))
   label2=tkinter.Label(FT,image=img2)
   label2.image=img2
   canvas2.create_image(200,10,image=img2)

   #button

b1=tkinter.Button(FT,text="Transfer",bg='green',fg='white',font=("arial",14),active
background='light green',

command=partial(transfer_fnc,account_entry,amount_entry)).place(x=320,y=310)

b2=tkinter.Button(FT,image=img,text="pic",font=("arial",14),bd=0,command=FT.d
estroy).place(x=30,y=20)
```

```
###*****************************************************************
***********
### 3. Check_bank_balance execute after selecting profile in home page
###    open profile window which show all available detail about account
###*****************************************************************
***********
def Profile():
   global Profile      #DECLARING GLOBAL
   profile=tkinter.Toplevel(mainwindow)
   profile.geometry("800x450")
   profile.title("Laxmi Cheat Fund")
   # label widget

   lb=tkinter.Label(profile, text="User Profile" , fg='Black', font=("Times New
Roman", 32,"bold")).place(x=280, y=25)
   lb1=tkinter.Label(profile, text="User Name:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=110)
   lb2=tkinter.Label(profile, text="Account No:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=145)
   lb3=tkinter.Label(profile, text="Date of Birth:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=180)
   lb4=tkinter.Label(profile, text="Mobile No:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=215)
   lb5=tkinter.Label(profile, text="E-mail ID:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=250)
   lb6=tkinter.Label(profile, text="Password:" , fg='Black', font=("Times New
Roman", 16,"bold")).place(x=140, y=285)
   acc_no,name,email,mob_no,dob,password=userprofile
   Lb1=tkinter.Label(profile, text=name , fg='Black', font=("Times New Roman",
16,"bold")).place(x=270, y=110)
   Lb2=tkinter.Label(profile, text=acc_no , fg='Black', font=("Times New Roman",
16,"bold")).place(x=270, y=145)
   Lb3=tkinter.Label(profile, text=dob , fg='Black', font=("Times New Roman",
16,"bold")).place(x=270, y=180)
```

```
   Lb4=tkinter.Label(profile, text=mob_no , fg='Black', font=("Times New Roman", 
16,"bold")).place(x=270, y=215)
   Lb5=tkinter.Label(profile, text=email , fg='Black', font=("Times New Roman", 
16,"bold")).place(x=270, y=250)
   Lb6=tkinter.Label(profile, text=password , fg='Black', font=("Times New Roman", 
16,"bold")).place(x=270, y=285)
   # insterting picture
   canvas=tkinter.Canvas(profile,width=0,height=0)
   canvas.place(x=0,y=0)
   img=ImageTk.PhotoImage(Image.open("arrow3.png"))
   canvas.create_image(00,0,image=img)
   label = tkinter.Label(profile,image=img)
   label.image =img # keep a reference

   canvas2=tkinter.Canvas(profile,width=800,height=5)
   canvas2.place(x=100,y=80)
   img2=ImageTk.PhotoImage(Image.open("line1.png"))
   label2=tkinter.Label(profile,image=img2)
   label2.image=img2
   canvas2.create_image(200,10,image=img2)

   #button

b1=tkinter.Button(profile,image=img,text="pic",font=("arial",14),bd=0,command=
profile.destroy).place(x=30,y=20)


###*************************************************************
***********
### 4. login_window opens after succesfully logging in
###    opens home page to select further options
###*************************************************************
***********
def login_window():
   global login    #DECLARING GLOBAl
   login= tkinter.Toplevel(mainwindow)
```

```
    login.geometry("800x450")
    login.title("Laxmi Cheat Fund")
    #Create a Label in New window
    tkinter.Label(login, text="Laxmi Cheat Fund", font=("Times New Roman", 32,
'bold')).place(x=220,y=25)

    #adding check box
    global var
    var=tkinter.IntVar()
    R1=tkinter.Radiobutton(login,text="Check bank balance",font=("Times New
Roman",16),variable=var,value=1).place(x=250,y=100)
    R2=tkinter.Radiobutton(login,text="Fund Transfer/Transfer
money",font=("Times New Roman",16),variable=var,value=2).place(x=250,y=150)
    R3=tkinter.Radiobutton(login,text="Profile",font=("Times New
Roman",16),variable=var,value=3).place(x=250,y=200)



    #adding button



    #inserting picture
    canvas=tkinter.Canvas(login,width=0,height=0)
    canvas.place(x=0,y=0)
    img=ImageTk.PhotoImage(Image.open("info2.png"))
    canvas.create_image(0,0,image=img)
    canvas2=tkinter.Canvas(login,width=0,height=0)
    canvas2.place(x=0,y=0)
    img2=ImageTk.PhotoImage(Image.open("arrow3.png"))
    canvas2.create_image(00,0,image=img)
    label = tkinter.Label(login,image=img)
    label.image =img # keep a reference
    label2 = tkinter.Label(login,image=img2)
    label2.image =img2 # keep a referenc

    #defining function
    def msg():
        text='Developed by:\nNishchay Sharma\nUtkarsh Saxena\nVikram Sarkar'
        a=tkinter.messagebox.showinfo("Laxmi Cheat Fund",text,parent=login)


    #button

B1=tkinter.Button(login,text="continue",font=("arial",16),bg='green',fg='white',co
mmand=continue_fnc).place(x=290,y=270)

B2=tkinter.Button(login,image=img,text="pic",font=("arial",14),bd=0,command=m
sg).place(x=35,y=380)

b2=tkinter.Button(login,image=img2,text="pic",font=("arial",14),bd=0,command=l
ogin.destroy).place(x=30,y=20)


###*************************************************************
**********************
### 5. create_new_acc_window opens after clicking create new account button on
login window
###    open create new account window to create account
###*************************************************************
**********************
def create_new_acc_window():
    global create_new_acc     #DECLARING GLOBAL
    create_new_acc=tkinter.Toplevel(mainwindow)
    create_new_acc.geometry("800x450")
    create_new_acc.title("Laxmi Cheat Fund")



    # label widget
    lbl=tkinter.Label(create_new_acc, text="Create New Account" , fg='Black',
font=("Times New Roman", 32,"bold")).place(x=220, y=25)
    lb2=tkinter.Label(create_new_acc, text="Account Number:" , fg='Black',
font=("Times New Roman", 12,"bold")).place(x=160, y=110)
```

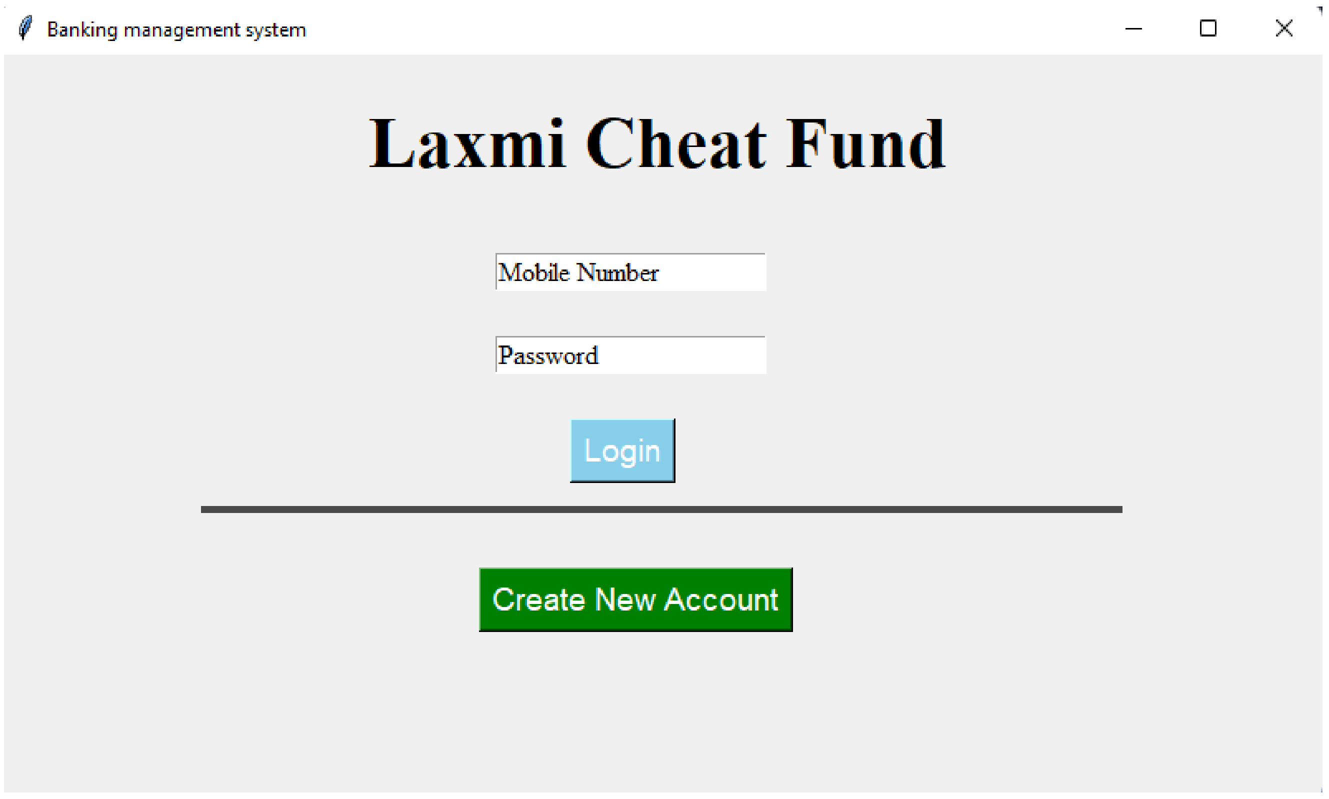
```
    lb3=tkinter.Label(create_new_acc, text="Name:" , fg='Black', font=("Times New
Roman", 12,"bold")).place(x=160, y=140)
    lb4=tkinter.Label(create_new_acc, text="Date of Birth:" , fg='Black',
font=("Times New Roman", 12,"bold")).place(x=160, y=170)
    lb5=tkinter.Label(create_new_acc, text="E-mail ID:" , fg='Black', font=("Times
New Roman", 12,"bold")).place(x=160, y=200)
    lb6=tkinter.Label(create_new_acc, text="Mobile No.:" , fg='Black', font=("Times
New Roman", 12,"bold")).place(x=160, y=230)
    lb7=tkinter.Label(create_new_acc, text="Password:" , fg='Black', font=("Times
New Roman", 12,"bold")).place(x=160, y=260)
    lb8=tkinter.Label(create_new_acc, text="Confirm Password:" , fg='Black',
font=("Times New Roman", 12,"bold")).place(x=160, y=290)

    #entry
    acc_no_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times
New Roman",12))
    acc_no_entry.place(x=300,y=110)
    name_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times
New Roman",12))
    name_entry.place(x=300,y=140)
    dob_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times New
Roman",12))
    dob_entry.insert(12,"YYYY/MM/DD")
    dob_entry.place(x=300,y=170)
    email_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times
New Roman",12))
    email_entry.place(x=300,y=200)
    mob_no_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times
New Roman",12))
    mob_no_entry.place(x=300,y=230)
    pass_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Times New
Roman",12))
    pass_entry.place(x=300,y=260)

confirm_pass_entry=tkinter.Entry(create_new_acc,width=30,fg='Black',font=("Tim
es New Roman",12))
```

```
    confirm_pass_entry.place(x=300,y=290)
    # insterting picture
    canvas=tkinter.Canvas(create_new_acc,width=0,height=0)
    canvas.place(x=0,y=0)
    img=ImageTk.PhotoImage(Image.open("arrow3.png"))
    canvas.create_image(00,0,image=img)
    label = tkinter.Label(create_new_acc,image=img)
    label.image =img # keep a reference



    #button
    b1=tkinter.Button(create_new_acc,text="Create
account",bg='green',fg='white',font=("arial",14),activebackground='light green',

command=partial(create_acc_fnc,acc_no_entry,name_entry,dob_entry,email_ent
ry,mob_no_entry,
                              pass_entry,confirm_pass_entry)).place(x=320,y=330)

b2=tkinter.Button(create_new_acc,image=img,text="pic",font=("arial",14),bd=0,co
mmand=create_new_acc.destroy).place(x=30,y=20)



###*****************************************************************
****************
### 7. main_window main function of the program executes just after running the
###    program, it opens login window which allows further operation to take place
###*****************************************************************
****************
def main_window():
    global mainwindow       #DECLARING GLOBAL
    mainwindow=tkinter.Tk()
    # add widgets here

    mainwindow.title('Banking management system')
    mainwindow.geometry("800x450")

    # label widget
    lbl=tkinter.Label(mainwindow, text="Laxmi Cheat Fund" , fg='Black',
font=("Times New Roman", 32,"bold"))
    lbl.place(x=220, y=25)

    #inserting entry
    mob_entry=tkinter.Entry(mainwindow,fg='black',font=("Times New Roman",12))
    mob_entry.insert(10,"Mobile Number")
    mob_entry.place(x=300,y=120)

    pass_entry=tkinter.Entry(mainwindow,fg='black',font=("Times New Roman",12))
    pass_entry.insert(10,"Password")
    pass_entry.place(x=300,y=170)

    # insterting picture
    canvas=tkinter.Canvas(mainwindow,width=800,height=5)
    canvas.place(x=120,y=270)
    img=ImageTk.PhotoImage(Image.open("line1.png"))
    canvas.create_image(200,10,image=img)



    # intsert button
    b1=tkinter.Button(mainwindow,text="Login",bg="sky
blue",font=("arial",14),fg="white",activebackground="Light blue",
                      command=partial(login_fnc,mob_entry,pass_entry))
    b1.place(x=345,y=220)

    b2=tkinter.Button(mainwindow ,text="Create New
Account",bg="green",font=("arial",14),fg="white",activebackground="light green",
                      command=create_new_acc_window)
    b2.place(x=290,y=310)

    mainwindow.mainloop()
```

```
###*****************************************************************
***********
### 1. connector_function connects mysql with python
###*****************************************************************
***********
def connector_function():

connect1=mysql.connector.connect(host="localhost",user="root",passwd='raaj007
0',database="banking_management_system")
    return connect1

connect1 = connector_function()
if connect1.is_connected()==False:
    print("error")
cursor1=connect1.cursor()

#### Taking data from all the tables present in banking_management_system
database
cursor1.execute("select* from account_detail")
data_account_detail=cursor1.fetchall()

cursor2=connect1.cursor()
cursor2.execute("select* from bank_statement")
data_bank_statement=cursor2.fetchall()

cursor3=connect1.cursor()
cursor3.execute("select* from transaction order by date_time desc")
transaction=cursor3.fetchall()

main_window()
connect1.close()
```

INPUT AND OUTPUT SCREEN

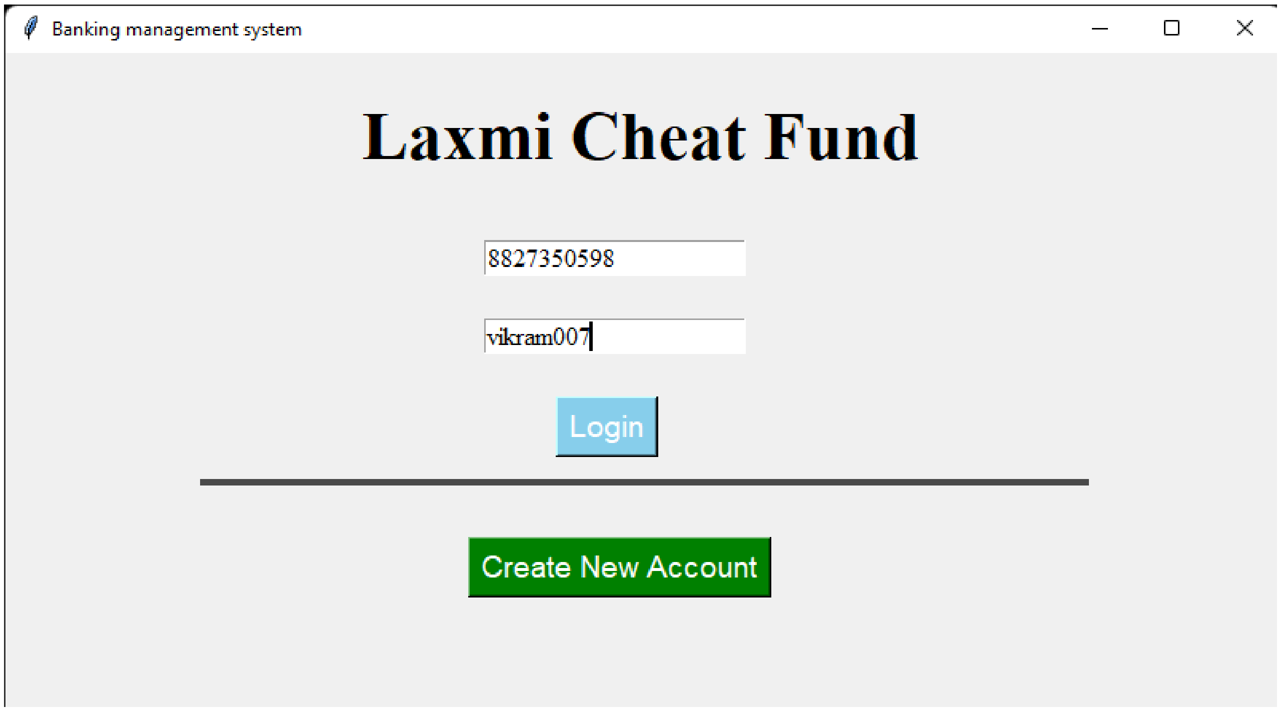
Login Window:



Banking management system

Laxmi Cheat Fund

Entering registered mobile number and password:



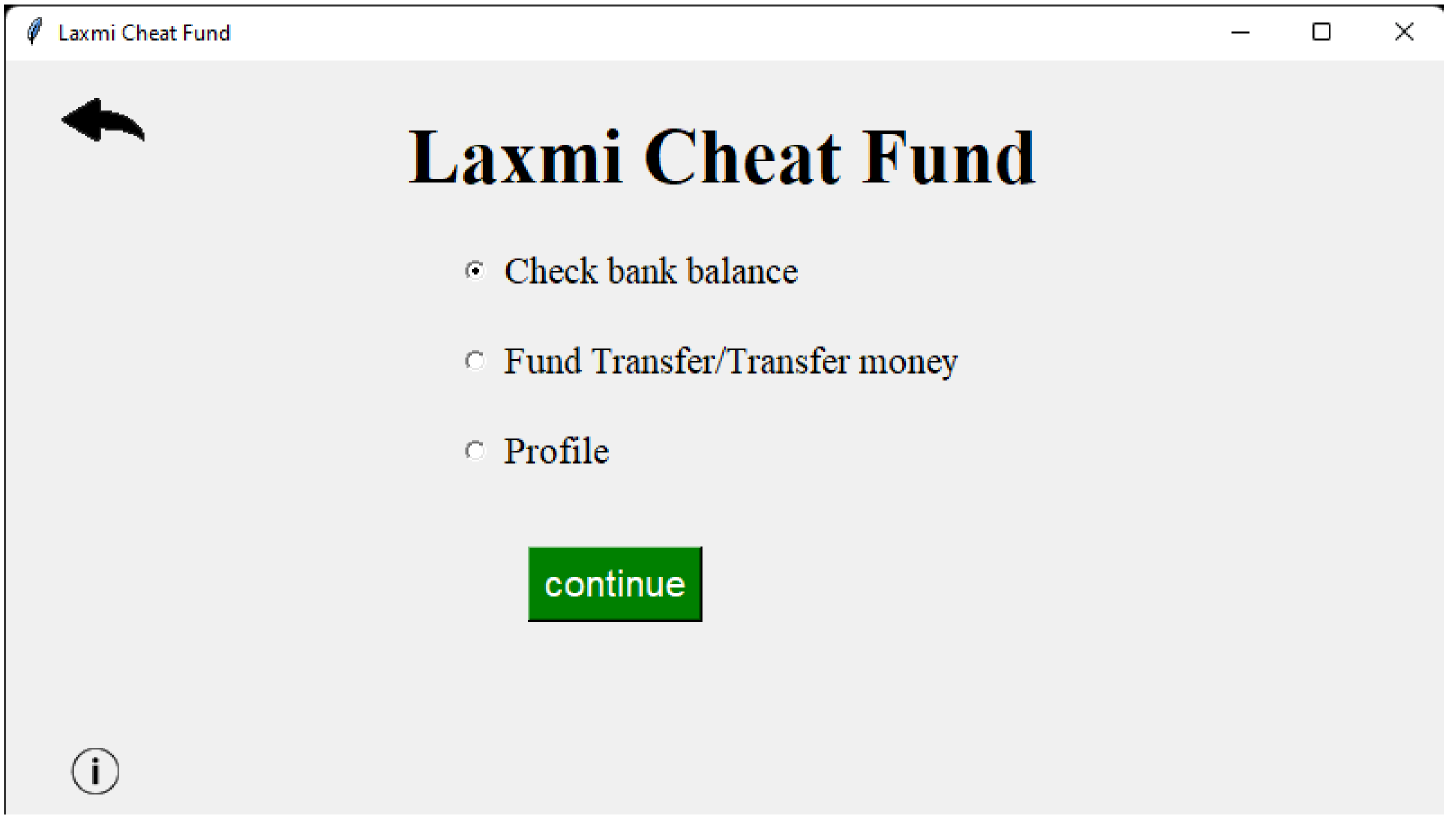
Banking management system

Laxmi Cheat Fund

8827350598

vikram007

Home window :


Laxmi Cheat Fund

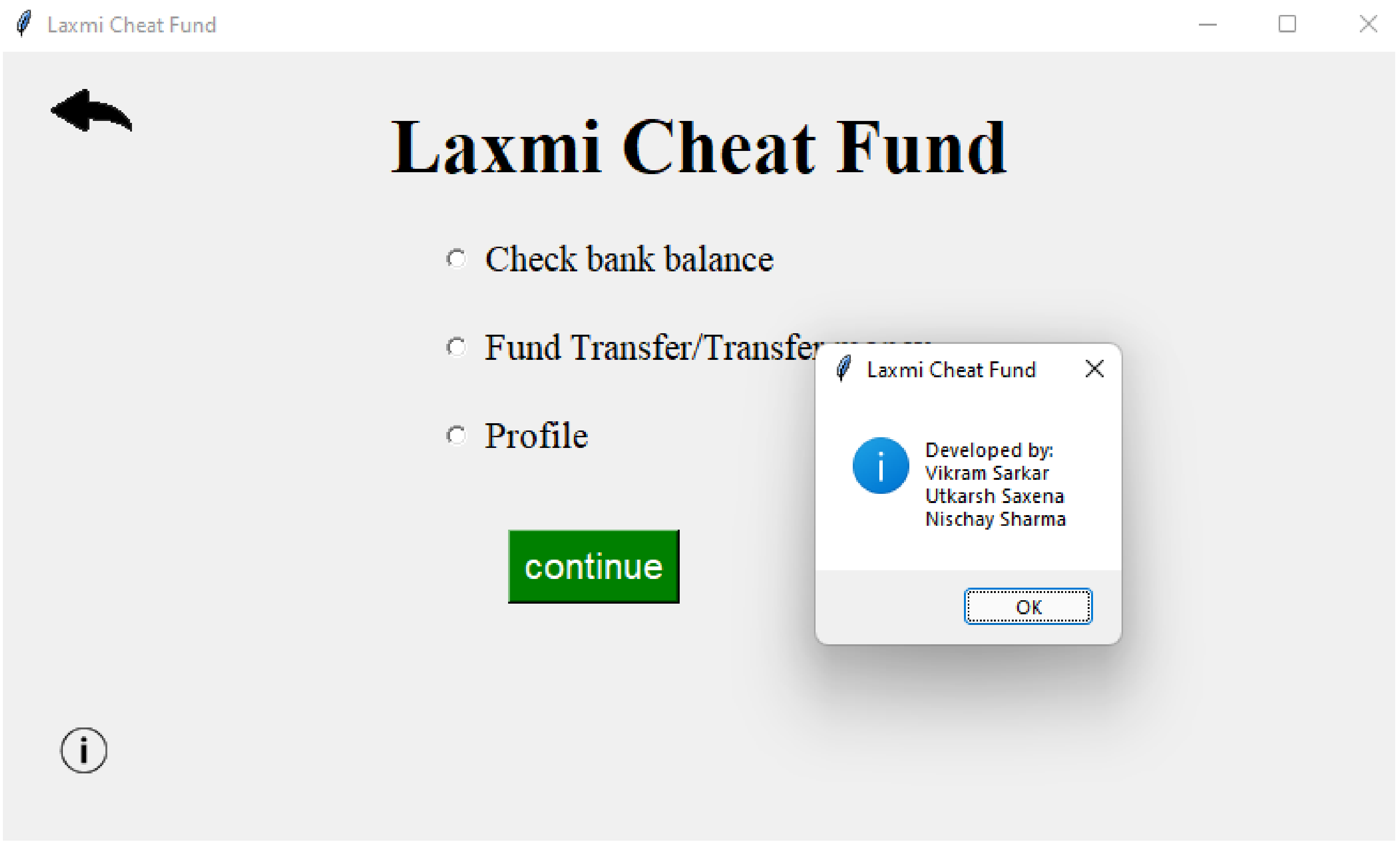
Laxmi Cheat Fund

(•) Check bank balance

( ) Fund Transfer/Transfer money

( ) Profile

On Pressing info button:


Laxmi Cheat Fund

Laxmi Cheat Fund

( ) Check bank balance

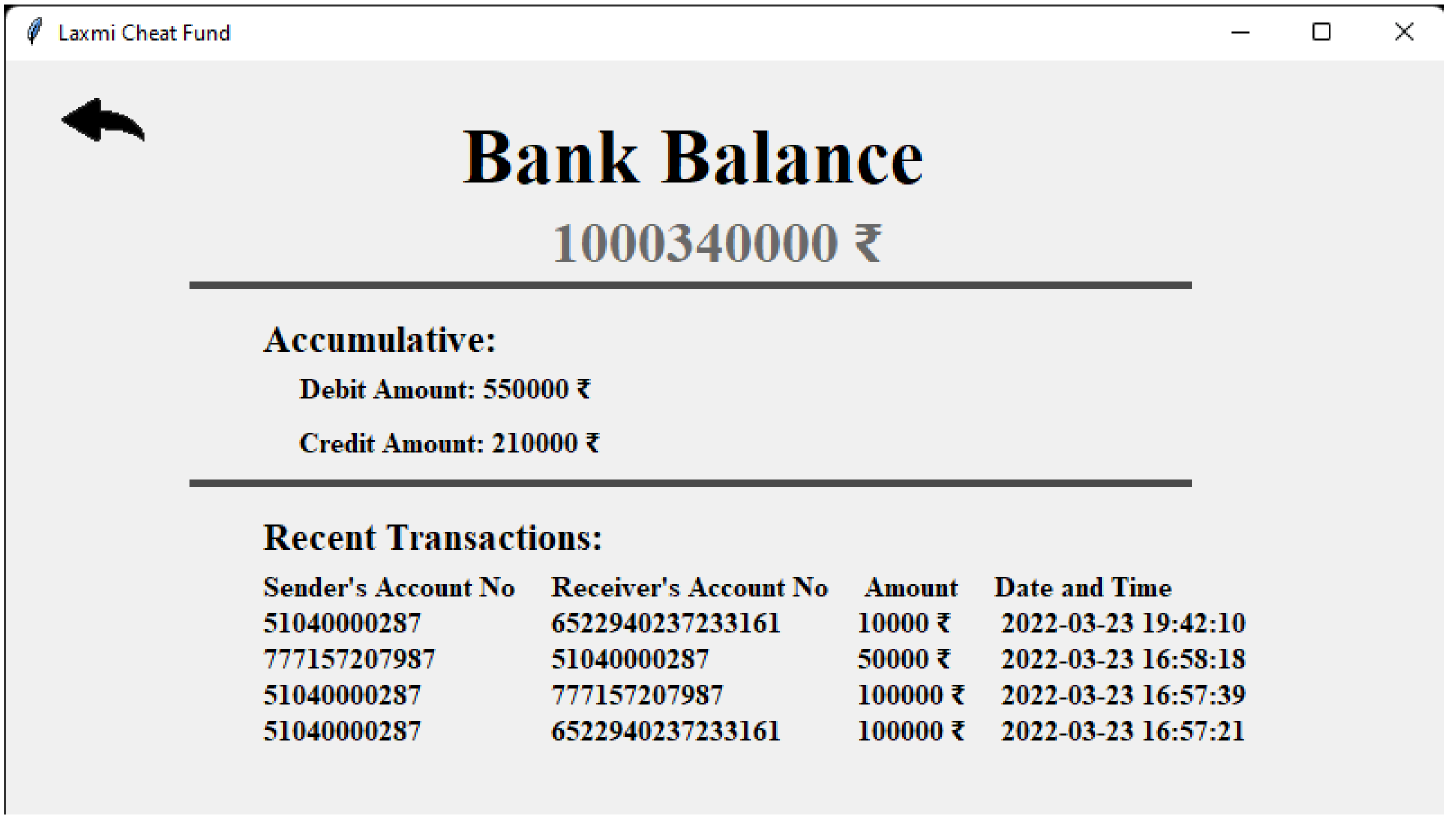
( ) Fund Transfer/Transfer money

( ) Profile

Laxmi Cheat Fund

Developed by:
Vikram Sarkar
Utkarsh Saxena
Nischay Sharma

On Selecting Check balance:



Bank Balance

1000340000 ₹

Accumulative:

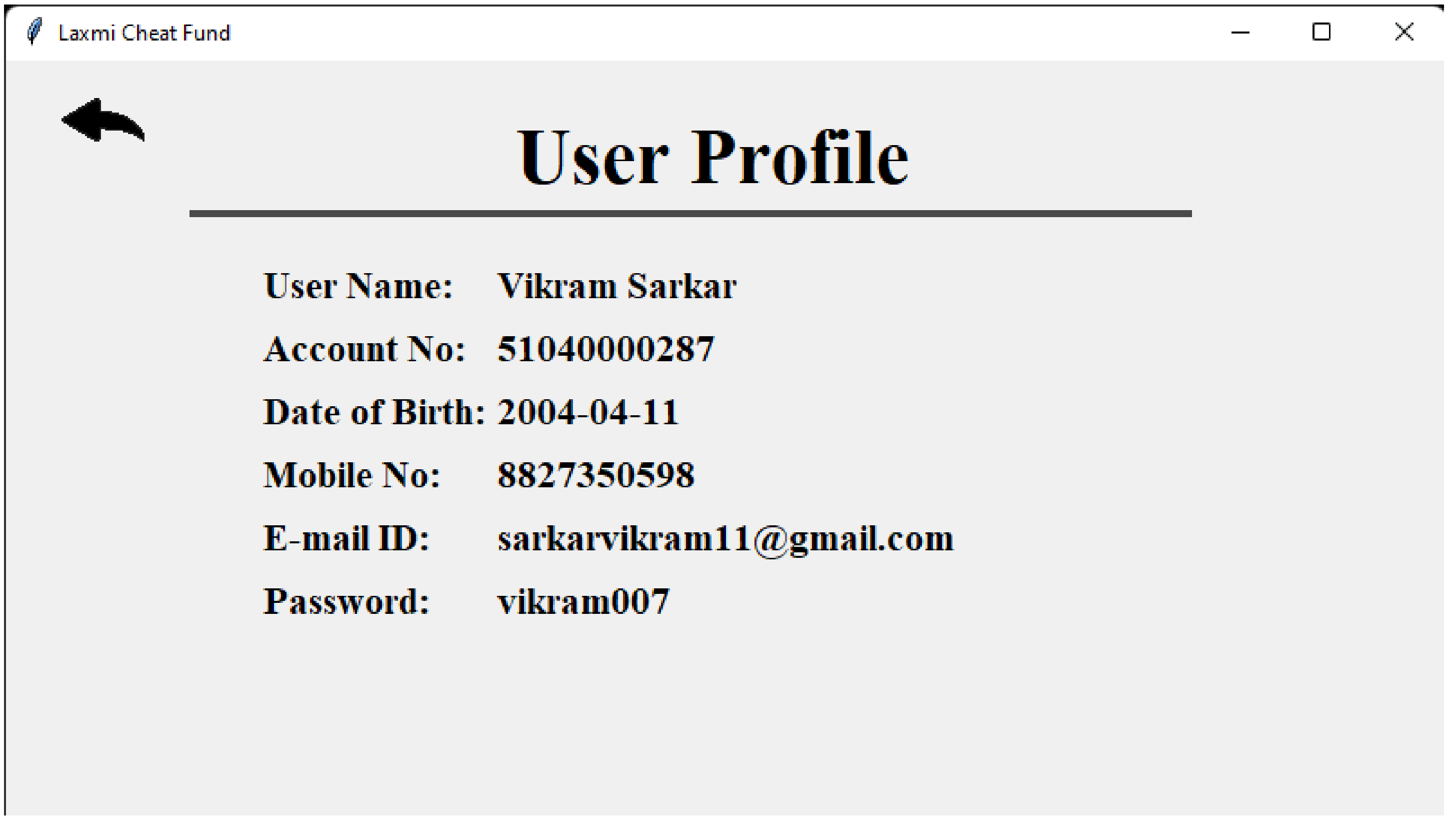
Debit Amount: 550000 ₹

Credit Amount: 210000 ₹

Recent Transactions:

Sender's Account No	Receiver's Account No	Amount	Date and Time
51040000287	6522940237233161	10000 ₹	2022-03-23 19:42:10
777157207987	51040000287	50000 ₹	2022-03-23 16:58:18
51040000287	777157207987	100000 ₹	2022-03-23 16:57:39
51040000287	6522940237233161	100000 ₹	2022-03-23 16:57:21

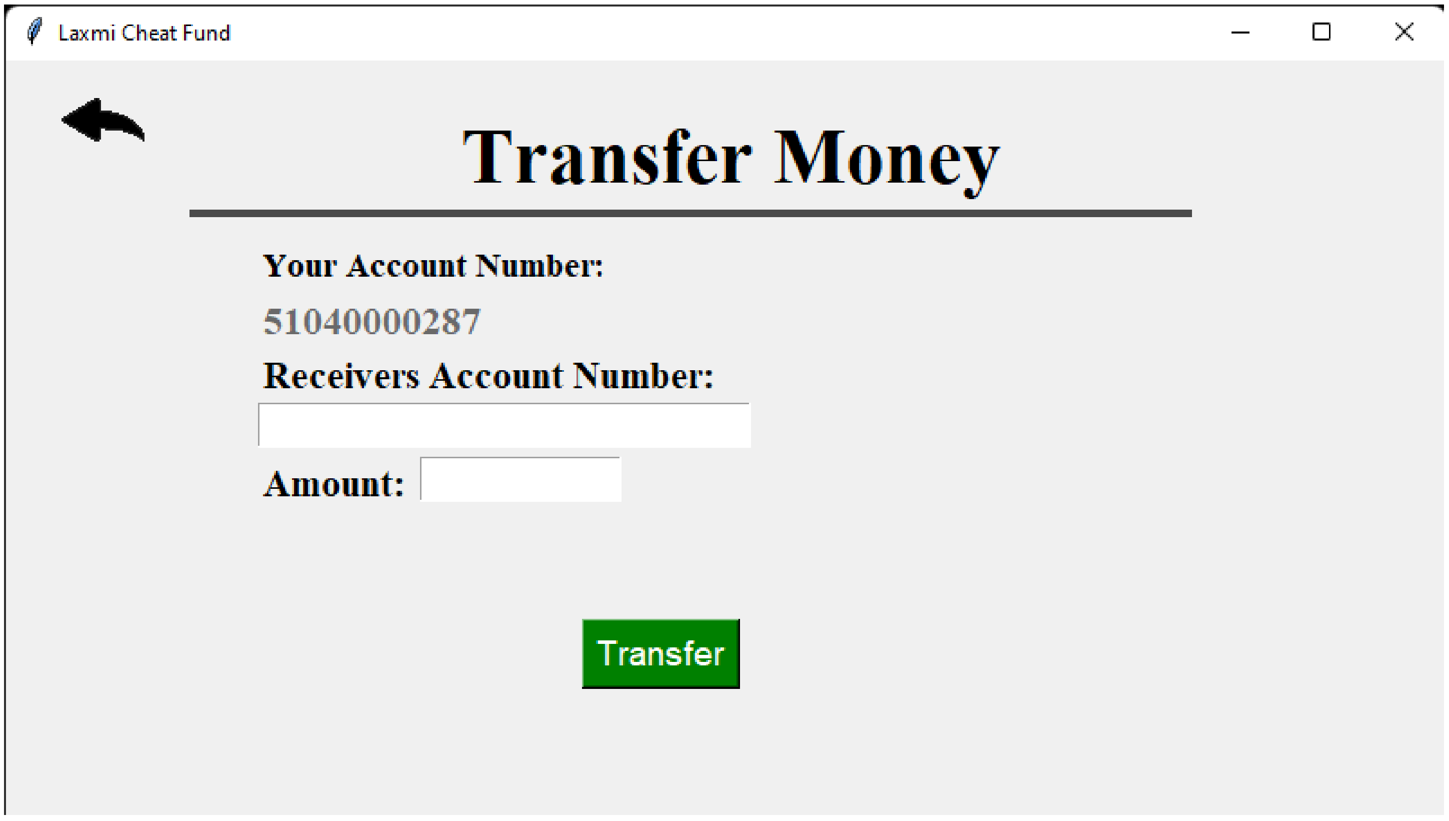
On Selecting Profile:



User Profile

User Name:	Vikram Sarkar
Account No:	51040000287
Date of Birth:	2004-04-11
Mobile No:	8827350598
E-mail ID:	sarkarvikram11@gmail.com
Password:	vikram007

On Selecting Transfer Money:



Laxmi Cheat Fund

Transfer Money

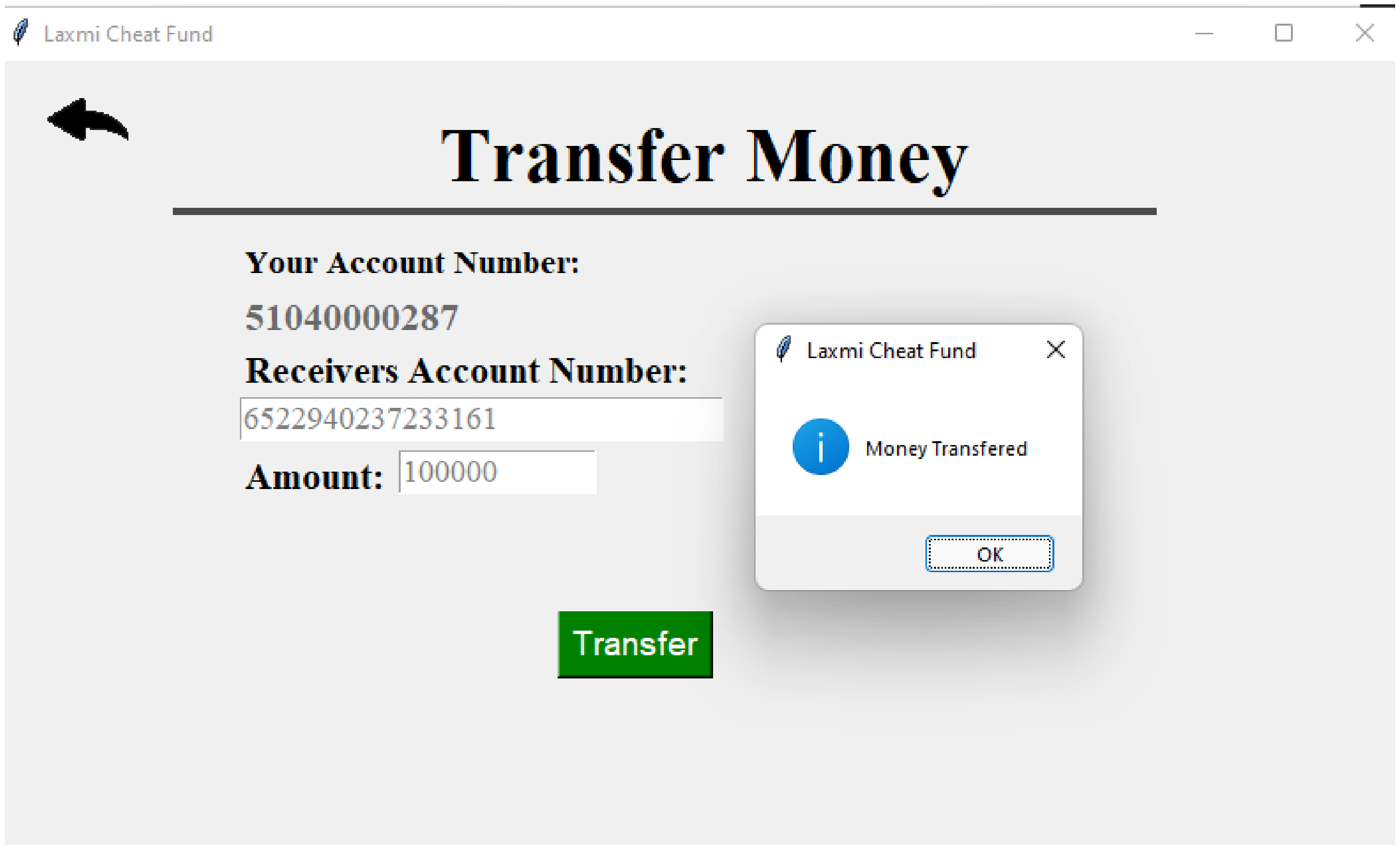
Your Account Number:

51040000287

Receivers Account Number:

Amount:

On Entering details and pressing Transfer:



Laxmi Cheat Fund

Transfer Money

Your Account Number:

51040000287

Receivers Account Number:

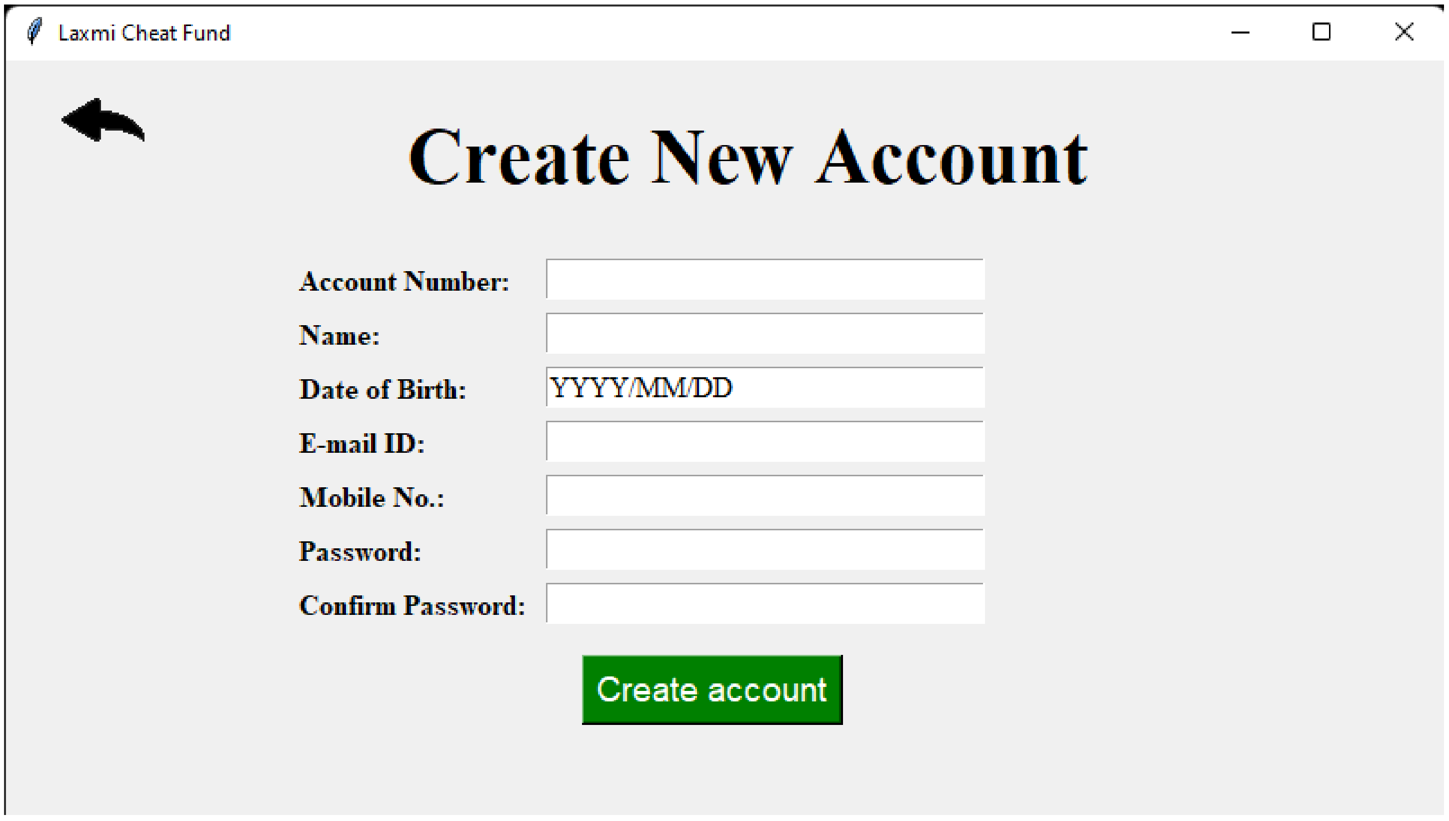
6522940237233161

Amount: 100000

Laxmi Cheat Fund

Money Transfered

On Selecting Create New Account:

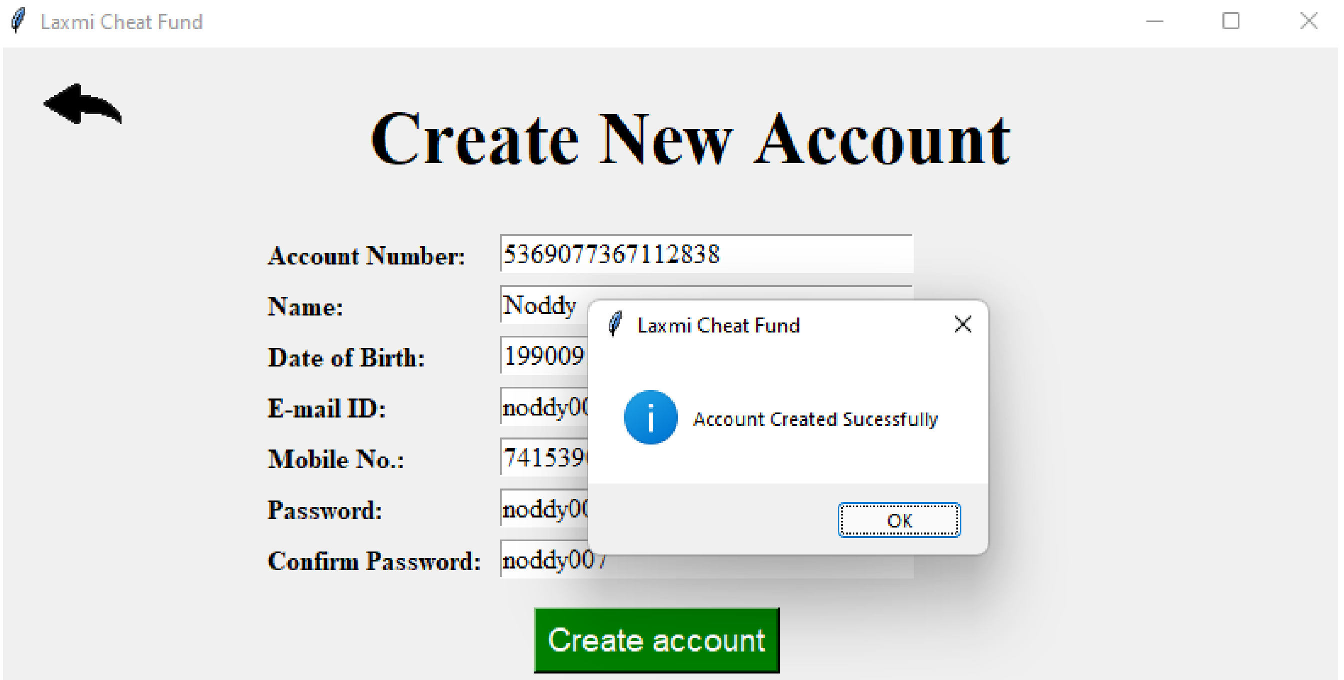


Laxmi Cheat Fund

Create New Account

Account Number:	
Name:	
Date of Birth:	YYYY/MM/DD
E-mail ID:	
Mobile No.:	
Password:	
Confirm Password:	

On Entering correct detail and pressing create account:



Laxmi Cheat Fund

Create New Account

Account Number:	5369077367112838
Name:	Noddy
Date of Birth:	199009
E-mail ID:	noddy0
Mobile No.:	741539
Password:	noddy0
Confirm Password:	noddy007

Laxmi Cheat Fund

Account Created Sucessfully

TESTING AND DEBUGGING

INTRODUCTION:

The implementation phase of software development is concerned with translating design specification into source code. The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages.

The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

TERMS IN TESTING FUNDAMENTAL

1. <u>Error</u>

The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also to use to refer to human action that result in software containing a defect or fault.

2. <u>Fault</u>

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

3. <u>Failure</u>

Failure is the inability of a system or component to perform a required function according to its specifications. A software failure occurs if the behavior of the software is the different from the specified behavior. Failure may be caused due to functional or performance reasons.

a. <u>Unit Testing</u>

The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system.

A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

b. <u>Module Testing</u>

A module and encapsulates related component. So can be tested without other system module.

c. <u>Subsystem Testing</u>

Subsystem testing may be independently design and implemented common problems are sub-system interface mistake in this checking we concentrate on it.

There are four categories of tests that a programmer will typically perform on a program unit.

1) Functional test
2) Performance test
3) Stress test
4) Structure test

1) <u>Functional Test</u>

Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

2) <u>Performance Test</u>

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit. A certain amount of avoid expending too much effort on fine-tuning of a program unit that contributes little to the overall performance of the entire system. Performance testing is most productive at the subsystem and system levels.

3) <u>Stress Test</u>

Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

4) <u>Structure Test</u>

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test date to exercise those paths, determining the test coverage criterion to be used, executing the test, and measuring the test coverage achieved when the test cases are exercised.

<u>DEBUGGING</u>

Defect testing is intended to find areas where the program does not confirm to its specifications. Tests are designed to reveal the presence of defect in the system. When defect have been found in the program. There must be discovered and removed. This is called “Debugging”.

CONCLUSION

It was a wonderful and learning experience for me while working on this project. This project took me through the various phases of project development and gave me real insight into the world of computer science. The joy of work and the thrill involved while tackling the various problems and challenges gave me a feel of developer's industry.

REFERENCES

BOOK: Computer science with python class 12 by Sumita Arora

WEBSITES: www.wikipedia.com
www.geeksforgeeks.org
www.javatpoint.com
www.gist.github.com
www.stackoverflow.com

SEARCH ENGINES: MSN and GOOGLE