**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

|  |  |  |
| --- | --- | --- |
|  | **A** |  |
| **MINI PROJECT REPORT** | | |
| ***on*** | | |
| **ATM TRANSACTION SYSTEM** | | |
| **BACHELOR OF ENGINEERING**  **in**  **COMPUTER SCIENCE AND ENGINEERING** | | |
| **Submitted by**  **N Vamshee Teja - 2453-18-733-164**  **P Jashwanth - 2453-18-733-167**  **V Manjunath - 2453-18-733-177**  **B Anupama - 2453-18-733-128** | | |
| **Under the Guidance of**  **Mrs. Vijaya Madhavi**    **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  **Neil Gogte Institute of Technology**  Kachawanisingaram Village, Hyderabad, Telangana 500058.  **March 2021** | | |



**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

|  |  |
| --- | --- |
| **CERTIFICATE** | |
| This is to certify that the project report titled “**ATM Transaction System”** is being submitted by **N** **Vamshee Teja (2453-18-733-164)**, **P Jashwanth (2453-18-733-167), V Manjunath (2453-18-733-177), B Anupama (2453-18-733-128)** of III year B.E.V Semester **Computer Science and Engineering** is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree. | |
| **Internal Guide** | **HOD** |
|  | **External Examiner** |

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

# DECLARATION



|  |  |
| --- | --- |
| We hereby declare that the Mini Project Report entitled, “**ATM Transaction System”** submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree. | |
| **Date:** |  |
| **N Vamshee Teja**  **(2453-18-733-164)**  **P Jashwanth**  **(2453-18-733-167)** | **V Manjunath**  **(2453-18-733-177)**  **B Anupama**  **(2453-18-733-128)** |

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**ACKNOWLEDGEMENT**

We are happy to express our deep sense of gratitude to the principal of the college **Dr. D Jaya Prakash,** Professor, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank **Dr. K.V.Ranga Rao,** Professor and Head, Department of Computer Science and Engineering,Neil Gogte Institute of Technology , for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We are very grateful to our project guide “**Mrs. Vijaya Madhavi”,** Designation, Department of Computer Science and Engineering, Neil Gogte Institute of Technology, for her extensive patience and guidance throughout our project work.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

We would also like to thank classmates for always being there whenever we needed help or moral support. With great respect and obedience, We thank our parents, sisters and brother who were the backbone behind our deeds.

Finally, We express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**



# CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **S NO.** |  | **TITLE** | **PAGE NO.** |
|  |  | |  |
| **1** | **INTRODUCTION** | | **1** |
|  | 1.1 | Problem Statement | 1 |
|  | 1.2 | Solution | 1 |
|  | 1.3 | Objective | 2 |
| **2** | **LITERATURE SURVEY** | | **3** |
|  | 2.1 | Surveys | 3 |
|  | 2.2 Existing System | |  |
|  | 2.3 | Disadvantages of Existing System | 4 |
|  | 2.4 | Proposed System | 4 |
|  | 2.5 Advantages of Proposed System | | 5 |
|  | 2.6 Conclusion | | 5 |
| **3** | **ANALYSIS** | | **6** |
|  | 3.1 | Software and Hardware Requirements | 6 |
|  | 3.2 | Content Diagrams  a) Flow Diagram | 7 |
|  |  | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **DESIGN** | | **8** |
|  | 4.1 | Class Diagram | 8 |
|  | 4.2  4.3  4.4 | Use Case Diagram  Activity Diagram  ER Diagram | 8  9  9 |
| **5** | **IMPLEMENTATION** | | **10** |
|  | 5.1 | Source code | 10 |
| **6** | **TESTING AND VALIDATION** | | **30** |
|  | 6.1 | Project Screenshots | 30 |
| **7** | **CONCLUSION AND FUTURE ENHANCEMENTS** | | **35** |
| **8** | **REFERENCES** | |  |

**List Figures**

|  |  |
| --- | --- |
| **Fig. no.** | **Page no.** |
| fig 3.2.1 | 7 |
| fig 4.1.1 | 8 |
| fig 4.1.2 | 8 |
| fig 4.1.3 | 9 |
| fig 4.1.4 | 9 |

**Abstract**

The proposed Python project is an engineering approach to enhance current banking activities. The software works as a controller of the ATM machine during transactions. The implementation of project is beneficial to both the [bank](https://www.codewithc.com/online-bank-management-system-project-java/)s and the costumers. The **ATM Banking System** is the project by which the clients can create bank accounts and perform general cash transactions like Deposits, Withdrawals etc. The Bank System has 3 modes: 1) Sign-up mode, 2) Atm mode and 3) Exit mode. In Sign-up mode the user has to enter all the personal information like mobile no, Aadhar card number, PAN card number and related details and has to deposit a minimum amount of Rs. 1000 to create a Bank Account (Savings). After all the details of the client identity is verified, his/her account will be created and will be given a pin number which is useful for making future credit and debit transactions through atm. The second mode is the Atm mode in which functions equivalent to a normal atms like withdrawals, deposits and bank balance status. The ATM will service one customer at a time. A customer will be required to enter personal identification number (PIN) – which will be sent to the database for validation as part of each transaction. The customer will then be able to perform one or more transactions. The ATM will communicate each transaction to the database and obtain verification that it was allowed by the database. In the case of a cash withdrawal, a second message will be sent after the transaction has been physically completed (cash dispensed or envelope accepted).  If the database determines that the customer’s PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed.

1. **INTRODUCTION**

An Automatic Teller Machine (ATM) is a Specialized Computer that allows bank account holders to: check their account balances, withdraw or deposit money, transfer money from one account to another, print a statement of bank transactions and etc.

**ATM Evolution**

1960: Luther George Simjian (America) invented the Bankograph (a machine that allowed customers to deposit cash and cheques into it). 1967 (Worlds First ATM was set up): First ATM was set up in June 1967 on a street in Enfield, London. John Shepherd Barron (British) is credited with its invention. By 1984 ATMs installed worldwide were 1,000,000 (1 million). As of 2018 ATMs installed worldwide were more than 3 million.

* 1. **Problem Statement**

ATMs are convenient, allowing customers to perform quick self-service transactions such as deposits, cash withdrawals, bill payments, and transfers between accounts. Fees are commonly charged when cash withdrawal is done from bank by the operator of the ATM, or by both. Some or all of these fees can be avoided by using an ATM directly by the bank that holds the account. We can specify ATM transaction as a machine that allows customers to complete basic transactions without the aid of a branch representative, but due to no assistance from anyone our data can be stolen by Cyber Criminals which is vastly a disadvantage. It is very important for cardholders to protect their cards from being misused.

* 1. **Solution**

There are so many incidents which reveal that the Cash has been drawn out without prior knowledge of the Cash holder. So, to minimize the risk of cheating we as a group came up with an OTP alert for every transaction made by the

Customer, i,e. a OTP will be sent to a customer before performing a transaction through atm. Details of every transaction made by the Client is recorded at any basis.

* 1. **Objective**

Our main objective is to speed up the transaction done by customers in a very safe, secure and convenient way. The second objective is to save the time which is very important now-a-days. It will include other objectives such as: To render accurate services to customer, the reduction of fraudulent activities, and to achieve speedy processing of customer data

1. **LITERATURE SURVEY**

**2.1 Survey**

The growth of Indian economic system in the past decade is found to grow at a rapid rate. Banking industries in the financial sectors are introducing a new concept on a regular basis to attract the customers. The first ATM in India was presented by HSBC bank in 1987 at Mumbai branch for withdrawal. The ATM was introduced with an objective to serve the customers during emergency situations where cash deposits and withdrawals after regular banking hours are required. The next development in the ATM field is the introduction of an enquiry system to know the account balance and statement so that the customers do not waste time waiting inside the bank premises. Many significant changes like account transfer from one-person account to another account holders, requisition claims like chequebook need, message alert etc. were noted after 2000. After 2010, technological developments increased in ATM non-banking services. Such services include bill payments, ticket bookings, mobile recharges, etc. Even though many developments have been introduced in the sector, much more should be brought in to increase the quality of ATM services in India.

**2.2 Existing System**

Most people of a certain age know how to use an ATM without a problem. But few people have a great understanding of the process that happens behind the scenes that make bank transactions via an ATM possible. Fortunately, transaction processing is not as complicated as you might think and not all that hard to understand. First, the user will swipe his or her ATM card and enter the pin number associated with that card. This confirms the cardholder’s identity and allows him or her to request a bank transaction, usually a withdrawal of money. The machine then contacts a host server with the cardholder’s information and transaction request. Years ago, this would be done through a telephone line. That is still the case for some machines, although more modern ATMs connect with host servers via the internet nowadays. The host server acts as an intermediary for contacting the bank or financial institution that issued the

ATM user his or her card. Once the user’s bank is contacted, the host server is able to facilitate an electronic transfer of the funds being requested by the cardholder, assuming the user’s home bank approves the transaction. The host server will then send the ATM an approval code that enables the machine to dispense the funds the cardholder requested. If requested, the ATM will also be able to share with the cardholder the balance in his or her account. Finally, the host processor uses an automated clearing house (ACH) to transfer funds from the cardholder’s account into the account of whatever entity owns the ATM, whether it be another bank or a business. This will typically happen the next business day and ensures that the party responsible for filling the ATM machine is reimbursed for the funds the cardholder just took out of the machine. Essentially, when you request money from an ATM, the money moves electronically from your account to the host server and then to the party that owns or operates the ATM. All of this happens within a matter of seconds, with important information like your pin and account number encrypted during the process for security reasons.

**2.3 Disadvantages of Existing System**

Present ATM Transaction process is not that safe for customers for their transaction because it has some loop holes like: if a person knows your ATM pin, he can have access for making a transaction and he may steal money from your account. And in a cut short the c problem is: If an **ATM** card is lost, it can be misused.

**2.4 Proposed System**

So, to resolve this problem we are adding an additional feature like OTP based transaction. Here, customer when making his transaction in the ATM an additional OTP is sent to his registered Mobile number. Here even if hackers have your Card-details they can't make a transaction because while making a transaction OTP is sent to your registered number. So, if the OTP verification fails, then the system doesn’t allow to make a transaction. This feature made the present ATM Transaction quite more secure.

**2.5 Advantages of Proposed System**

Sending a OTP to verified/registered mobile number in order to make a transaction makes it more secured. Even if your atm card and its information is lost, the hackers may fail in achieving their task.

**2.6 Conclusion**

So, by collaborating we could conclude that there is more to add for securing the ATM transactions Virtually to be hidden from the Hackers and people who are doing frauds stealing the personal information for illegal purposes etc.

**3. ANALYSIS**

**3.1 Software and Hardware Requirements**

**Hardware Requirements**

Processor: Intel Pentium IV 2.0 GHz and above

RAM: 512 MB and above

Hard disk :80GB and above

Monitor: CRT or LED monitor

**Software Requirements**

Language: Python

Backend: Python Programming Language

Frontend: Tkinter module (Python)

Database: Sqlite3

OTP: Twilio API

**3.2 Content Diagrams**

**a) Flow Diagram**

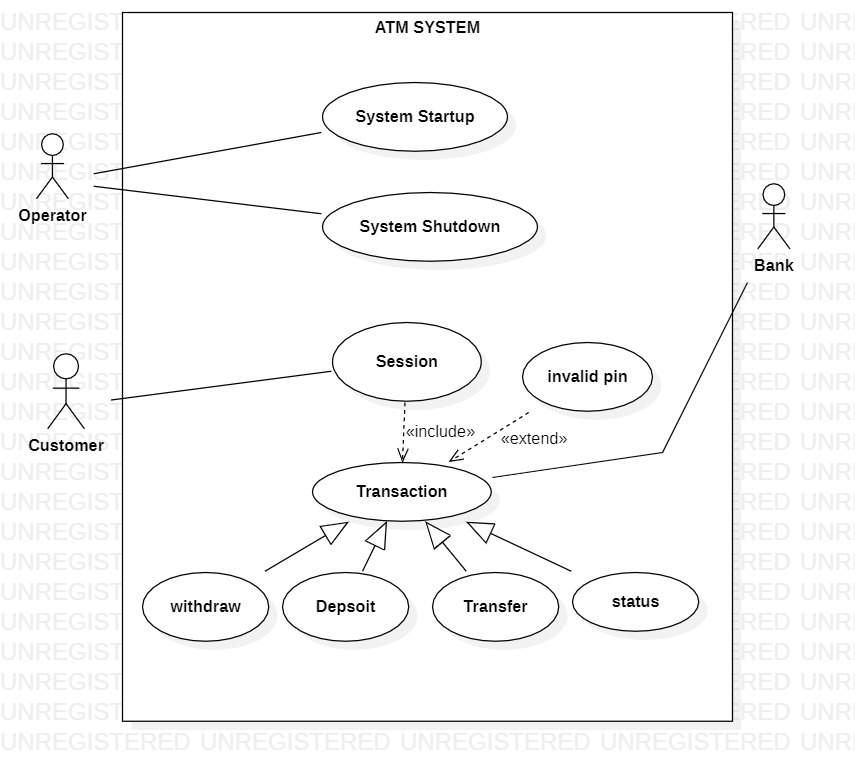


fig 3.2.1

**4. DESIGN**

**4.1 Class Diagram**

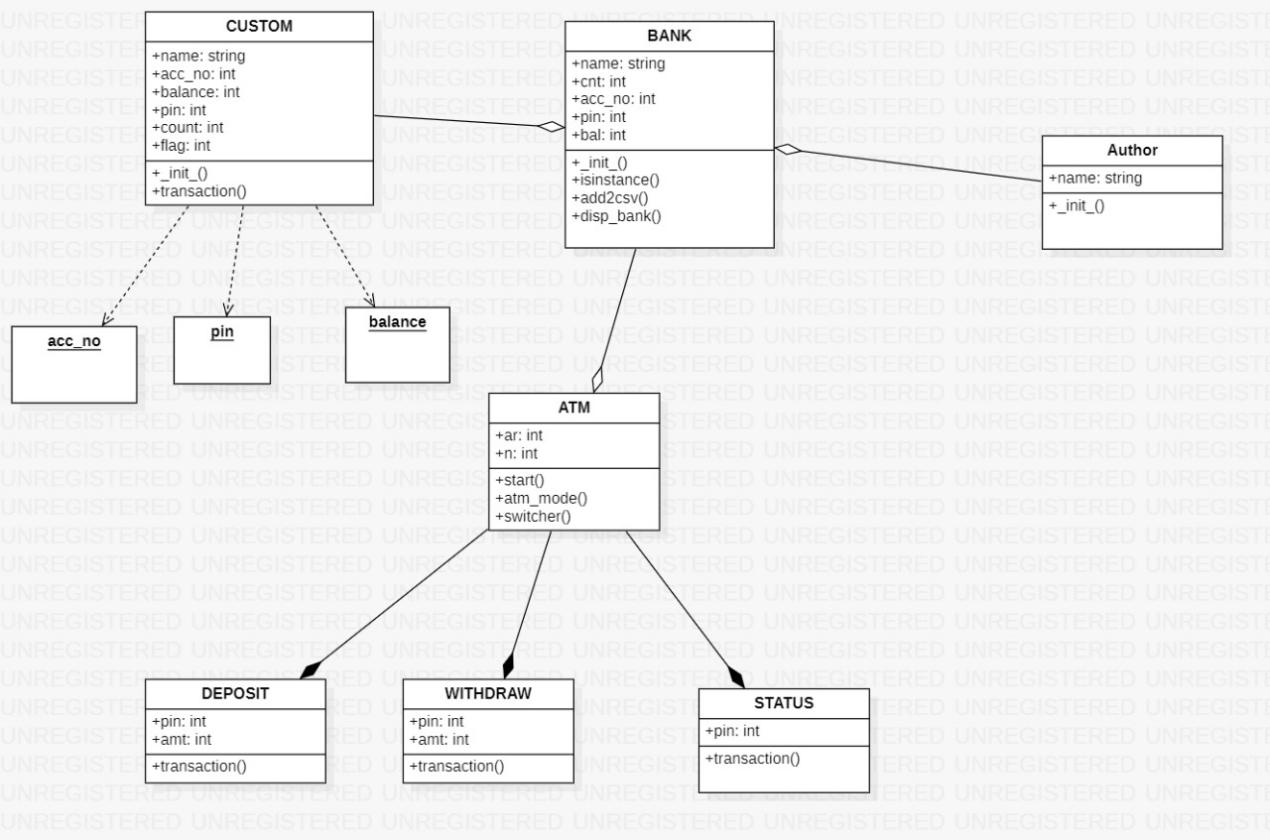


fig 4.1.1

**4.2 Use Case Diagram**

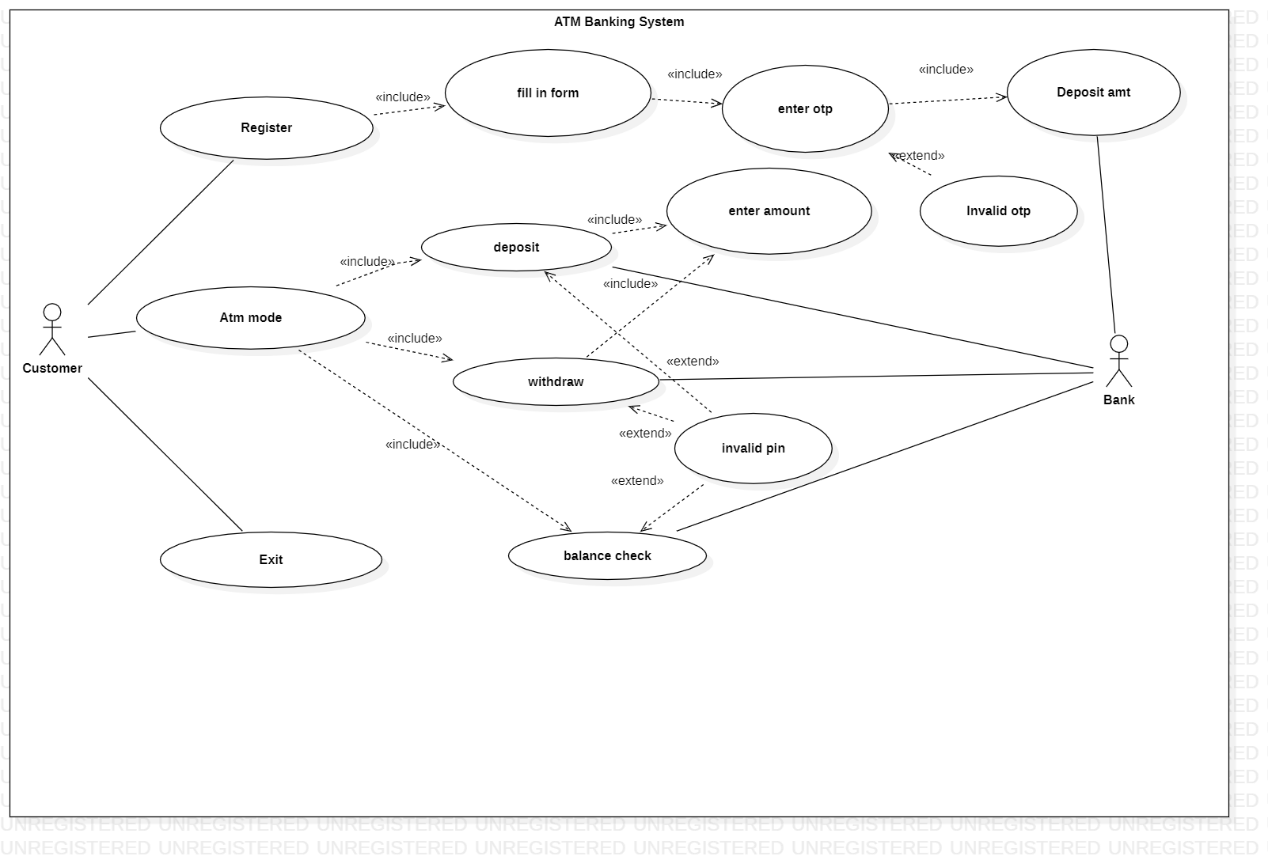


fig 4.1.2

**4.3 Use Case Diagram**

For particular withdraw method activity diagram

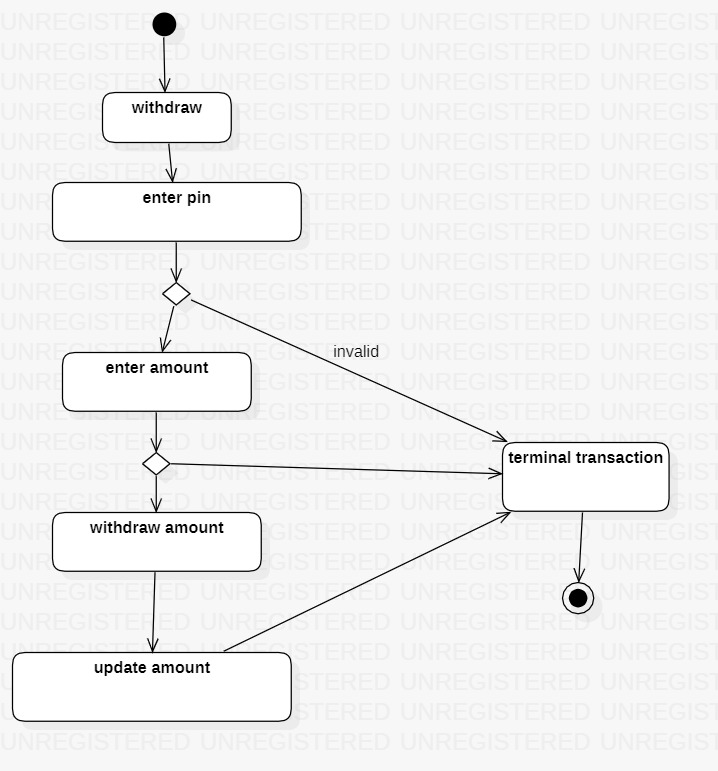
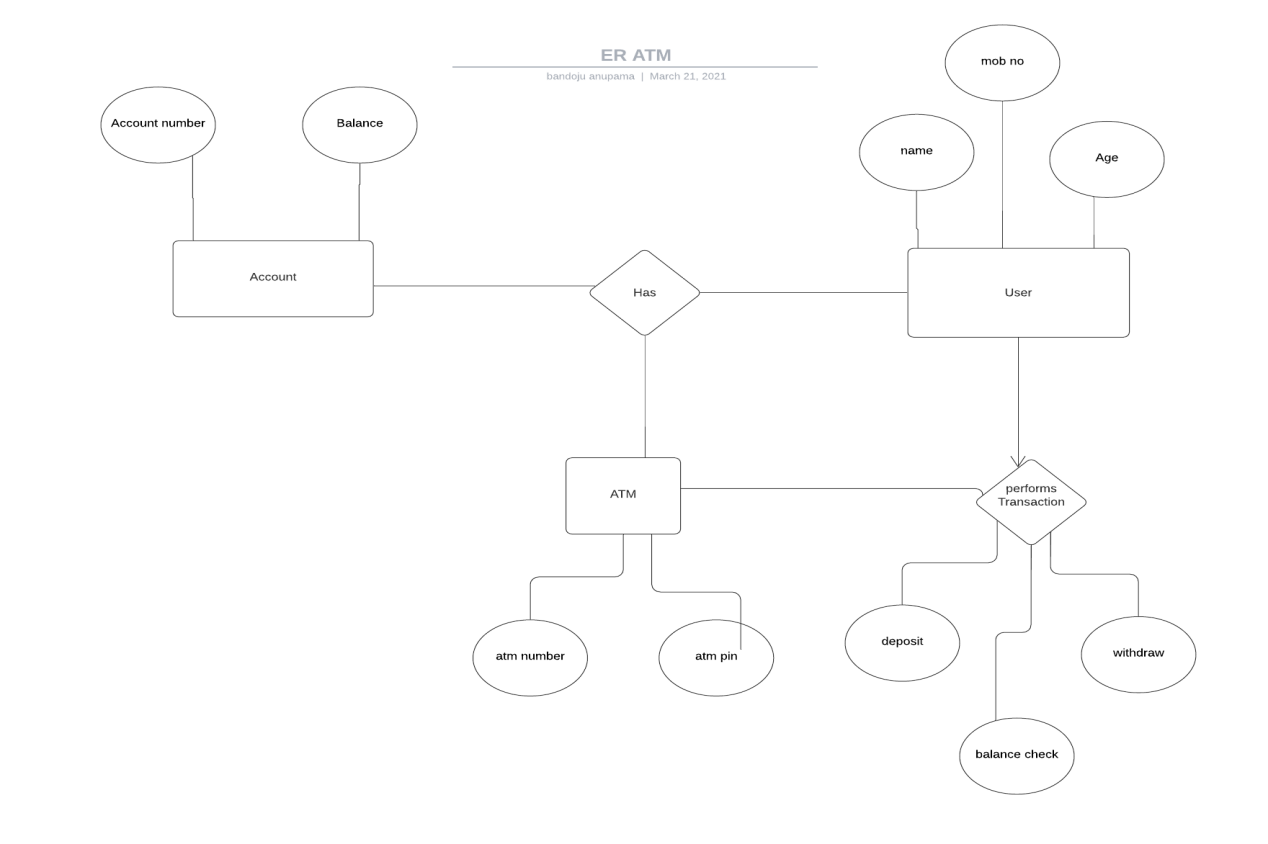


fig 4.1.3

**4.4 ER Diagram**

fig 4.1.4

**5. IMPLEMENTATION**

**5.1 Source Code**

''' Note: all ui & backend is present in this single file. Apologies for the clumbsy code. '''

# this project is completely based on Python OOP concepts and some knowledge of tkinter for GUI in python

# UI imports

from tkinter import \*

from PIL import ImageTk, Image

# Back-end imports

import sys

import random

import sqlite3

from sqlite3 import Error

# extra imports

from twilio.rest import Client

import os

class Bank():

# the sqlite database

@staticmethod

def create\_connection(db\_file):

""" creates a db connection and returns connection object """

conn = None

try:

conn = sqlite3.connect(db\_file)

return conn

except Error as e:

print(e)

return conn

# creates the table

@staticmethod

def create\_table(conn, create\_table\_sql):

""" utility function for creating a table """

try:

c = conn.cursor()

c.execute(create\_table\_sql)

except Error as e:

print(e)

finally:

c.close()

@staticmethod

def main():

database = r"pysqlite.db"

sql\_create\_bank\_accounts = """

CREATE TABLE IF NOT EXISTS bank\_accounts (

account\_no INTEGER,

pin INTEGER,

name TEXT,

age INTEGER,

phone\_no INTEGER,

balance REAL

);

""" # Schema : acc\_no, pin, name, age, phone\_no, balance

conn = Bank.create\_connection(database)

if conn is not None:

Bank.create\_table(conn, sql\_create\_bank\_accounts)

else:

print("Error! cannot create the database connection.")

@staticmethod

def otp\_opr(v\_fl, phone\_number=0, otp=0):

""" this is the otp generator code """

account\_sid = 'ACf30a1fed8d98a3f1576de13e511db950'

auth\_token = '2c3914ae241f4d9af34f5f7aaeb40089'

client = Client(account\_sid, auth\_token)

verify = client.verify.services('VA3f98da905f54dc5f728300d7321f6ef4')

if v\_fl == "send\_otp":

# print("sent otp")

verify.verifications.create(to=phone\_number, channel='sms')

return

elif v\_fl == "match\_otp":

# print("matching otp")

result = verify.verification\_checks.create(to=phone\_number, code=otp)

return result.status

def \_\_init\_\_(self, acc\_no=None, pin=None, name=None, age=None, phone\_no=None, balance=None):

if(isinstance(self, Custom)): # creates a object for each new entry

self.acc = acc\_no

self.pin = pin

self.name = name

self.age = age

self.phone = phone\_no

self.bal = balance

self.add2database(self.acc, self.pin, self.name, self.age, self.phone, self.bal)

def add2database(self, acc, pin, name, age, phone, bal):

""" this method inserts the data into the sqlite db """

# print("in add2database...")

# connecting to sqlite db

conn = Bank.create\_connection("pysqlite.db")

# creating a cursor object using the cursor() method

cursor = conn.cursor()

# query /insertion

cursor.execute("INSERT INTO bank\_accounts (account\_no, pin, name, age, phone\_no, balance) VALUES (?, ?, ?, ?, ?, ?)", (self.acc, self.pin, self.name, self.age, self.phone, self.bal))

conn.commit()

conn.close()

# whenever this application is started this get clicked and as you can see it call the main method (in the bank class) and creates the sql table.

if \_\_name\_\_ == '\_\_main\_\_':

Bank.main() # as you can see that the main() function is defined inside the class 'Bank'. we have declared it as a @staticmethod decorator, so that we can call it outside the class without creating an object.

class Custom(Bank):

# counter = 0

# print("in Custom class")

def \_\_init\_\_(self, acc\_no, epin, name, age, phone\_no, balance):

# print("in Custom constructor")

super().\_\_init\_\_(acc\_no, epin, name, age, phone\_no, balance)

# transactions method.

@staticmethod

def transaction(fl, pin, amt=0):

def checkPin(pin):

conn = Bank.create\_connection("pysqlite.db")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM bank\_accounts WHERE pin=?", (pin,))

rows = cursor.fetchall()

conn.close()

if len(rows) == 0:

return False

return True

if fl == "pin\_chk":

return checkPin(pin)

def fetch(fl, pin):

# connecting to sqlite db

conn = Bank.create\_connection("pysqlite.db")

# creating a cursor object using the cursor() method

cursor = conn.cursor()

# fetching the data (i,e. row) from the database

cursor.execute("SELECT \* FROM bank\_accounts WHERE pin=?", (pin,))

rows = cursor.fetchall()

conn.close()

# returns the ac\_number, ac\_holder, ac\_balance

return rows[0][0], rows[0][2], rows[0][5]

def updateBalance(pin, uamt):

""" used for deposits and withdrawals """

conn = Bank.create\_connection("pysqlite.db")

cursor = conn.cursor()

cursor.execute("UPDATE bank\_accounts SET BALANCE = ? WHERE pin = ?", (uamt, pin))

conn.commit()

conn.close()

# all else : msg is printed if pin not found in db

# deposit opr

if fl=='d':

if(checkPin(pin)):

# deposit message

dep\_msg\_screen = Toplevel(master)

ac\_no, ac\_name, curr\_amt = fetch(fl, pin)

upd\_amt = int(curr\_amt) + int(amt)

updateBalance(pin, upd\_amt)

msg = "Transaction Successful" + "\nDeposited Rs." + str(amt) + " to Account Number: "+ str(ac\_no) +", Account Holder: " + str(ac\_name) + " New Balance: " + str(upd\_amt)

Label(dep\_msg\_screen, text=msg, font=('Calibri', 14)).grid(row=1, sticky=N, pady=10)

# withdraw opr

elif fl=='w':

if(checkPin(pin)):

# witd screen

witd\_msg\_screen = Toplevel(master)

ac\_no, ac\_name, curr\_amt = fetch(fl, pin)

if ((int(curr\_amt) - int(amt)) <= 500):

msg = "Sorry you can't make that transaction! your current balance is: " + str(curr\_amt) + " minimum balance in bank shouldn't be less than 500"

Label(witd\_msg\_screen, text=msg, font=('Calibri', 14)).grid(row=1, sticky=N, pady=10)

return

upd\_amt = int(curr\_amt) - int(amt)

updateBalance(pin, upd\_amt)

# withdraw message

msg = "Transaction Successful" + "\nRs." + str(amt) + " withdrawn from Account Number: "+ str(ac\_no) +", Account Holder: " + str(ac\_name) + " New Balance: " + str(upd\_amt)

Label(witd\_msg\_screen, text=msg, font=('Calibri', 14)).grid(row=1, sticky=N, pady=10)

# status opr

elif fl == 's':

if(checkPin(pin)):

# balance-check screen

bal\_chk\_msg\_screen = Toplevel(master)

ac\_no, ac\_name, curr\_amt = fetch(fl, pin)

# bal check msg

msg = "Account Number: " + str(ac\_no) + "\nAccount Holder: " + str(ac\_name) + "\nNew Balance: " + str(curr\_amt)

Label(bal\_chk\_msg\_screen, text=msg, font=('Calibri', 14)).grid(row=1, sticky=N, pady=10)

# main screen

master = Tk()

master.title("Banking App")

# functions

def dep\_go():

pin = temp\_spin.get()

amt = temp\_samt.get()

global notif2\_

notif2 = Label(deposit\_screen, font=('Calibri', 12))

notif2\_ = Label(deposit\_screen, font=('Calibri', 12))

notif2.grid(row=6, sticky=N, pady=10)

notif2\_.grid(row=7, sticky=N, pady=10)

if pin == "" or amt == "":

notif2.config(fg="red", text="All fields required\*")

elif Custom.transaction("pin\_chk", pin) == False:

notif2\_.config(fg="red", text="Invalid PIN\*")

else:

deposit\_screen.destroy()

Custom.transaction('d', pin, amt)

# deposit

def dep():

global temp\_spin

global temp\_samt

temp\_spin = StringVar()

temp\_samt = StringVar()

# deposit\_screen

global deposit\_screen

deposit\_screen = Toplevel(master)

deposit\_screen.title("deposit amount")

Label(deposit\_screen, text="Please enter your pin and deposit amount", font=('Calibri', 12)).grid(row=0, sticky=N, pady=10)

Label(deposit\_screen, text="Pin", font=('Calibri', 12)).grid(row=2, sticky=W)

Label(deposit\_screen, text="Amount", font=('Calibri', 12)).grid(row=3, sticky=W)

# entries

Entry(deposit\_screen, textvariable=temp\_spin, show="\*").grid(row=2, column=1)

Entry(deposit\_screen, textvariable=temp\_samt).grid(row=3, column=1)

# buttons

Button(deposit\_screen, text="Submit", command = lambda:[dep\_go()], font=("Calibri", 12)).grid(row=5, sticky=N, pady=10)

# withdraw

def witd\_go():

pin = temp\_spin.get()

amt = temp\_samt.get()

global notif3\_

notif3 = Label(withdraw\_screen, font=('Calibri', 12))

notif3\_ = Label(withdraw\_screen, font=('Calibri', 12))

notif3.grid(row=6, sticky=N, pady=10)

notif3\_.grid(row=7, sticky=N, pady=10)

if pin == "" or amt == "":

notif3.config(fg="red", text="All fields required\*")

elif Custom.transaction("pin\_chk", pin) == False:

notif3\_.config(fg="red", text="Invalid PIN\*")

else:

withdraw\_screen.destroy()

Custom.transaction('w', pin, amt)

def witd():

global temp\_spin

global temp\_samt

temp\_spin = StringVar()

temp\_samt = StringVar()

# withdraw\_screen

global withdraw\_screen

withdraw\_screen = Toplevel(master)

withdraw\_screen.title("withdraw amount")

Label(withdraw\_screen, text="Please enter your pin and amount to be withdrawn", font=('Calibri', 12)).grid(row=0, sticky=N, pady=10)

Label(withdraw\_screen, text="Pin", font=('Calibri', 12)).grid(row=2, sticky=W)

Label(withdraw\_screen, text="Amount", font=('Calibri', 12)).grid(row=3, sticky=W)

# entries

Entry(withdraw\_screen, textvariable=temp\_spin, show="\*").grid(row=2, column=1)

Entry(withdraw\_screen, textvariable=temp\_samt).grid(row=3, column=1)

# buttons

Button(withdraw\_screen, text="Submit", command = lambda:[witd\_go()], font=("Calibri", 12)).grid(row=5, sticky=N, pady=10)

# status/ Balance-check

def stat\_go():

pin = temp\_spin.get()

global notif4\_

notif4 = Label(stat\_screen, font=('Calibri', 12))

notif4\_ = Label(stat\_screen, font=('Calibri', 12))

notif4.grid(row=6, sticky=N, pady=10)

notif4\_.grid(row=7, sticky=N, pady=10)

if pin == "":

notif4.config(fg="red", text="field required\*")

elif Custom.transaction("pin\_chk", pin) == False:

notif4\_.config(fg="red", text="Invalid PIN\*")

else:

stat\_screen.destroy()

Custom.transaction('s', pin)

def stat():

global temp\_spin

temp\_spin = StringVar()

# bal\_check\_screen

global stat\_screen

stat\_screen = Toplevel(master)

stat\_screen.title("Balance Check")

Label(stat\_screen, text="Please enter your pin", font=('Calibri', 12)).grid(row=0, sticky=N, pady=10)

Label(stat\_screen, text="Pin", font=('Calibri', 12)).grid(row=2, sticky=W)

# entries

Entry(stat\_screen, textvariable=temp\_spin, show="\*").grid(row=2, column=1)

# buttons

Button(stat\_screen, text="Submit", command = lambda:[stat\_go()], font=("Calibri", 12)).grid(row=4, sticky=N, pady=10)

# finish register

def finish\_reg():

name = temp\_name.get().upper()

age = temp\_age.get()

phone = temp\_phone.get()

damt = temp\_damt.get()

if damt == "" or int(damt) < 500:

fv\_notif.config(fg="red", text="enter amount")

return

finish\_verify\_screen.destroy()

# finish\_register\_screen

finish\_reg\_screen = Toplevel(master)

finish\_reg\_screen.title("Success")

ac\_gen = "43518733" + str(random.randrange(10 \*\* 3, (10 \*\* 4)-1))

acc\_no = int(ac\_gen)

pin = int(random.randint((10 \*\* 3), (10 \*\* 4)-1))

Custom(acc\_no, pin, name, age, phone, damt)

Label(finish\_reg\_screen, text="your account has been created. Thanks for using PyBank!", font=('Calibri', 14)).grid(row=0, sticky=N, pady=10)

message = "Your Account Number: "+ str(acc\_no) +"\nAccount Holder: " + name + "\nPhone: "+ phone +"\nPin: " + str(pin)

Label(finish\_reg\_screen, text=message, font=('Calibri', 14)).grid(row=1, sticky=N, pady=10)

notif1 = Label(finish\_reg\_screen, font=('Calibri', 12))

notif1.grid(row=6, sticky=N, pady=10)

notif1.config(fg="red", text="WARNING! PLEASE DON'T SHARE YOUR PIN WITH ANYONE.")

# finish verification

def finish\_verify():

global temp\_damt

temp\_damt = StringVar()

phone = temp\_phone.get()

in\_otp = temp\_otp.get()

if in\_otp == "":

vnotif.config(fg="red", text="required\*")

return

ap = "+91"+phone

result = Bank.otp\_opr("match\_otp", phone\_number = ap, otp = in\_otp)

if result != "approved":

vnotif.config(fg="red", text="invalid\*")

return

verify\_screen.destroy()

global finish\_verify\_screen

# finish\_verify\_screen

# Label(register\_screen, text="Deposit amt (>= 1000)", font=('Calibri', 12)).grid(row=4, sticky=W)

finish\_verify\_screen = Toplevel(master)

finish\_verify\_screen.title("Verified Successfully")

# labels

Label(finish\_verify\_screen, text="phone number verified successfully", font=('Calibri', 8)).grid(row=0, sticky=W)

Label(finish\_verify\_screen, text="please enter amount to be deposited", font=('Calibri', 12)).grid(row=1, sticky=W)

Label(finish\_verify\_screen, text="Deposit amt (>= 500)", font=('Calibri', 12)).grid(row=2, sticky=W)

# fv\_notif

global fv\_notif

fv\_notif = Label(finish\_verify\_screen, font=('Calibri', 12))

fv\_notif.grid(row=4, sticky=N, pady=10)

# entries

Entry(finish\_verify\_screen, textvariable=temp\_damt).grid(row=2, column=1)

# buttons

Button(finish\_verify\_screen, text="Deposit", command = lambda:[finish\_reg()], font=("Calibri", 12)).grid(row=6, sticky=N, pady=10)

# verify screen

def verify():

global temp\_otp

temp\_otp = StringVar()

name = temp\_name.get().upper()

age = temp\_age.get()

phone = temp\_phone.get()

if name=="" or age=="" or phone=="":

notif.config(fg="red", text="All fields required\*")

return

if len(phone) != 10:

notif.config(fg="red", text="Enter a valid 10-digit phone number.\*")

return

# appending india code "+91"

ap = "+91"+phone

try:

Bank.otp\_opr("send\_otp", phone\_number = ap)

except :

print("in except")

emsg = "entered phone number is invalid \*/ if it is a valid, there might be a api problem. please try after some time. Sorry for the inconvenience"

notif.config(fg="red", text=emsg)

return

register\_screen.destroy()

global verify\_screen

# phone\_verify screen

verify\_screen = Toplevel(master)

# labels

v\_msg = "Please enter the 6-digit OTP sent to " + str(phone) + " to verify your Identity"

Label(verify\_screen, text=v\_msg, font=('Calibri', 12)).grid(row=1, sticky=W)

Label(verify\_screen, text="OTP", font=('Calibri', 12)).grid(row=3, sticky=W)

global vnotif

vnotif = Label(verify\_screen, font=('Calibri', 12))

vnotif.grid(row=3, sticky=N, pady=10)

# entries

Entry(verify\_screen, textvariable=temp\_otp).grid(row=3, column=1)

# button

Button(verify\_screen, text="Register", command = lambda:[finish\_verify()], font=("Calibri", 12)).grid(row=5, sticky=N, pady=10)

# register mode function

def register():

# vars

global temp\_name

global temp\_age

global temp\_phone

global notif

temp\_name = StringVar()

temp\_age = StringVar()

temp\_phone = StringVar()

# register screen

global register\_screen

register\_screen = Toplevel(master)

register\_screen.title("Register")

# labels

Label(register\_screen, text="Please enter your details below to register", font=('Calibri', 12)).grid(row=0, sticky=N, pady=10)

Label(register\_screen, text="Name", font=('Calibri', 12)).grid(row=1, sticky=W)

Label(register\_screen, text="Age", font=('Calibri', 12)).grid(row=2, sticky=W)

Label(register\_screen, text="Phone Number", font=('Calibri', 12)).grid(row=3, sticky=W)

notif = Label(register\_screen, font=('Calibri', 12))

notif.grid(row=6, sticky=N, pady=10)

# entries

Entry(register\_screen, textvariable=temp\_name).grid(row=1, column=1)

Entry(register\_screen, textvariable=temp\_age).grid(row=2, column=1)

Entry(register\_screen, textvariable=temp\_phone).grid(row=3, column=1)

# Entry(register\_screen, textvariable=temp\_damt).grid(row=4, column=1)

# button

Button(register\_screen, text="Register", command = lambda:[verify()], font=("Calibri", 12)).grid(row=8, sticky=N, pady=10)

# atm\_mode function

def atm():

global atm\_mode\_screen

# atm\_mode screen

atm\_mode\_screen = Toplevel(master)

atm\_mode\_screen.title("Atm Mode")

# labels

Label(atm\_mode\_screen, text = "ATM-Mode", font=('Calibri', 14)).grid(row=0, sticky=N, pady=10)

Label(atm\_mode\_screen, text = "select the option", font=('Calibri', 12)).grid(row=0, sticky=N, pady=10)

# buttons

Button(atm\_mode\_screen, text="Deposit", command = lambda:[dep(), atm\_mode\_screen.destroy()], font=("Calibri", 12), width=20).grid(row=2)

Button(atm\_mode\_screen, text="Withdraw", command = lambda:[witd(), atm\_mode\_screen.destroy()], font=("Calibri", 12), width=20).grid(row=3)

Button(atm\_mode\_screen, text="Balance Check", command = lambda:[stat(), atm\_mode\_screen.destroy()] , font=("Calibri", 12), width=20).grid(row=4, sticky=N)

# exit mode

def exit():

sys.exit()

# Main page #

# image

img = Image.open("pyBank\_logo.png")

img = img.resize((250, 250))

img = ImageTk.PhotoImage(img)

# Labels

Label(master, text = "Automatic Teller Machine", font=('Calibri', 14)).grid(row=0, sticky=N, pady=10)

Label(master, image = img).grid(row=1, sticky=N, pady=15)

# Buttons

Button(master, text="Register", font=('Calibri', 12), width=20, command=register).grid(row=3)

Button(master, text="Atm Mode", font=('Calibri', 12), width=20, command=atm).grid(row=4)

Button(master, text="Exit", font=('Calibri', 12), width=20, command=exit).grid(row=5, sticky=N)

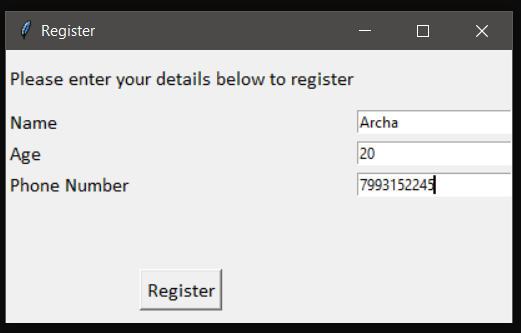
master.mainloop()

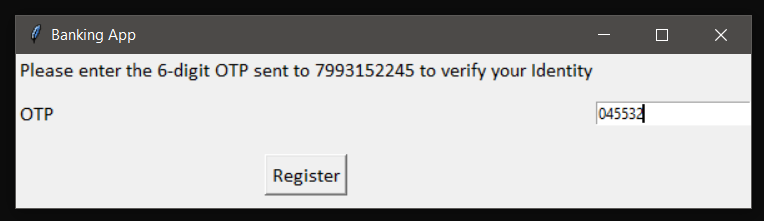
**6. TESTING AND VALIDATION**

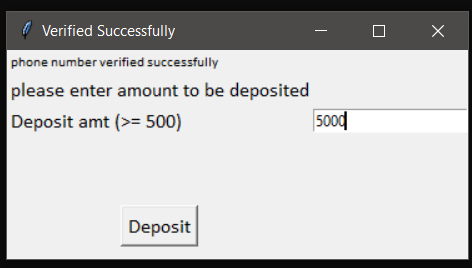
During the software-testing phase each module of software is thoroughly tested for bugs and for accuracy of output. The system developed is very user-friendly and the detailed documentation is also given to the user as online help wherever necessary.  The implementation phase normally ends with the formal test involving all the components.

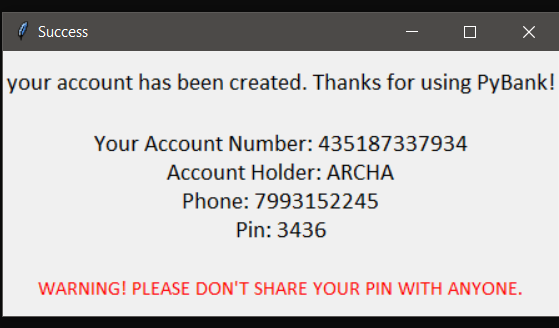
**6.1 Project Screenshots**

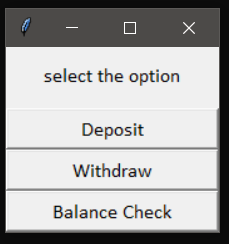


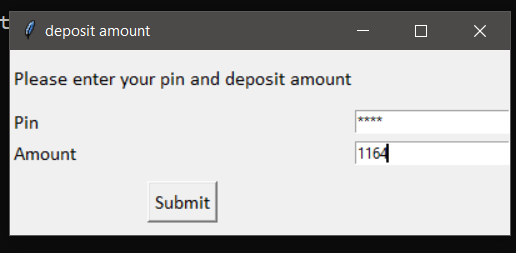


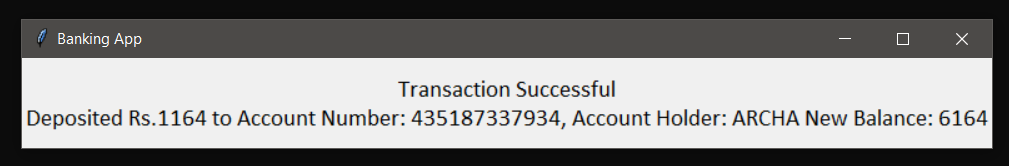


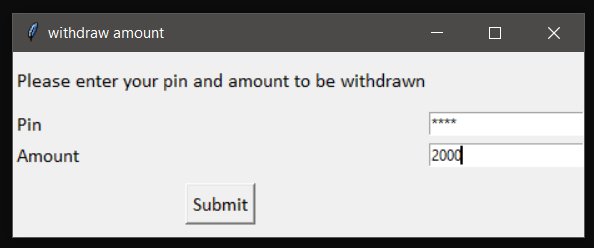


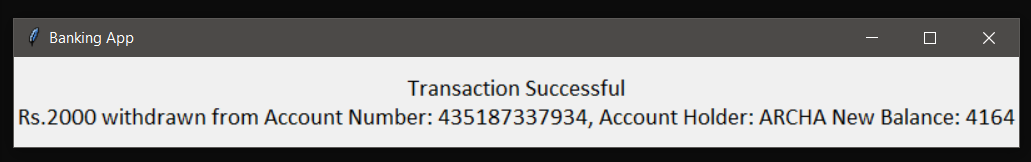


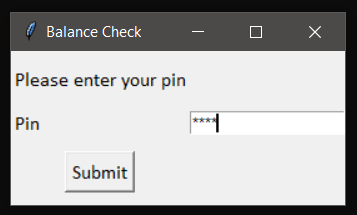


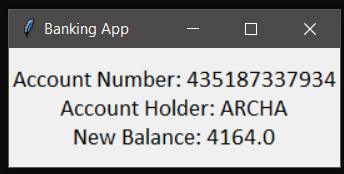












1. **CONCLUSION AND FUTURE ENHANCEMENTS**

As the banking sector computerize day by day, and atm have become a part of modern banking system. The banks in developing country adopt ATMs to improve their own internal process and also for increase facilities and services of their customers. Now customers become aware about this machine. The growth of a ATM rapidly high at the world wide level also in India. This technology is simple, safe and secure. By this using OTP generating process we can easily secure ATM from frauds. System will automatically generate a one- time password (OTP) and send to the registered mobile number. It will ask user to enter the OTP. If it matches an authorized access will be granted.

1. **REFERENCES**

[1] Python: <https://www.javatpoint.com/python-oops-concepts>

[2] Tkinter: <https://www.youtube.com/watch?v=71X58zIzrgA&t=1355s>,

<https://www.tutorialspoint.com/python/python_gui_programming.htm>

[3] twilio (for OTP): <https://www.twilio.com/blog/phone-verification-with-twilio-for-python-developers>

[4] Articles and related blogs: <https://financialyard.com/advantages-and-disadvantages-of-atm-automated-teller-machines/>,

<https://www.scribd.com/doc/133970440/Project-Report-on-ATM-System>,

<https://www.researchgate.net/publication/322428014_Growth_and_Development_of_ATM_in_India>